Part I - (Dataset Exploration Title)

by (Francis Kipkogei)

Introduction

> About the dataset This databset utilized in this project is prosperLoan Data which was obtained from https://s3.amazonaws.com/udacity-hosted-downloads/ud651/prosperLoanData.csv

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.

> **Variable Description**

The Variable descriptionis provided on https://www.google.com/url?q=https://docs.google.com/spreadsheet/ccc?key%3D0AllIqIyvWZdadDd5NTlqZ1pBMHlsUjdrOTZHaVBuSlE%26usp%3Dsharing&sa=D&source=editors&

- `ListingKey` : Unique key for each listing, same value as the 'key' used in the listing object in the API.
- `ListingNumber` : The number that uniquely identifies the listing to the public as displayed on the website.
- `ListingCreationDate` : The date the listing was created.
- `CreditGrade` : The Credit rating that was assigned at the time the listing went live. Applicable for listings pre-2009 period and will only be populated for those listings.
- `Term` : The length of the loan expressed in months.
- `LoanStatus` : The current status of the loan: Cancelled, Chargedoff, Completed, Current, Defaulted, FinalPaymentInProgress, PastDue. The PastDue status will be accompanied by a delinquency bucket.
- `ClosedDate` : Closed date is applicable for Cancelled, Completed, Chargedoff and Defaulted loan statuses.
- `BorrowerAPR` : The Borrower's Annual Percentage Rate (APR) for the loan.
- `BorrowerRate` : The Borrower's interest rate for this loan.
- `LenderYield` : The Lender yield on the loan. Lender yield is equal to the interest rate on the loan less the servicing fee.
- `EstimatedEffectiveYield` : Effective yield is equal to the borrower interest rate (i) minus the servicing fee rate, (ii) minus estimated uncollected interest on charge-offs, (iii) plus estimated collected late fees. Applicable for loans originated after July 2009.
- `EstimatedLoss` : Estimated loss is the estimated principal loss on charge-offs. Applicable for loans originated after July 2009.
- `EstimatedReturn` : The estimated return assigned to the listing at the time it was created. Estimated return is the difference between the Estimated Effective Yield and the Estimated Loss Rate. Applicable for loans originated after July 2009.
- `ProsperRating (numeric)` : The Prosper Rating assigned at the time the listing was created: 0 - N/A, 1 - HR, 2 - E, 3 - D, 4 - C, 5 - B, 6 - A, 7 - AA. Applicable for loans originated after July 2009.

- `ProsperRating (Alpha)` : The Prosper Rating assigned at the time the listing was created between AA - HR. Applicable for loans originated after July 2009.
- `ProsperScore` : A custom risk score built using historical Prosper data. The score ranges from 1-10, with 10 being the best, or lowest risk score. Applicable for loans originated after July 2009.
- `ListingCategory` : The category of the listing that the borrower selected when posting their listing: 0 - Not Available, 1 - Debt Consolidation, 2 - Home Improvement, 3 - Business, 4 - Personal Loan, 5 - Student Use, 6 - Auto, 7- Other, 8 - Baby&Adoption, 9 - Boat, 10 - Cosmetic Procedure, 11 - Engagement Ring, 12 - Green Loans, 13 - Household Expenses, 14 - Large Purchases, 15 - Medical/Dental, 16 - Motorcycle, 17 - RV, 18 - Taxes, 19 - Vacation, 20 - Wedding Loans
- `BorrowerState` : The two letter abbreviation of the state of the address of the borrower at the time the Listing was created.
- `Occupation` : The Occupation selected by the Borrower at the time they created the listing.
- `EmploymentStatus` : The employment status of the borrower at the time they posted the listing.
- `EmploymentStatusDuration` : The length in months of the employment status at the time the listing was created.
- `IsBorrowerHomeowner` : A Borrower will be classified as a homowner if they have a mortgage on their credit profile or provide documentation confirming they are a homeowner.
- `CurrentlyInGroup` : Specifies whether or not the Borrower was in a group at the time the listing was created.
- `GroupKey` : The Key of the group in which the Borrower is a member of. Value will be null if the borrower does not have a group affiliation.
- `DateCreditPulled` : The date the credit profile was pulled.
- `CreditScoreRangeLower` : The lower value representing the range of the borrower's credit score as provided by a consumer credit rating agency.
- `CreditScoreRangeUpper` : The upper value representing the range of the borrower's credit score as provided by a consumer credit rating agency.
- `FirstRecordedCreditLine` : The date the first credit line was opened.
- `CurrentCreditLines` : Number of current credit lines at the time the credit profile was pulled.
- `OpenCreditLines` : Number of open credit lines at the time the credit profile was pulled.
- `TotalCreditLinespast7years` : Number of credit lines in the past seven years at the time the credit profile was pulled.
- `OpenRevolvingAccounts` : Number of open revolving accounts at the time the credit profile was pulled.
- `OpenRevolvingMonthlyPayment` : Monthly payment on revolving accounts at the time the credit profile was pulled.
- `InquiriesLast6Months` : Number of inquiries in the past six months at the time the credit profile was pulled.
- `TotalInquiries` : Total number of inquiries at the time the credit profile was pulled.
- `CurrentDelinquencies` : Number of accounts delinquent at the time the credit profile was pulled.
- `AmountDelinquent` : Dollars delinquent at the time the credit profile was pulled.
- `DelinquenciesLast7Years` : Number of delinquencies in the past 7 years at the time the credit profile was pulled.
- `PublicRecordsLast10Years` : Number of public records in the past 10 years at the time the credit profile was pulled.
- `PublicRecordsLast12Months` : Number of public records in the past 12 months at the time the credit profile was pulled.
- `RevolvingCreditBalance` : Dollars of revolving credit at the time the credit profile was pulled.

- `BankcardUtilization` : The percentage of available revolving credit that is utilized at the time the credit profile was pulled.
- `AvailableBankcardCredit` : The total available credit via bank card at the time the credit profile was pulled.
- `TotalTrades` : Number of trade lines ever opened at the time the credit profile was pulled.
- `TradesNeverDelinquent` : Number of trades that have never been delinquent at the time the credit profile was pulled.
- `TradesOpenedLast6Months` : Number of trades opened in the last 6 months at the time the credit profile was pulled.
- `DebtToIncomeRatio` : The debt to income ratio of the borrower at the time the credit profile was pulled. This value is Null if the debt to income ratio is not available. This value is capped at 10.01 (any debt to income ratio larger than 1000% will be returned as 1001%).
- `IncomeRange` : The income range of the borrower at the time the listing was created.
- `IncomeVerifiable` : The borrower indicated they have the required documentation to support their income.
- `StatedMonthlyIncome` : The monthly income the borrower stated at the time the listing was created.
- `LoanKey` : Unique key for each loan. This is the same key that is used in the API.
- `TotalProsperLoans` : Number of Prosper loans the borrower at the time they created this listing. This value will be null if the borrower had no prior loans.
- `TotalProsperPaymentsBilled` : Number of on time payments the borrower made on Prosper loans at the time they created this listing. This value will be null if the borrower had no prior loans.
- `OnTimeProsperPayments` : Number of on time payments the borrower had made on Prosper loans at the time they created this listing. This value will be null if the borrower has no prior loans.
- `ProsperPaymentsLessThanOneMonthLate` : Number of payments the borrower made on Prosper loans that were less than one month late at the time they created this listing. This value will be null if the borrower had no prior loans.
- `ProsperPaymentsOneMonthPlusLate` : Number of payments the borrower made on Prosper loans that were greater than one month late at the time they created this listing. This value will be null if the borrower had no prior loans.
- `ProsperPrincipalBorrowed` : Total principal borrowed on Prosper loans at the time the listing was created. This value will be null if the borrower had no prior loans.
- `ProsperPrincipalOutstanding` : Principal outstanding on Prosper loans at the time the listing was created. This value will be null if the borrower had no prior loans.
- `ScorexChangeAtTimeOfListing` : Borrower's credit score change at the time the credit profile was pulled. This will be the change relative to the borrower's last Prosper loan. This value will be null if the borrower had no prior loans.
- `LoanCurrentDaysDelinquent` : The number of days delinquent.
- `LoanFirstDefaultedCycleNumber` : The cycle the loan was charged off. If the loan has not charged off the value will be null.
- `LoanMonthsSinceOrigination` : Number of months since the loan originated.
- `LoanNumber` : Unique numeric value associated with the loan.
- `LoanOriginalAmount` : The origination amount of the loan.
- `LoanOriginationDate` : The date the loan was originated.
- `LoanOriginationQuarter` : The quarter in which the loan was originated.
- `MemberKey` : The unique key that is associated with the borrower. This is the same identifier that is used in the API member object.
- `MonthlyLoanPayment` : The scheduled monthly loan payment.

- `LP_CustomerPayments` : Pre charge-off cumulative gross payments made by the borrower on the loan. If the loan has charged off, this value will exclude any recoveries.
- `LP_CustomerPrincipalPayments` : Pre charge-off cumulative principal payments made by the borrower on the loan. If the loan has charged off, this value will exclude any recoveries.
- `LP_InterestandFees` : Pre charge-off cumulative interest and fees paid by the borrower. If the loan has charged off, this value will exclude any recoveries.
- `LP_ServiceFees` : Cumulative service fees paid by the investors who have invested in the loan.
- `LP_CollectionFees` : Cumulative collection fees paid by the investors who have invested in the loan.
- `LP_GrossPrincipalLoss` : The gross charged off amount of the loan.
- `LP_NetPrincipalLoss` : The principal that remains uncollected after any recoveries.
- `LP_NonPrincipalRecoverypayments` : The interest and fee component of any recovery payments. The current payment policy applies payments in the following order: Fees, interest, principal.
- `PercentFunded` : Percent the listing was funded.
- `Recommendations` : Number of recommendations the borrower had at the time the listing was created.
- `InvestmentFromFriendsCount` : Number of friends that made an investment in the loan.
- `InvestmentFromFriendsAmount` : Dollar amount of investments that were made by friends.
- `Investors` : The number of investors that funded the loan.

Key Questions (Research Questions)

What factors affect a loan's outcome status?

What affects the borrower's APR or interest rate?

Are there differences between loans depending on how large the original loan amount was?

In [2]:
```
# import all packages and set plots to be embedded inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

%matplotlib inline
```

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

In [3]:
```
# Load our dataset using the pandas read_csv function
df = pd.read_csv('prosperLoanData.csv')
df.head(3)
```

Out[3]:

	ListingKey	ListingNumber	ListingCreationDate	CreditGrade	Term	LoanStatus	Closed
0	1021339766868145413AB3B	193129	2007-08-26 19:09:29.263000000	C	36	Completed	2009-(00:(
1	10273602499503308B223C1	1209647	2014-02-27 08:28:07.900000000	NaN	36	Current	
2	0EE9337825851032864889A	81716	2007-01-05 15:00:47.090000000	HR	36	Completed	2009- 00:(

3 rows × 81 columns

In [4]:
```
# Checking structure of the dataset (data types, number of columns, rows, missing values
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  float64
 33  InquiriesLast6Months                 113240 non-null  float64
 34  TotalInquiries                       112778 non-null  float64
 35  CurrentDelinquencies                 113240 non-null  float64
 36  AmountDelinquent                     106315 non-null  float64
 37  DelinquenciesLast7Years              112947 non-null  float64
 38  PublicRecordsLast10Years             113240 non-null  float64
 39  PublicRecordsLast12Months            106333 non-null  float64
 40  RevolvingCreditBalance               106333 non-null  float64
 41  BankcardUtilization                  106333 non-null  float64
 42  AvailableBankcardCredit              106393 non-null  float64
 43  TotalTrades                          106393 non-null  float64
 44  TradesNeverDelinquent (percentage)   106393 non-null  float64
 45  TradesOpenedLast6Months              106393 non-null  float64
 46  DebtToIncomeRatio                    105383 non-null  float64
 47  IncomeRange                          113937 non-null  object 
 48  IncomeVerifiable                     113937 non-null  bool   
 49  StatedMonthlyIncome                  113937 non-null  float64
 50  LoanKey                              113937 non-null  object 
 51  TotalProsperLoans                    22085 non-null   float64
 52  TotalProsperPaymentsBilled           22085 non-null   float64
 53  OnTimeProsperPayments                22085 non-null   float64
 54  ProsperPaymentsLessThanOneMonthLate  22085 non-null   float64
```

```
 55  ProsperPaymentsOneMonthPlusLate      22085 non-null   float64
 56  ProsperPrincipalBorrowed             22085 non-null   float64
 57  ProsperPrincipalOutstanding          22085 non-null   float64
 58  ScorexChangeAtTimeOfListing          18928 non-null   float64
 59  LoanCurrentDaysDelinquent            113937 non-null  int64
 60  LoanFirstDefaultedCycleNumber        16952 non-null   float64
 61  LoanMonthsSinceOrigination           113937 non-null  int64
 62  LoanNumber                           113937 non-null  int64
 63  LoanOriginalAmount                   113937 non-null  int64
 64  LoanOriginationDate                  113937 non-null  object
 65  LoanOriginationQuarter               113937 non-null  object
 66  MemberKey                            113937 non-null  object
 67  MonthlyLoanPayment                   113937 non-null  float64
 68  LP_CustomerPayments                  113937 non-null  float64
 69  LP_CustomerPrincipalPayments         113937 non-null  float64
 70  LP_InterestandFees                   113937 non-null  float64
 71  LP_ServiceFees                       113937 non-null  float64
 72  LP_CollectionFees                    113937 non-null  float64
 73  LP_GrossPrincipalLoss                113937 non-null  float64
 74  LP_NetPrincipalLoss                  113937 non-null  float64
 75  LP_NonPrincipalRecoverypayments      113937 non-null  float64
 76  PercentFunded                        113937 non-null  float64
 77  Recommendations                      113937 non-null  int64
 78  InvestmentFromFriendsCount           113937 non-null  int64
 79  InvestmentFromFriendsAmount          113937 non-null  float64
 80  Investors                            113937 non-null  int64
dtypes: bool(3), float64(50), int64(11), object(17)
memory usage: 68.1+ MB
```

There many features in this dataset (81 variables). Not all the variables are essential. Thus there is a need to extract important variables only.

Extracting variables of interest that can answer following research questions

```
What factors affect a loan’s outcome status?
What affects the borrower’s APR or interest rate?
Are there differences between loans depending on how large the original
loan amount was
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['ListingKey', 'ListingNumber', 'ListingCreationDate', 'CreditGrade',
       'Term', 'LoanStatus', 'ClosedDate', 'BorrowerAPR', 'BorrowerRate',
       'LenderYield', 'EstimatedEffectiveYield', 'EstimatedLoss',
       'EstimatedReturn', 'ProsperRating (numeric)', 'ProsperRating (Alpha)',
       'ProsperScore', 'ListingCategory (numeric)', 'BorrowerState',
       'Occupation', 'EmploymentStatus', 'EmploymentStatusDuration',
       'IsBorrowerHomeowner', 'CurrentlyInGroup', 'GroupKey',
       'DateCreditPulled', 'CreditScoreRangeLower', 'CreditScoreRangeUpper',
       'FirstRecordedCreditLine', 'CurrentCreditLines', 'OpenCreditLines',
       'TotalCreditLinespast7years', 'OpenRevolvingAccounts',
       'OpenRevolvingMonthlyPayment', 'InquiriesLast6Months', 'TotalInquiries',
       'CurrentDelinquencies', 'AmountDelinquent', 'DelinquenciesLast7Years',
       'PublicRecordsLast10Years', 'PublicRecordsLast12Months',
       'RevolvingCreditBalance', 'BankcardUtilization',
       'AvailableBankcardCredit', 'TotalTrades',
       'TradesNeverDelinquent (percentage)', 'TradesOpenedLast6Months',
       'DebtToIncomeRatio', 'IncomeRange', 'IncomeVerifiable',
       'StatedMonthlyIncome', 'LoanKey', 'TotalProsperLoans',
       'TotalProsperPaymentsBilled', 'OnTimeProsperPayments',
       'ProsperPaymentsLessThanOneMonthLate',
       'ProsperPaymentsOneMonthPlusLate', 'ProsperPrincipalBorrowed',
```

```
       'ProsperPrincipalOutstanding', 'ScorexChangeAtTimeOfListing',
       'LoanCurrentDaysDelinquent', 'LoanFirstDefaultedCycleNumber',
       'LoanMonthsSinceOrigination', 'LoanNumber', 'LoanOriginalAmount',
       'LoanOriginationDate', 'LoanOriginationQuarter', 'MemberKey',
       'MonthlyLoanPayment', 'LP_CustomerPayments',
       'LP_CustomerPrincipalPayments', 'LP_InterestandFees', 'LP_ServiceFees',
       'LP_CollectionFees', 'LP_GrossPrincipalLoss', 'LP_NetPrincipalLoss',
       'LP_NonPrincipalRecoverypayments', 'PercentFunded', 'Recommendations',
       'InvestmentFromFriendsCount', 'InvestmentFromFriendsAmount',
       'Investors'],
      dtype='object')
```

In [ ]:

In [6]:
```
cols = ['Term', 'LoanStatus',  'BorrowerAPR', 'BorrowerRate','LenderYield', 'ListingCate
        'EmploymentStatusDuration','IsBorrowerHomeowner', 'EmploymentStatus', 'StatedMon
        'IncomeVerifiable', 'DebtToIncomeRatio', 'LoanOriginalAmount',
        'DelinquenciesLast7Years','MonthlyLoanPayment']
df_sub = df[cols]
```

In [7]:
```
df_sub.head(3)
```

Out[7]:

	Term	LoanStatus	BorrowerAPR	BorrowerRate	LenderYield	ListingCategory (numeric)	EmploymentStatusDuratio
0	36	Completed	0.16516	0.158	0.138	0	2.
1	36	Current	0.12016	0.092	0.082	2	44.
2	36	Completed	0.28269	0.275	0.240	0	Na

In [8]:
```
df_sub.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 16 columns):
 #   Column                     Non-Null Count   Dtype  
---  ------                     --------------   -----  
 0   Term                       113937 non-null  int64  
 1   LoanStatus                 113937 non-null  object 
 2   BorrowerAPR                113912 non-null  float64
 3   BorrowerRate               113937 non-null  float64
 4   LenderYield                113937 non-null  float64
 5   ListingCategory (numeric)  113937 non-null  int64  
 6   EmploymentStatusDuration   106312 non-null  float64
 7   IsBorrowerHomeowner        113937 non-null  bool   
 8   EmploymentStatus           111682 non-null  object 
 9   StatedMonthlyIncome        113937 non-null  float64
 10  IncomeRange                113937 non-null  object 
 11  IncomeVerifiable           113937 non-null  bool   
 12  DebtToIncomeRatio          105383 non-null  float64
 13  LoanOriginalAmount         113937 non-null  int64  
 14  DelinquenciesLast7Years    112947 non-null  float64
 15  MonthlyLoanPayment         113937 non-null  float64
dtypes: bool(2), float64(8), int64(3), object(3)
memory usage: 12.4+ MB
```

In [9]:
```
((df_sub.isna().sum()/len(df_sub))*100).round(2)
```

Out[9]:
```
Term                          0.00
LoanStatus                    0.00
BorrowerAPR                   0.02
BorrowerRate                  0.00
```

```
LenderYield                  0.00
ListingCategory (numeric)    0.00
EmploymentStatusDuration     6.69
IsBorrowerHomeowner          0.00
EmploymentStatus             1.98
StatedMonthlyIncome          0.00
IncomeRange                  0.00
IncomeVerifiable             0.00
DebtToIncomeRatio            7.51
LoanOriginalAmount           0.00
DelinquenciesLast7Years      0.87
MonthlyLoanPayment           0.00
dtype: float64
```

In [10]:
```
#Making a copy of data
df_c=df_sub.copy()
```

In [11]:
```
df_c.dropna(inplace=True)
```

In [12]:
```
df_c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 97888 entries, 0 to 113936
Data columns (total 16 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Term                       97888 non-null  int64
 1   LoanStatus                 97888 non-null  object
 2   BorrowerAPR                97888 non-null  float64
 3   BorrowerRate               97888 non-null  float64
 4   LenderYield                97888 non-null  float64
 5   ListingCategory (numeric)  97888 non-null  int64
 6   EmploymentStatusDuration   97888 non-null  float64
 7   IsBorrowerHomeowner        97888 non-null  bool
 8   EmploymentStatus           97888 non-null  object
 9   StatedMonthlyIncome        97888 non-null  float64
 10  IncomeRange                97888 non-null  object
 11  IncomeVerifiable           97888 non-null  bool
 12  DebtToIncomeRatio          97888 non-null  float64
 13  LoanOriginalAmount         97888 non-null  int64
 14  DelinquenciesLast7Years    97888 non-null  float64
 15  MonthlyLoanPayment         97888 non-null  float64
dtypes: bool(2), float64(8), int64(3), object(3)
memory usage: 11.4+ MB
```

In [13]:
```
df_c.columns
```

Out[13]:
```
Index(['Term', 'LoanStatus', 'BorrowerAPR', 'BorrowerRate', 'LenderYield',
       'ListingCategory (numeric)', 'EmploymentStatusDuration',
       'IsBorrowerHomeowner', 'EmploymentStatus', 'StatedMonthlyIncome',
       'IncomeRange', 'IncomeVerifiable', 'DebtToIncomeRatio',
       'LoanOriginalAmount', 'DelinquenciesLast7Years', 'MonthlyLoanPayment'],
      dtype='object')
```

In [14]:
```
df.shape
```

Out[14]:
```
(113937, 81)
```

What is the structure of your dataset?

> The dataset (prosperLoan data)contains 113937 rows(observations) and 81 columns (features). The dataset contains various data types such as integers, floats, bool and strings (categories). However, not all the the features are essential, some variables which

> can answer research questions would be selected for analysis. A new dataset with the variables of interest was created with a structure of 106312 observations and 17 features.

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

> The main features of interest from the prosper dataset are
>
> - factors that affect a loan's outcome status,
> - factors that affects the borrower's APR or interest rate,
> - ascertaining if there are differences between loans depending on how large the original loan amount was?

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

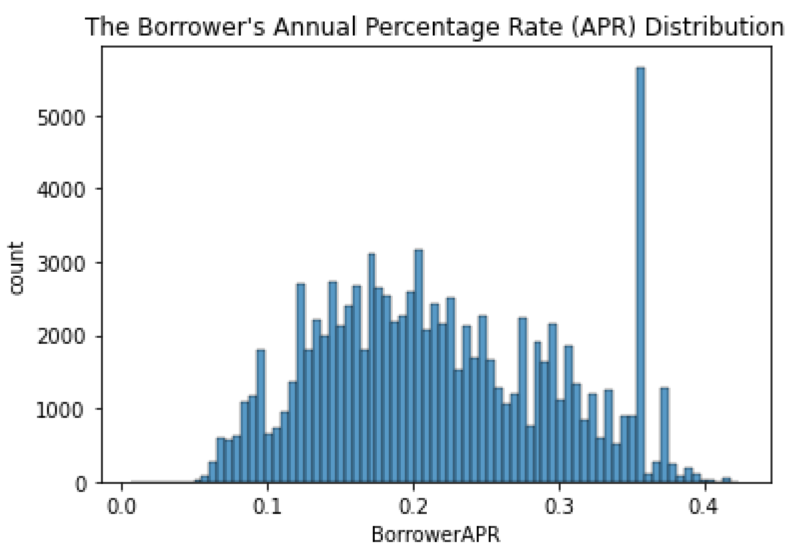
> Term','LoanStatus', 'BorrowerAPR', 'BorrowerRate', 'LenderYield','ListingCategory (numeric)', 'EmploymentStatusDuration','IsBorrowerHomeowner', 'EmploymentStatus', 'StatedMonthlyIncome','IncomeRange', 'IncomeVerifiable', 'DebtToIncomeRatio', 'LoanOriginalAmount', 'DelinquenciesLast7Years', 'MonthlyLoanPayment'

Univariate Exploration

Checking BorrowerAPR Distribution

Since BorrowerAPR is continous variable representing the it using histogram would br more informative.

In [15]:
```
sb.histplot(df_c['BorrowerAPR'])
plt.ylabel('count')
plt.title("The Borrower's Annual Percentage Rate (APR) Distribution");
```

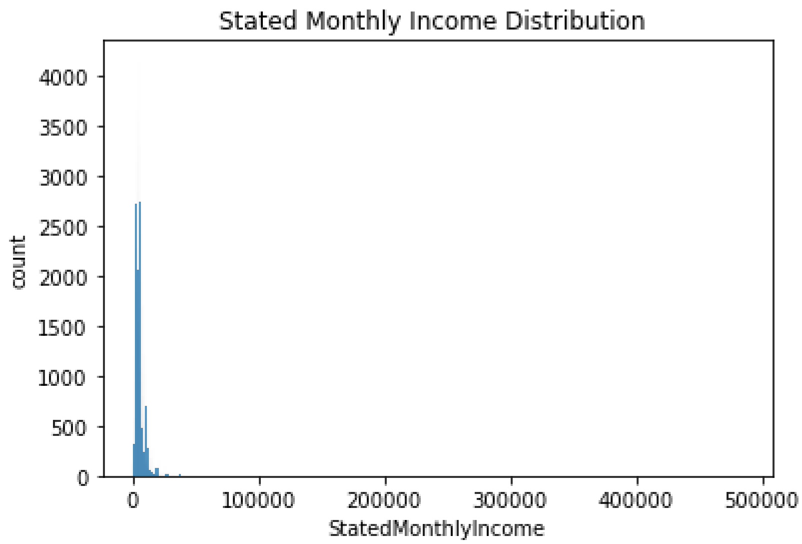


We can see that the above BorrowerAPR distribution a multimodal since it has atleast three peaks, the first smaller peak is observed between 0.0 & 0.1, then large peak at 0.2 and very high peak between 0.35 & 0.36, but above 0.4 BorrowerAPR the peak decreases. Thus BorrowerAPR is high between 0.35 & 0.36 and at around 0.2.

Checking StatedMonthlyIncome Distribution

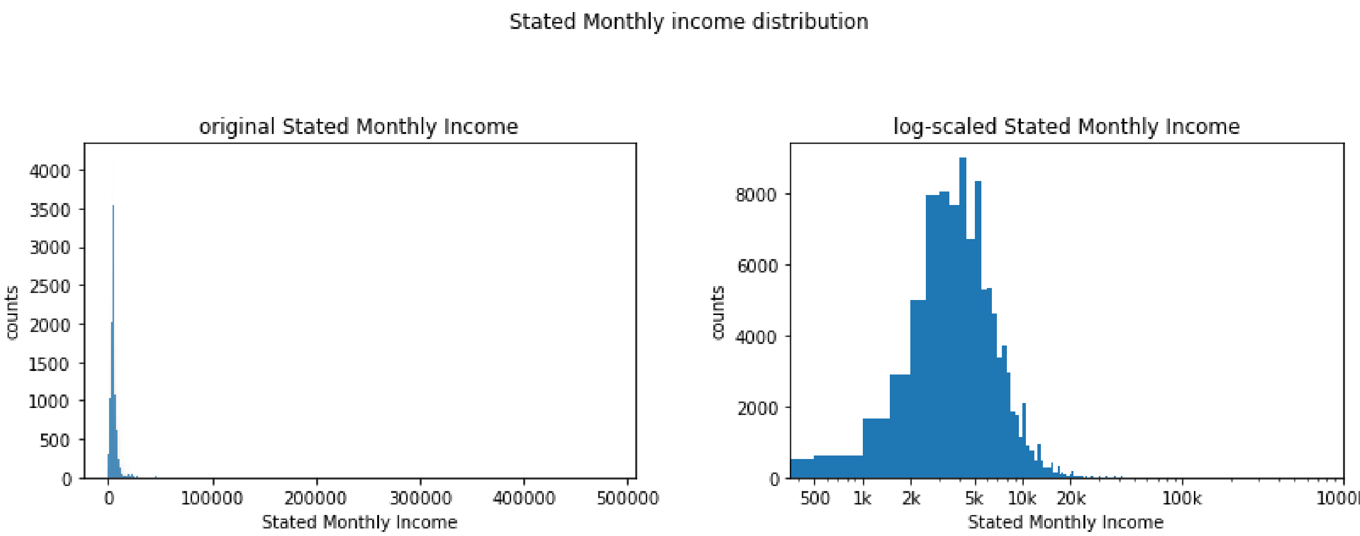
Since StatedMonthlyIncome is continous variable representing the it using histogram would br more informative.

In [16]:
```
sb.histplot(df_c['StatedMonthlyIncome'])
plt.ylabel('count')
plt.title('Stated Monthly Income Distribution ');
```



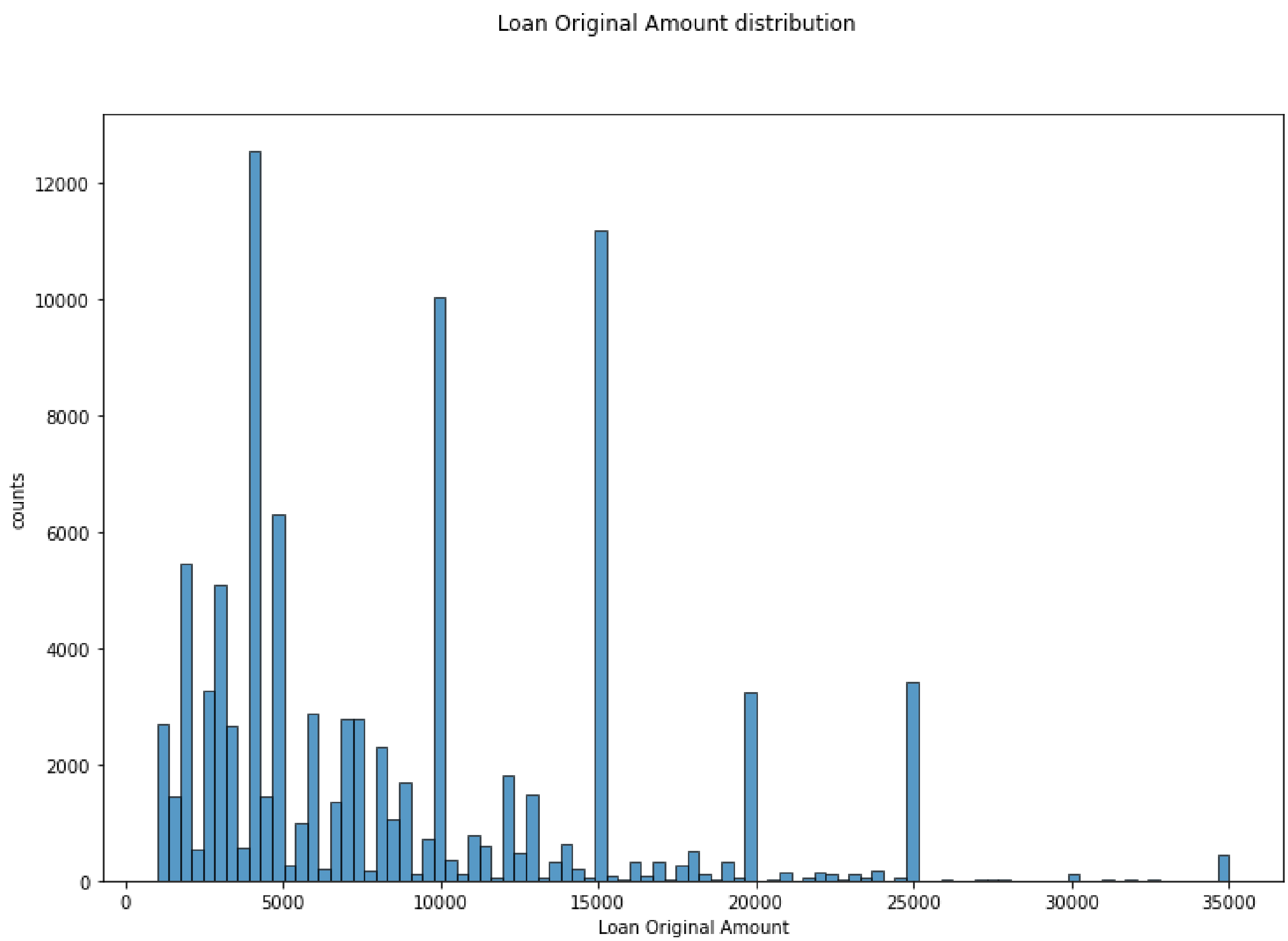
- We can see that there is high level of skewness and the far outliers are also observed when Stated Monthly Income is above 100,000 so, thus it would be significant to transform or perform log scaling to this feature and check if there is improvement or outliers

In [17]:
```
fig= plt.figure(figsize=[12,8])
a1=fig.add_subplot(2,2,1)
sb.histplot(df_c['StatedMonthlyIncome'])
plt.xlabel("Stated Monthly Income")
plt.ylabel('counts')
a1.set_title('original Stated Monthly Income')

# there's a long tail in the distribution, so let's put it on a log scale instead
a2=fig.add_subplot(2,2,2)
bins = np.arange(0,df_c['StatedMonthlyIncome'].max()+500,500)

plt.hist(data=df_c,x='StatedMonthlyIncome',bins=bins)
plt.xscale('log')
plt.xticks([500,1e3,2e3,5e3,1e4,2e4,1e5,1e6],['500','1k','2k','5k','10k','20k','100k','1
plt.xlabel("Stated Monthly Income")
plt.ylabel('counts')
a2.set_title('log-scaled Stated Monthly Income')
fig.suptitle(" Stated Monthly income distribution")
fig.tight_layout(pad=3.0);
```



The Stated Monthly income in the original data has a right long-tailed distribution 'right skewed'.There is high level of skewness and the far outliers are also observed when Stated Monthly Income is above 100,000 so, thus it would be significant to transform or perform log scaling to this feature and check if there is improvement or outliers. After performing log-scaling on Stated Monthly income variable some outliers were observed above than 20K.

In [18]:
```
#Create a histogram to show the distribution of loan original amount
fig= plt.figure(figsize=[12,8])
# a1=fig.add_subplot(2,2,1)
sb.histplot(df_c['LoanOriginalAmount'])
plt.xlabel("Loan Original Amount")
plt.ylabel('counts')
fig.suptitle(" Loan Original Amount distribution");
```

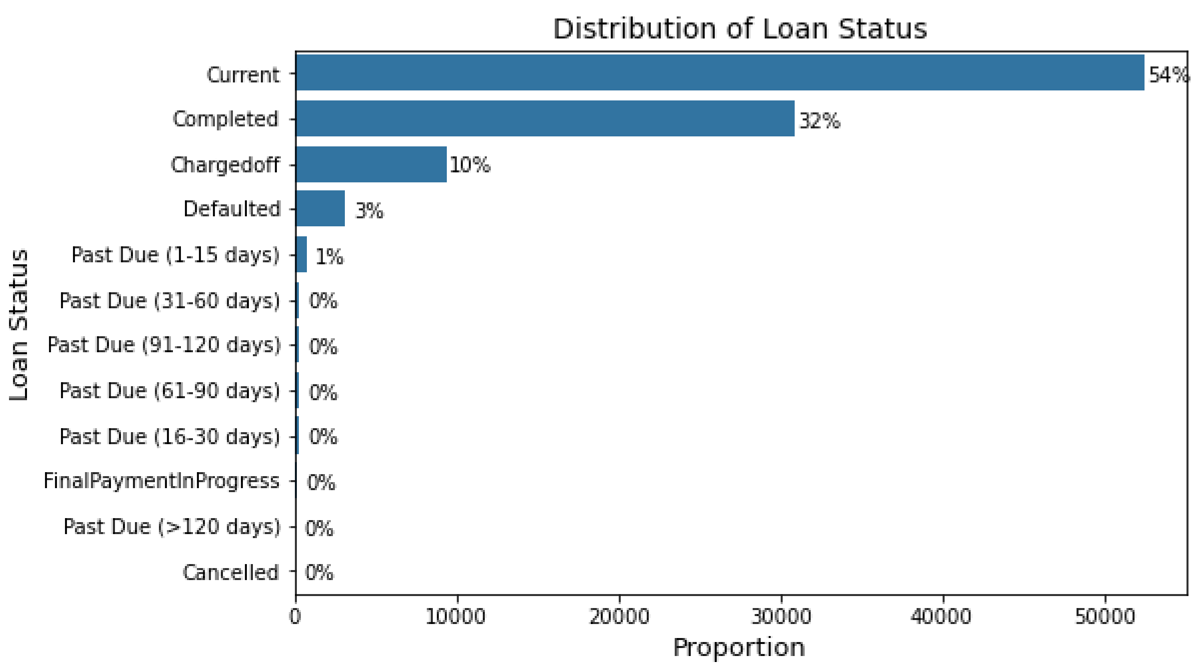


We can see that the above original loan amount distribution is a multimodal with various peak variations. Moreover, we can see outliers above 25000,this depicts that few individuals were granted huge amount of loans as compared to those who were given lower especeially those who granted loan amount in range of 15000 and below.

Checking LoanStatus Distribution

Since LoanStatus is categorical variable representing the it using harizontal bar graph would be more informative.

In [19]:
```
# Loan Status Distribution
status_order = df_c['LoanStatus'].value_counts().index
base_color = sb.color_palette()[0]
plt.figure(figsize=[8, 5])
sb.countplot(data=df_c,y='LoanStatus',color=base_color,order=status_order);
plt.title('Distribution of Loan Status', fontsize=14)
plt.ylabel('Loan Status', fontsize=13)
plt.xlabel('Proportion', fontsize=13)

# add annotations
n_points = df_c.shape[0]
cat_counts = df_c['LoanStatus'].value_counts()
locs, labels = plt.yticks() # get the current tick locations and labels

# loop through each pair of locations and labels
for loc, label in zip(locs, labels):

    # get the text property for the label to get the correct count
    count = cat_counts[label.get_text()]
    pct_string = '{:0.0f}%'.format(100*count/n_points)

    # print the annotation just below the top of the bar
    plt.text(count+1500, loc+0.2,  pct_string, ha = 'center', color = 'black');
```

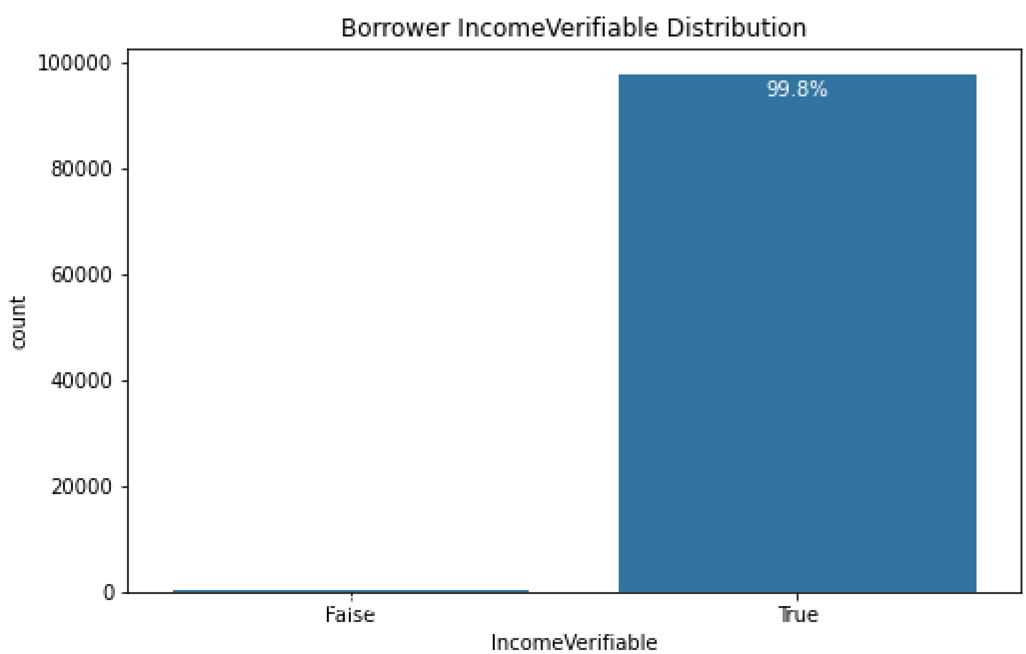


We can see that current status of the loan has largest proportion of loan status followed by those who have completed there are few cases loan defaulters.

Checking IncomeVerifiable Distribution

Since IncomeVerifiable is categorical variable representing the it using bar graph would be more informative.

In [20]:
```
#Is Income-Verifiable Status Distrobution
plt.figure(figsize=[8, 5])

def str2bool(v):
    return str(v).lower() in ("yes", "true", "True", "1")

sb.countplot(data = df_c, x = 'IncomeVerifiable', color = base_color)

# add annotations
n_points = df_c.shape[0]
cat_counts = df_c['IncomeVerifiable'].value_counts()
locs, labels = plt.xticks() # get the current tick locations and labels

# loop through each pair of locations and labels
for loc, label in zip(locs, labels):

    # get the text property for the label to get the correct count
    count = cat_counts[str2bool(label.get_text())]
    pct_string = '{:0.1f}%'.format(100*count/n_points)

    # print the annotation just below the top of the bar
    plt.text(loc, count-4000, pct_string, ha = 'center', color = 'w')

#sb.countplot(data=df_loans_clean,x='IsBorrowerHomeowner',color=base_color);
plt.title('Borrower IncomeVerifiable Distribution');
```



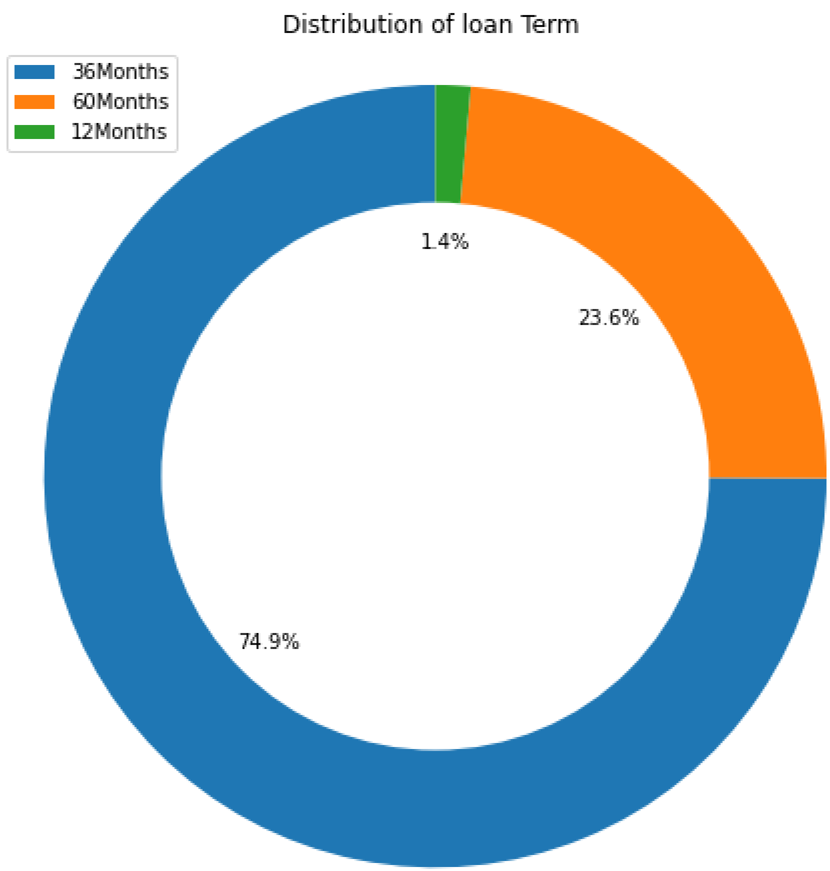
The proportion of the individuals taking loans have verifiable income, which might be a requirement for taking loans

Checking Term Distribution

Since Term variable is categorical variable representing it using pie chart or bar graph would be more informative.

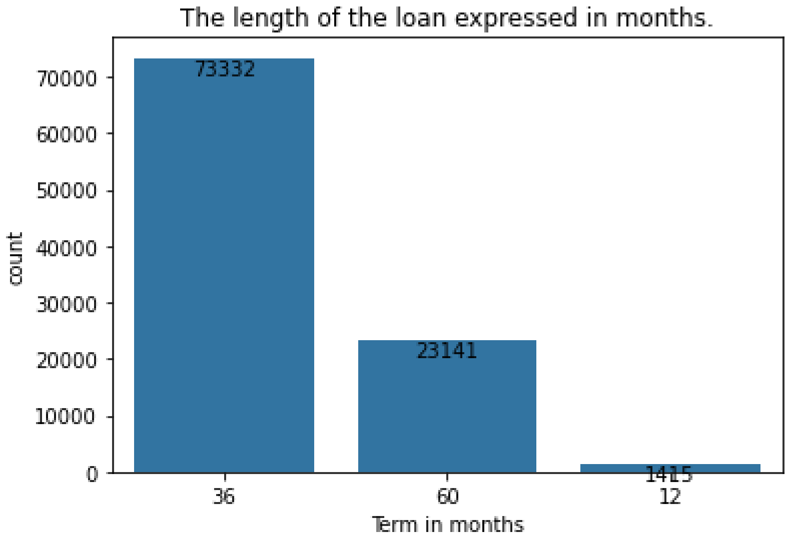
In [21]:
```
plt.figure(figsize=[8,8])
c = df_c.Term.value_counts()
labels = '36Months','60Months','12Months'
```

```
plt.pie(c, autopct='%1.1f%%', startangle=90)
# draw circle
centre_circle = plt.Circle((0, 0), 0.70, fc='white')
fig = plt.gcf()

# Adding Circle in Pie chart
fig.gca().add_artist(centre_circle)

# Adding Title of chart
plt.title('Favourite Fruit Survey')


plt.title('Distribution of loan Term')
plt.axis('square')
plt.legend(labels);
```



In [22]:

```
#Using Barplot for this distribution
term_order = df_c.Term.value_counts().index
sb.countplot(x='Term', data = df_c, color = base_color, order = term_order)
plt.title('The length of the loan expressed in months.');
plt.xlabel('Term in months')

# Adding counts of each term in our data on top of of each bar.
for i in range (df_c.Term.value_counts().shape[0]):
    count = df_c.Term.value_counts().values[i]
    plt.text(i, count, count, ha = 'center', va='top')
```



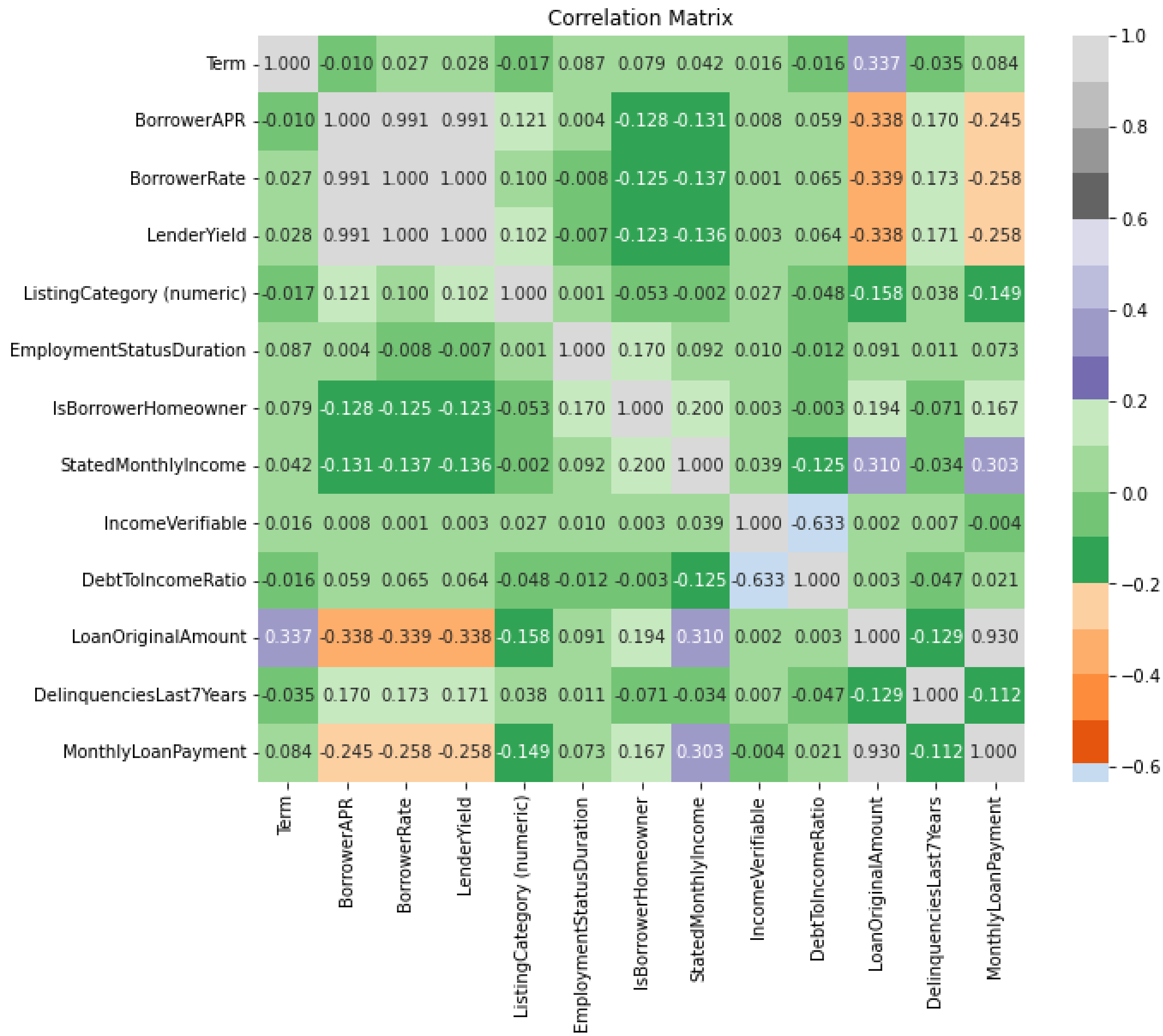
We can see that most of the individuals who took loan prefer 36 months plan on but few borrowers opted for 12 months term

Bivariate Exploration

> Let us check if the association/ correlation of other factors on Borrower's APR and BorrowerRate

Checking relationship between various variables (including loan status , Borrower's APR loan original amount etc

In [23]:
```
# Correlation matrix for all numeric variables
plt.figure(figsize = [10, 8])
sb.heatmap(df_c.corr(), annot = True, fmt = '.3f',cmap = 'tab20c', center = 0)
plt.title('Correlation Matrix')
plt.margins(x = 0.5, y= 0.3)
plt.show()
```



We can see that there is high multicollinearity between LenderYield, BorrowerAPR and BorrowerRate and so we can drop LenderYield.

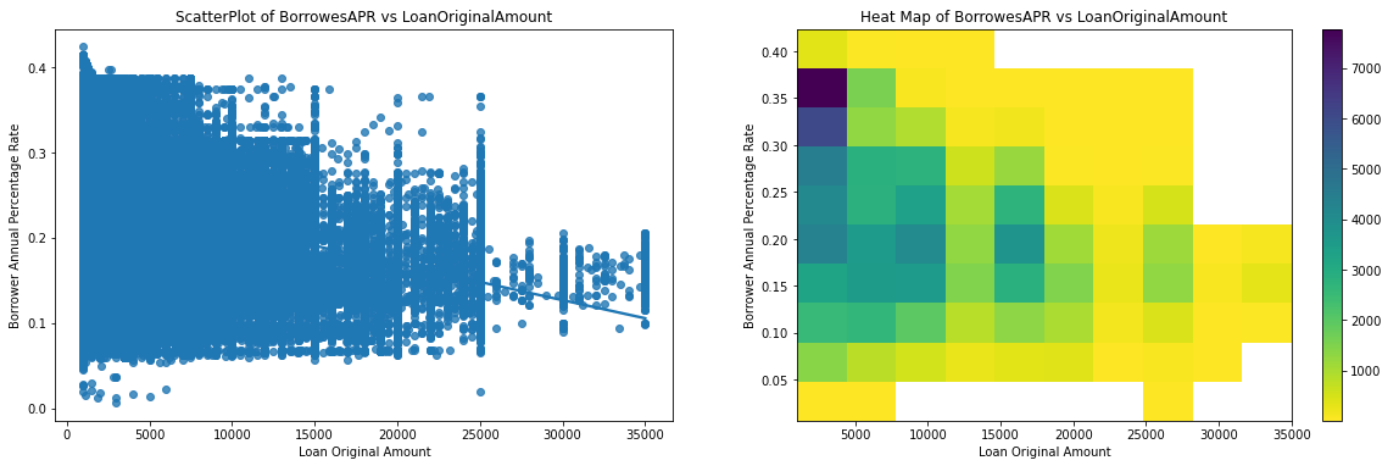
From the correlation matrix heatmap, Term has a weak positive correlation (0.027) with Borrower Rate and a waek negative correlation (-0.01) with BorrowerAPR.

We can see that factors that affect borrower's APR or interest rate are mainly LoanOriginalAmount followed by MonthlyLoanPayment, DelinquenciesLast7Years, StatedMonthlyIncome,IsBorrowerHomeowner,ListingCategory

Checking relationship between BorrowerAPR and LoanOriginalAmount

In [24]:
```
# Create a subplot for LoanOriginal Amount and Term
plt.figure(figsize=[20,13])

plt.subplot(2,2,1)
sb.regplot(data=df_c, y='BorrowerAPR', x='LoanOriginalAmount') #creating a scatter plot
plt.ylabel('Borrower Annual Percentage Rate')
plt.xlabel('Loan Original Amount')
plt.title('ScatterPlot of BorrowesAPR vs LoanOriginalAmount');

plt.subplot(2,2,2)
plt.hist2d(data=df_c,  y='BorrowerAPR', x='LoanOriginalAmount', cmap='viridis_r', cmin=0
```

```
plt.ylabel('Borrower Annual Percentage Rate')
plt.xlabel('Loan Original Amount')
plt.title('Heat Map of BorrowesAPR vs LoanOriginalAmount');
plt.colorbar();
```



From the correlation matrix and the relationship shown on both the heatmap and scatter plot, a negative correlation clearly exists between the BorrowerAPR and the LoanOriginalAmount and also the Borrower Interest Rate vs Loan Original Amount. Loan original amounts greater than `$20,000` are much more prone to have lower Borrower APR and Borrower Interest Rate compared to lesser amount of `$10,000` and below which are more likely to have higher Borrower APR and Borrower Interest Rate. Thus, there is clearly a negative correlation albeit a weak one.

Checking relationship between BorrowerAPR and Term

In [50]:

```
def SplitString(string):
    '''
    Splitting string()- adding a space before an upper case'''
    #loop through each character if a char is lower case leave it as it is else put a sp
    return ''.join([x if x.islower() else f" {x}" for x in string])
```

In [53]:

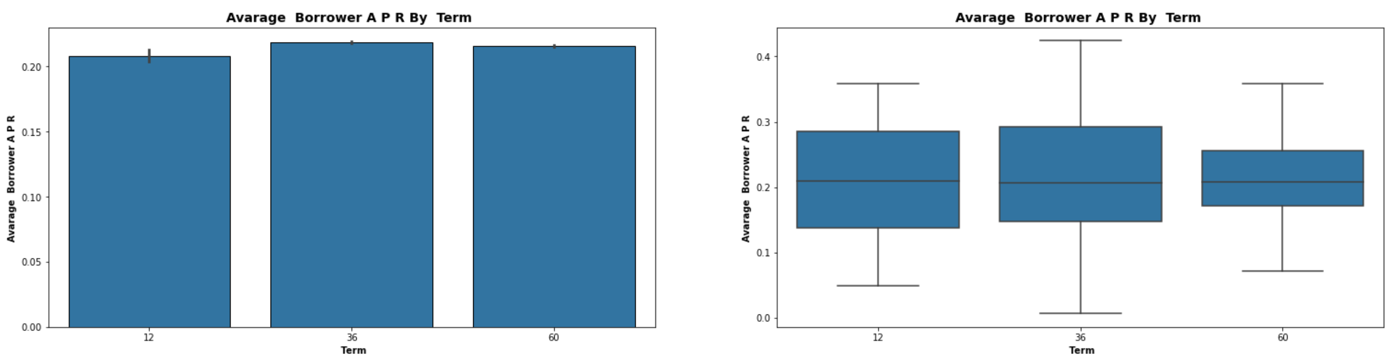
```
def MyBarPlot(df, xVar, yVar,hue=None, color=0, palette=None,order=None,hue_order=None,
    '''Splitting string()- adding a space before an upper case'''
    if not ax:
        #set plot dimensions
        plt.figure(figsize=figsize)
        ax=plt.gca()

    sb.barplot(data=df,x=xVar,y=yVar,hue=hue,color=sb.color_palette()[color],palette=pal
               order=order,hue_order=hue_order)
    #clean up variable names
    xVar=SplitString(xVar).replace("_"," ")
    yVar=SplitString(yVar).replace("_"," ")
    if hue:
        hue=splitString(hue)
    #add title and format it
    ax.set_title(f'''Avarage {yVar} by {xVar} {'and' if hue else ''}'''.title(),fontsize
    #add x label and format it
    ax.set_xlabel(xVar.title(),fontsize=10, weight='bold')
        #add y label and format it
    ax.set_ylabel(f'Avarage {yVar}'.title(),fontsize=10, weight='bold')
```

In [65]:

```
def MyBoxPlot(df, xVar, yVar,hue=None, color=0, palette=None,order=None,hue_order=None,
    '''Splitting string()- adding a space before an upper case'''
    if not ax:
        #set plot dimensions
        plt.figure(figsize=figsize)
```

```
        ax=plt.gca()

    sb.boxplot(data=df,x=xVar,y=yVar,hue=hue,color=sb.color_palette()[color],palette=pal
               order=order,hue_order=hue_order)
    #clean up variable names
    xVar=SplitString(xVar).replace("_"," ")
    yVar=SplitString(yVar).replace("_"," ")
    if hue:
        hue=splitString(hue)
    #add title and format it
    ax.set_title(f'''Avarage {yVar} by {xVar} {'and' if hue else ''}'''.title(),fontsize
    #add x label and format it
    ax.set_xlabel(xVar.title(),fontsize=10, weight='bold')
        #add y label and format it
    ax.set_ylabel(f'Avarage {yVar}'.title(),fontsize=10, weight='bold')
```

In [67]:
```
plt.figure(figsize=[26,6])

ax=plt.subplot(1,2,1)
MyBarPlot(df_c, 'Term','BorrowerAPR', ax=ax)

ax1=plt.subplot(1,2,2)
MyBoxPlot(df_c, 'Term','BorrowerAPR', ax=ax1)
```

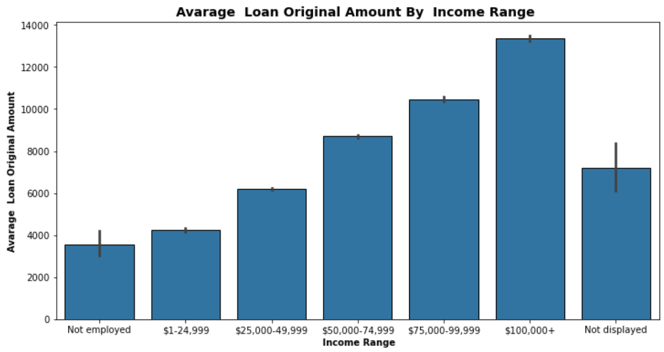


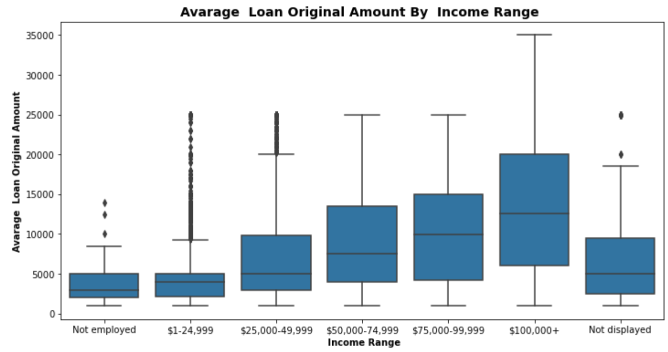
The barplot and box plot shows Term have effect on Borrower Interest. A closer assessment of Term on Borrower APR, showed that average Borrower APR rates for borrowers who takes 12 month Term is lower than borrowers who take 36 and 60 months plans. The Borrower Rate, a 36 and 60 month Term would have a higher BorrowerRate than a loan of a 12 month Term.

Checking relationship between LoanOriginalAmount and IncomeRange

In [ ]:
```
inrange=['Not employed', '$1-24,999','$25,000-49,999', '$50,000-74,999', '$75,000-99,999
        'Not displayed']
```

In [69]:
```
plt.figure(figsize=[26,6])

ax=plt.subplot(1,2,1)
MyBarPlot(df_c, 'IncomeRange','LoanOriginalAmount', order=inrange,ax=ax)


ax1=plt.subplot(1,2,2)
MyBoxPlot(df_c, 'IncomeRange','LoanOriginalAmount', order=inrange,ax=ax1)
```

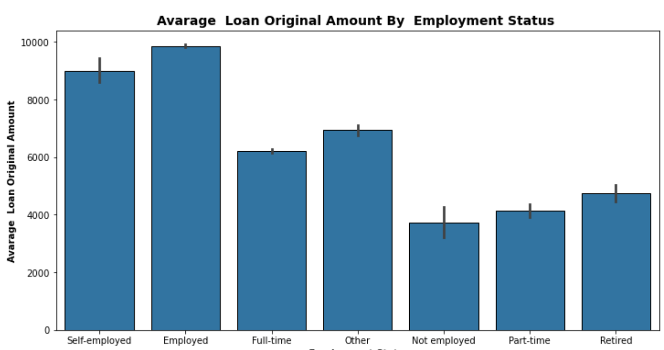


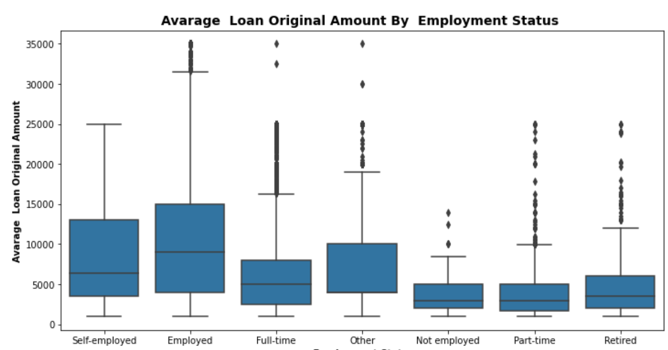


We can see that high income earners such as those who earn above `$100,000` got access to larger sizes of loans followed by those who earn between `$75,000-99,999`. On other hand low income earners such as those earning between `$1-$25,000` had access to the smallest size of loans. It seems the size of income is directly proportional to amount of loan that one can access the the higher the income the higher the loan that one can access, the lower the income the lower the loan that one can access.

Checking relationship between LoanOriginalAmount and EmploymentStatus

In [68]:
```
plt.figure(figsize=[26,6])

ax=plt.subplot(1,2,1)
MyBarPlot(df_c, 'EmploymentStatus','LoanOriginalAmount', ax=ax)


ax1=plt.subplot(1,2,2)
MyBoxPlot(df_c, 'EmploymentStatus','LoanOriginalAmount', ax=ax1)
```

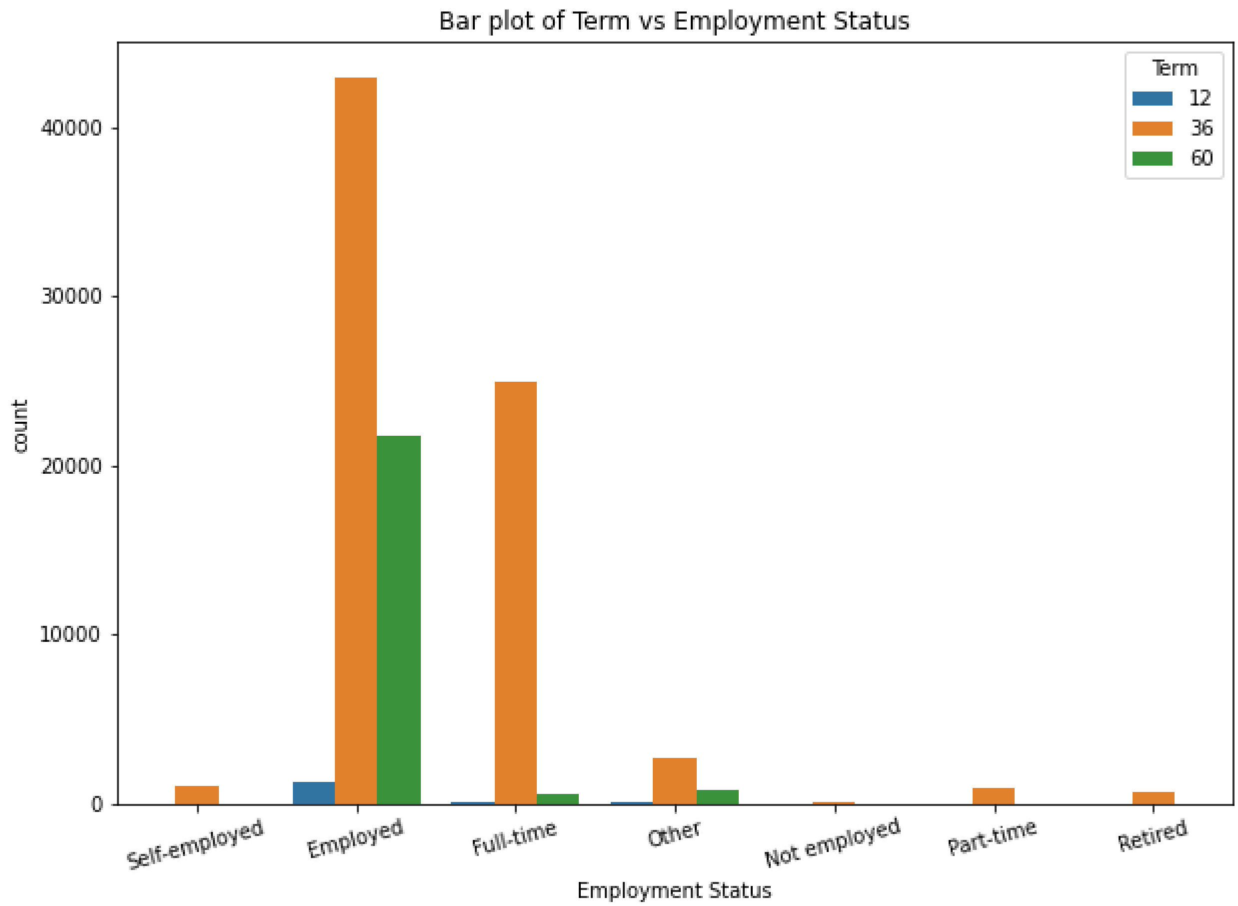




There is is quite relatationship between employement status and loan amount. Those who are employed have access to higher sizes of loans, followed by those who are self-employed. Also full time borrowers could access higher loan than part-time borrowers. Persons who are not employed could access the smallest sizes of loans.

Checking relationship between EmploymentStatus and Term

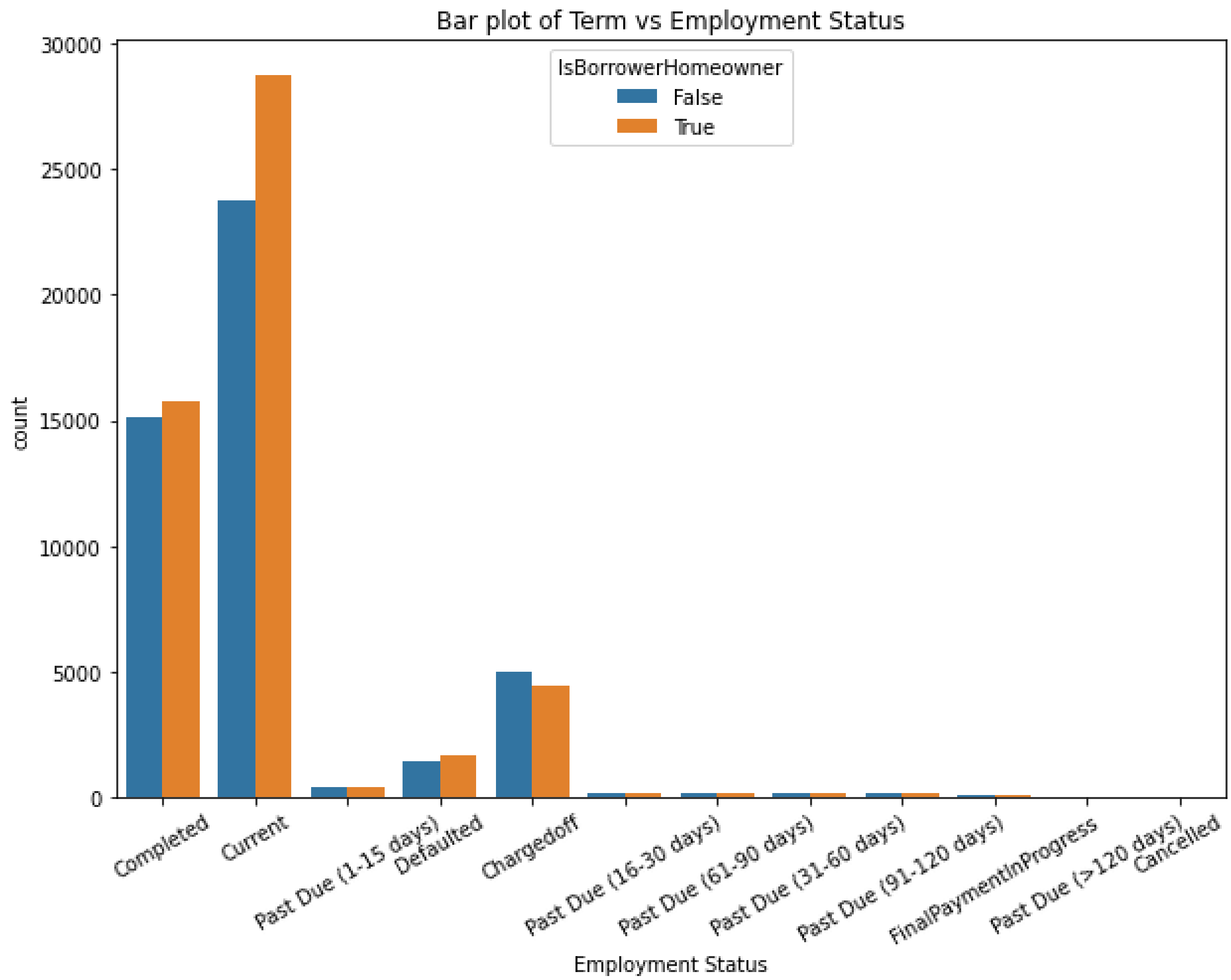
In [30]:
```
plt.figure(figsize=[10,7])
sb.countplot(data=df_c,  hue='Term', x='EmploymentStatus') #creating a boxplot for loan
plt.xlabel('Employment Status')
plt.xticks(rotation=15)
plt.title('Bar plot of Term vs Employment Status');
```



Employement status has no significant effect on term of loan since most of the borrowers prefer 36 months plan irrespective of employment status. Most of the employed indivuals prefer 36 months term and some 60 months plan. Most of borrowers who take 36 months plan are those who are employed followed by full time and others.

Checking relationship between IsBorrowerHomeowner and LoanStatus

In [31]:
```
plt.figure(figsize=[10,7])
sb.countplot(data=df_c,  hue='IsBorrowerHomeowner', x='LoanStatus') #creating a boxplot
plt.xlabel('Employment Status')
plt.xticks(rotation=30)
plt.title('Bar plot of Term vs Employment Status');
```

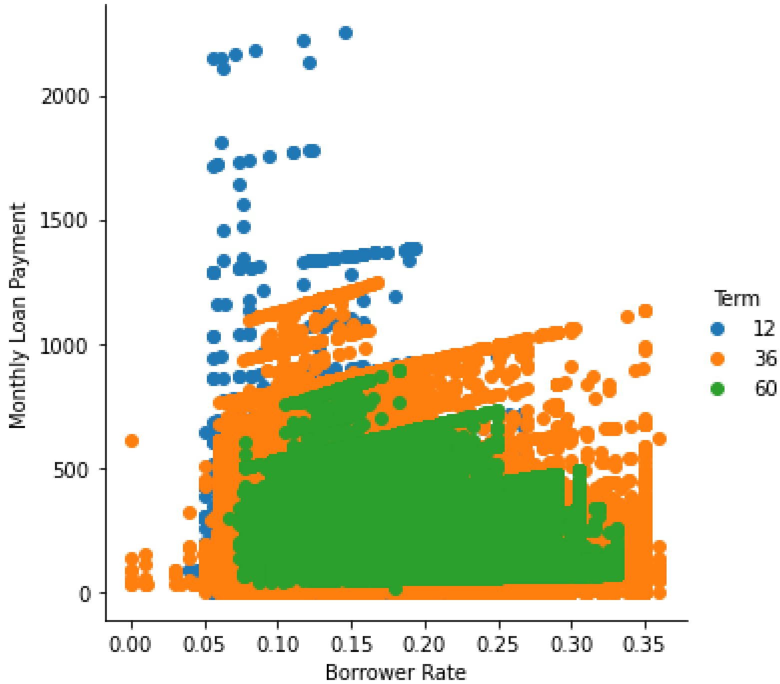


We can see that most of the borrowers who are homeoweners have either completed the loans or current, however most of those who have defaulted are not homeowners (do not own homes)

Multivariate Exploration

Checking relationship between MonthlyLoanPayment, BorrowerRate and Term

In [32]:
```
g = sb.FacetGrid(data = df_c, hue = 'Term', height = 5)
g.map(plt.scatter, 'BorrowerRate','MonthlyLoanPayment')
plt.xlabel('Borrower Rate')
plt.ylabel('Monthly Loan Payment');
g.add_legend();
```



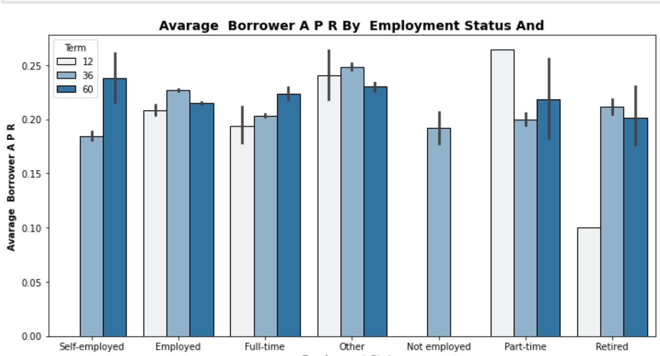
The borrower rate seems to be negatively correlated with monthly loan payment. Borrowers who chose 12 months plan are making higher monthly loan payments and their rate of borrowing (The Borrower's interest rate for the loan) is also low (between 0.05 and 0.18). On other hand borrowes who take 36 months and 60 months term make lower monthly loan payments and their Borrower's interest rate for the loan is high.

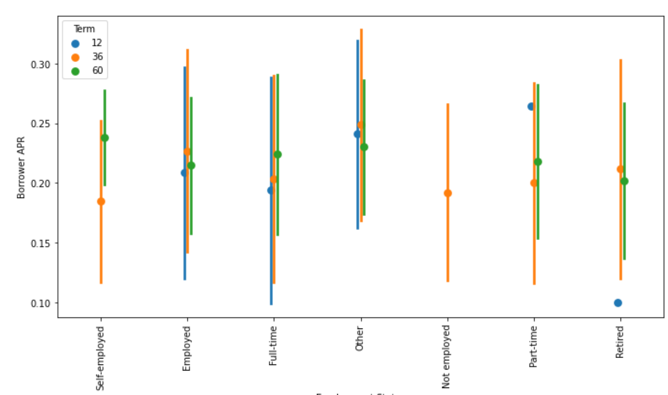
Checking relationship between EmploymentStatus, BorrowerAPR Payment and Term

In [ ]:

In [70]:

```
plt.figure(figsize=[26,6])

ax=plt.subplot(1,2,1)
MyBarPlot(df_c, 'EmploymentStatus','BorrowerAPR',hue='Term', ax=ax)
plt.subplot(1,2,2)
sb.pointplot(data=df_c, x='EmploymentStatus', y='BorrowerAPR',hue='Term',ci='sd',
             linestyles="", dodge=True)
plt.xticks(rotation=90);
plt.xlabel('Employment Status')
plt.ylabel('Borrower APR');
```





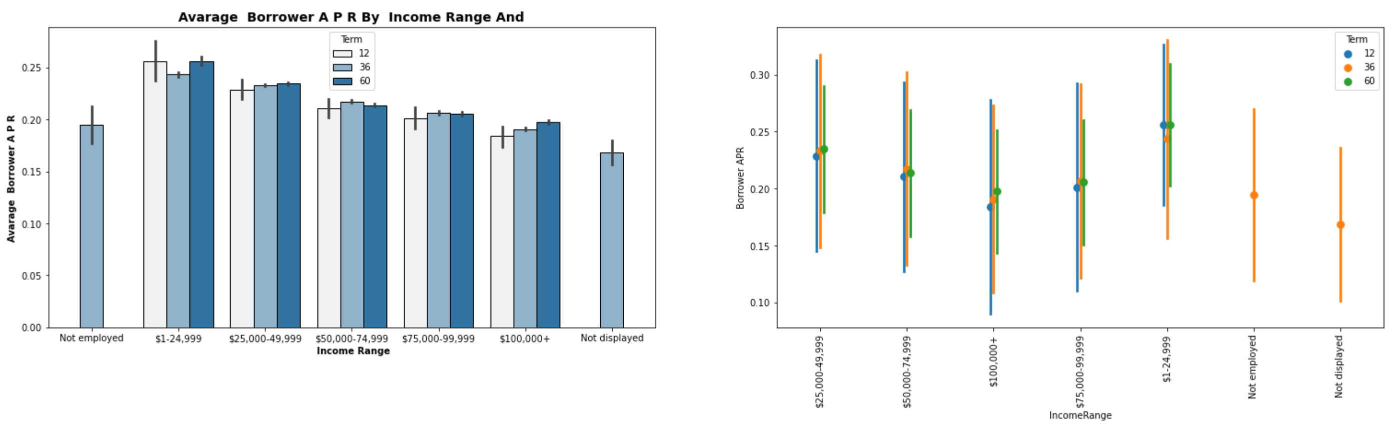
Self-employed borrowers who took 60 months plan have highest average Borrower's Annual Percentage Rate (APR) compared to average borrower rate of employed, part-time, retired etc of the same plan.

Part-time borrowers have the highest 12 months term average borrower rate ompared to average Borrower's Annual Percentage Rate (APR) of employed, part-time, retired etc of the same plan.

Also the mean Borrower rate Self-employed borrowers who take 36 months term is the lowest compared average Borrower's Annual Percentage Rate (APR) of employed, part-time, retired etc of the same plan.

Checking relationship between IncomeRange, BorrowerAPR Payment and Term

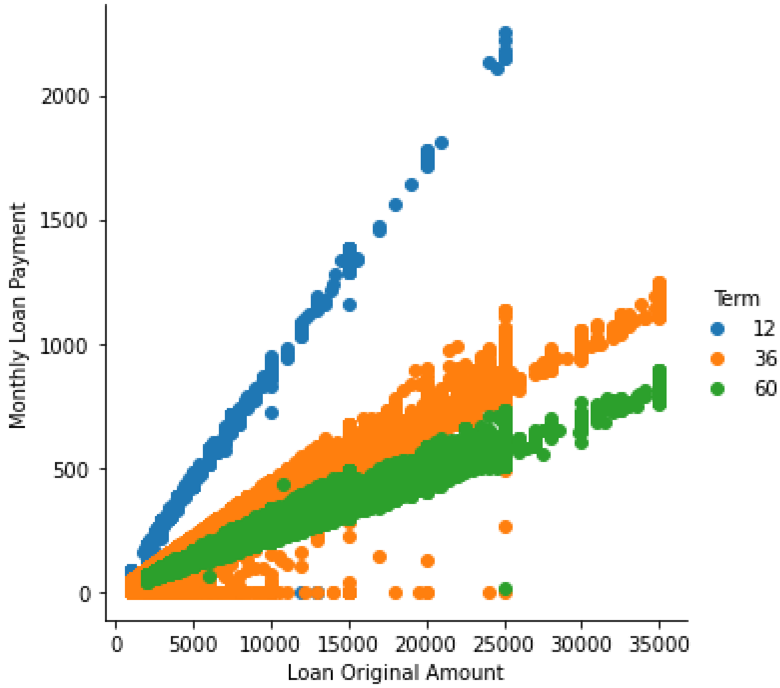
In [71]:
```
plt.figure(figsize=[26,6])

ax=plt.subplot(1,2,1)
MyBarPlot(df_c, 'IncomeRange','BorrowerAPR',hue='Term',order=inrange, ax=ax)
plt.subplot(1,2,2)
sb.pointplot(data=df_c, x='IncomeRange', y='BorrowerAPR',hue='Term',ci='sd',
             linestyles="", dodge=True)
plt.xticks(rotation=90);
plt.ylabel('Borrower APR');
```



The borrowers who earn income range of `$1–24999` have the highest average Borrower's Annual Percentage Rate (APR) for all terms (12, 36 and 60 months plans). On other hand borrowers who earn income of atleast `$100,000` have the lowest average Borrower's Annual Percentage Rate (APR) especially those taking 12, and 60 months term.

Checking relationship between Term, Monthly Loan Payment and LoanOriginalAmount

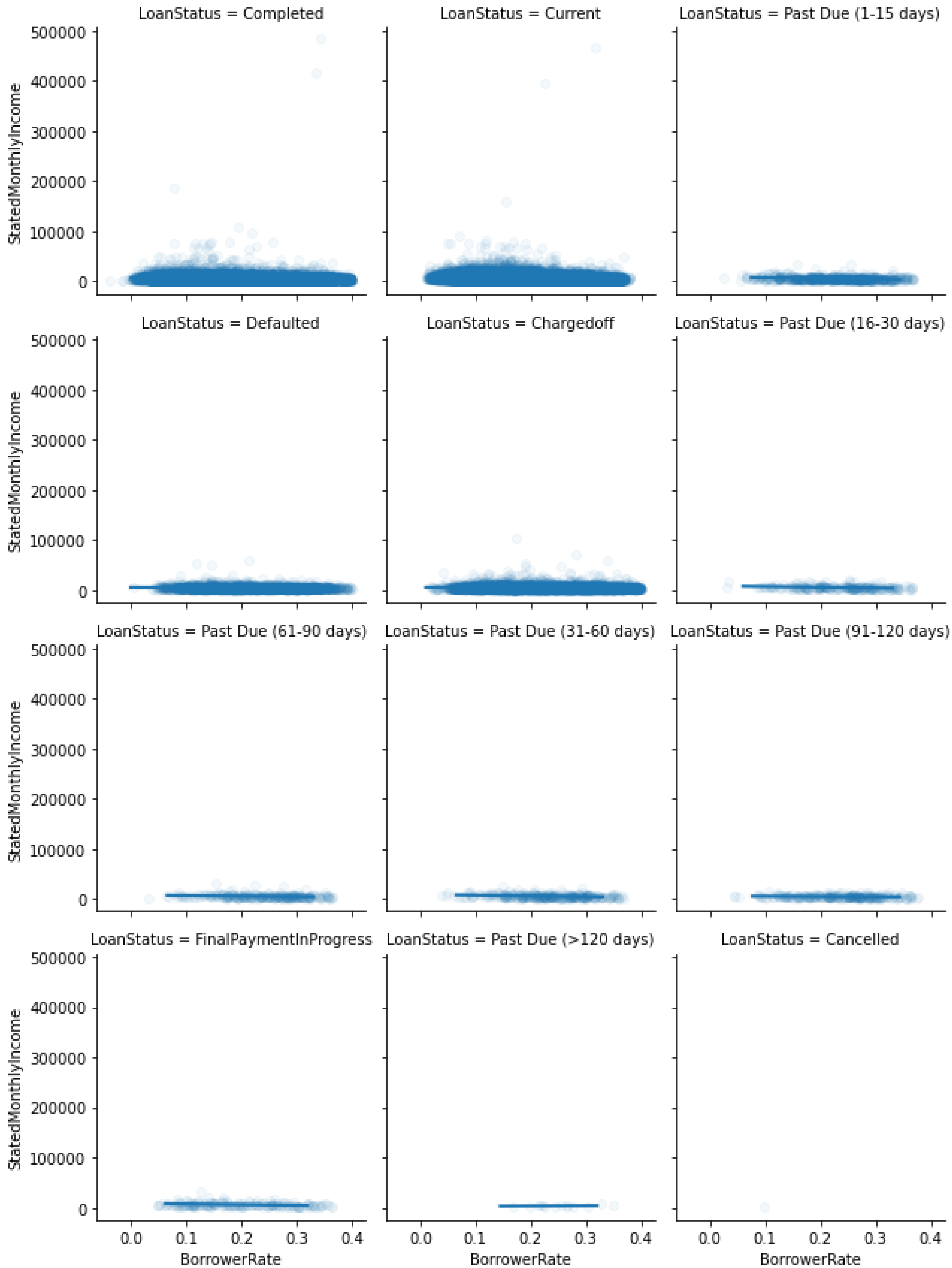
In [36]:
```
g = sb.FacetGrid(data = df_c, hue = 'Term', height = 5)
g.map(plt.scatter, 'LoanOriginalAmount','MonthlyLoanPayment')
plt.xlabel('Loan Original Amount')
plt.ylabel('Monthly Loan Payment');
g.add_legend();
```



There is positive correlation between Monthly Loan Payment and Loan Original Amount. Borrowers who took 12 months term loan pay more loan every month than those who took 36 and 60 months plan.

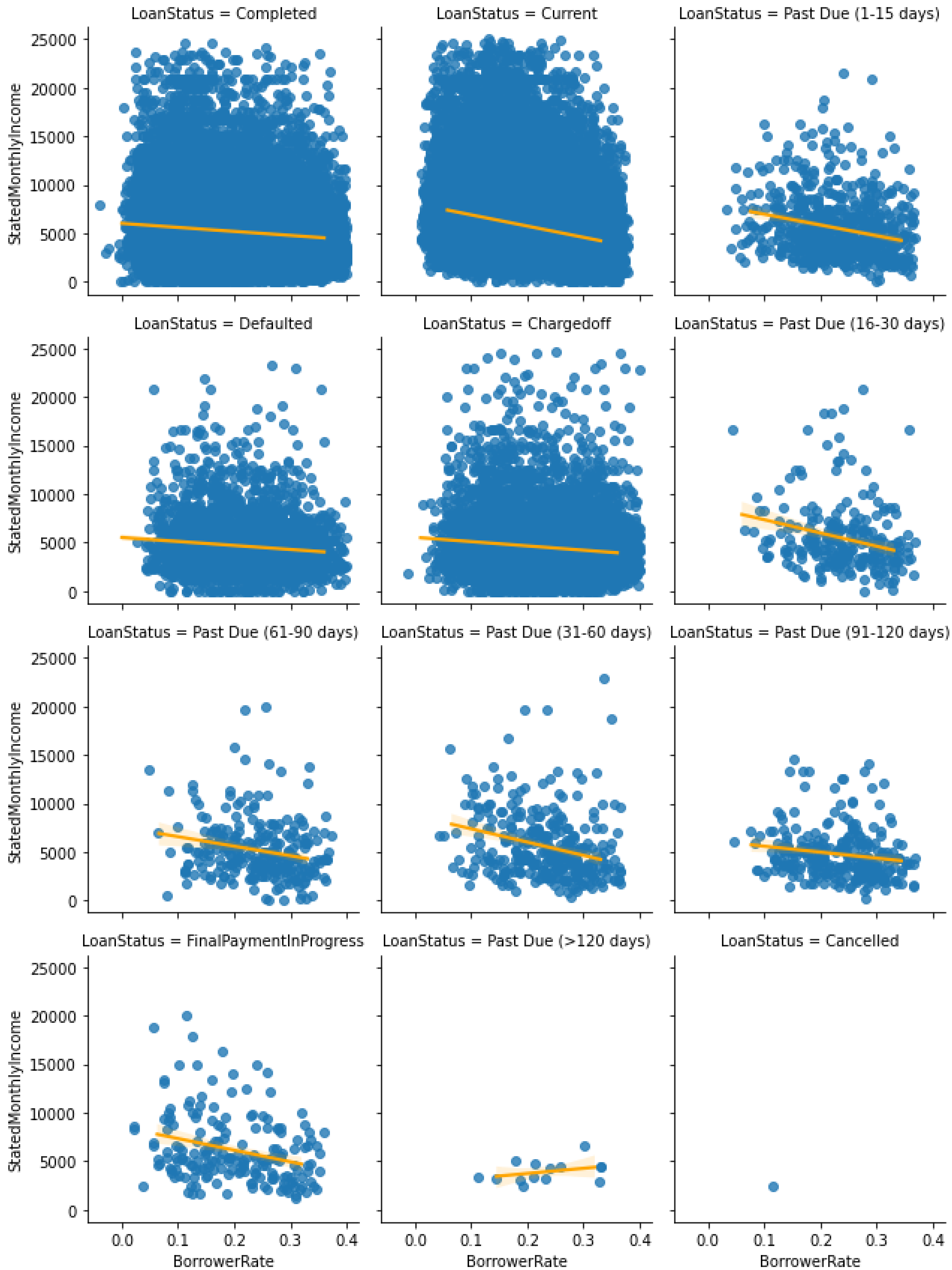
Checking relationship between LoanStatus, BorrowerRate and LoanOriginalAmount

In [74]:
```
g = sb.FacetGrid(data = df_c, col = 'LoanStatus', col_wrap = 3)
g.map(sb.regplot,'BorrowerRate', 'StatedMonthlyIncome', x_jitter = 0.05, scatter_kws = {
```



In [ ]:

In [75]:
```
g = sb.FacetGrid(data = df_c.query('StatedMonthlyIncome<StatedMonthlyIncome.quantile(0.9
                                    col = 'LoanStatus', col_wrap = 3)
g.map(sb.regplot,'BorrowerRate', 'StatedMonthlyIncome', x_jitter = 0.05, line_kws = {'co
```



In [76]:
```
df_c.LoanStatus.unique()
```

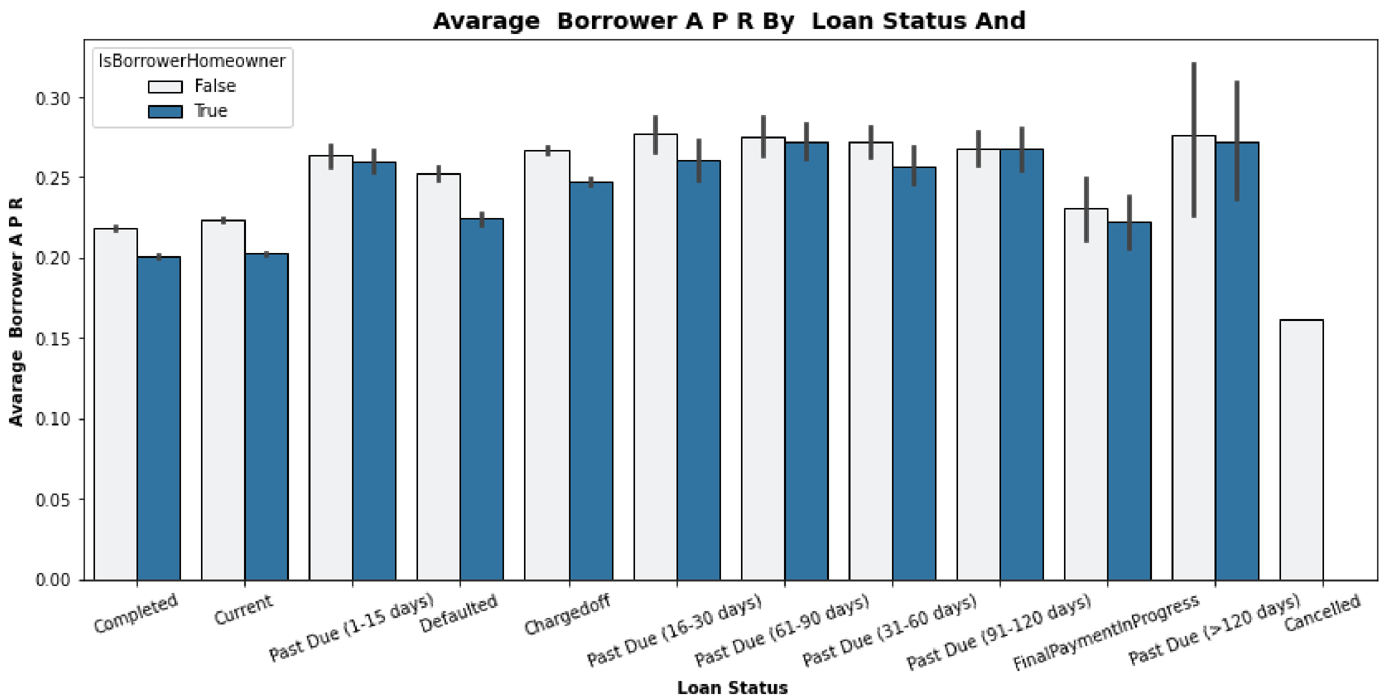
Out[76]:
```
array(['Completed', 'Current', 'Past Due (1-15 days)', 'Defaulted',
       'Chargedoff', 'Past Due (16-30 days)', 'Past Due (61-90 days)',
       'Past Due (31-60 days)', 'Past Due (91-120 days)',
       'FinalPaymentInProgress', 'Past Due (>120 days)', 'Cancelled'],
      dtype=object)
```

The first plot the it showed that Loan Status has no effect on relationship between Stated Monthly Income and borrower rates since the plot was dominated by outliers, so an additional visualization will be made with the outliers excluded:

After excluding outliers it is clear Loan Status has some effect on relationship between Stated Monthly Income and borrower rates. Since excecpt for loan status where 'Past Due (>120 days)' which exhibit moderate positive correlation between tated Monthly Income and borrower rates , most of the loan status such as Completed', 'Current', 'Past Due (1-15 days)', 'Defaulted','Chargedoff', 'Past Due (16-30 days)', 'Past Due (61-90 days)', 'Past Due (31-60 days)', 'Past Due (91-120 days)', 'FinalPaymentInProgress' which show oderate negative correlation between tated Monthly Income and borrower rates.

Checking relationship between LoanStatus, BorrowerAPR and IsBorrowerHomeowner

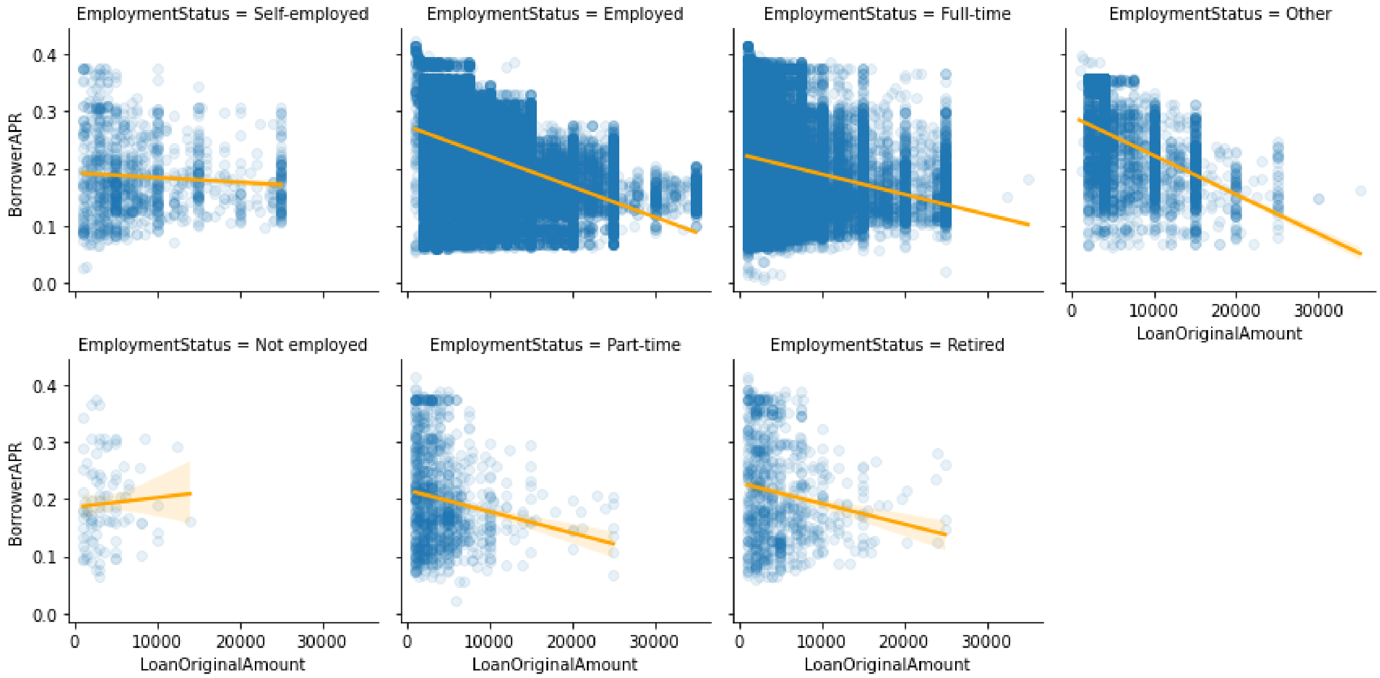
In [82]:
```
MyBarPlot(df_c, 'LoanStatus','BorrowerAPR',hue='IsBorrowerHomeowner')
plt.xticks(rotation=20);
```



Whether borrower is homeowner affect the relationship between loan status and Borrower's Annual Percentage Rate (APR). It seems borrowers who have competed loan and the current have lower Borrower's Annual Percentage Rate (APR) as compared to those who have defaulted. Homeowners borrowers who have completed,current and those who have defaulted their loans have lower borrower's annual percentage rate (APR) than those who are not homeowners.|

Checking relationship between LoanOriginalAmount, BorrowerAPR and EmploymentStatus

In [84]:
```
g = sb.FacetGrid(data = df_c, col = 'EmploymentStatus', col_wrap = 4)
g.map(sb.regplot,'LoanOriginalAmount', 'BorrowerAPR', x_jitter = 0.03, scatter_kws = {'a
```



We can see that employment status has impact on the correlation between the original loan amount and The Borrower's Annual Percentage Rate (APR) for the loan. There is strong negative correlation between the original loan amount and The Borrower's Annual Percentage Rate (APR) for employed, full-time borrowers while the others can be characterized as moderately negative. However borrowes who are not employed exhibit no correlation or weak positive correlation.

Conclusion

We can see that there is high multicollinearity between LenderYield, BorrowerAPR and BorrowerRate and so LenderYield may be removed since it is redundant. Factors that affect borrower's APR or interest rate are mainly LoanOriginalAmount followed by MonthlyLoanPayment, DelinquenciesLast7Years, StatedMonthlyIncome,IsBorrowerHomeowner,ListingCategory

The high income earners such as those who earn above `$100,000` got access to larger sizes of loans followed by those who earn between `$75,000-99,999`. On other hand low income earners such as those earning between `$1-$25,000` had access to the smallest size of loans. It seems the size of income is directly proportional to amount of loan that one can access the the higher the income the higher the loan that one can access, the lower the income the lower the loan that one can access.

There is is quite relatationship between employement status and loan amount. Those who are employed have access to higher sizes of loans, followed by those who are self-employed. Also full time borrowers could access higher loan than part-time borrowers. Persons who are not employed could access the smallest sizes of loans.

Employement status has no significant effect on term of loan since most of the borrowers prefer 36 months plan irrespective of employment status. Most of the employed indivuals prefer 36 months term and some 60 months plan. Most of borrowers who take 36 months plan are those who are employed followed by full time and others.

IsBorrowerHomeowner variable affect the relationship between loan status and Borrower's Annual Percentage Rate (APR). It seems borrowers who have competed loan and the current have lower Borrower's Annual Percentage Rate (APR) as compared to those who have defaulted. Homeowners borrowers who have completed,current and those who have defaulted their loans have lower borrower's annual percentage rate (APR) than those who are not homeowners.|

In [ ]: