Project 7

Communicate Data Findings

In this Project i am going to choose the dataset Loan Data from Prosper (https://s3.amazonaws.com/udacity-hosted-downloads/ud651/prosperLoanData.csv&sa=D&ust=1554486256021000), and then perform Exploratory and Explanotory Data Analysis. I am also going to write down the steps on how i did the project

Lets import the libraries 😊

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
%matplotlib inline
```

Now lets read the downloaded dataset into a pandas dataframe 🗄

In [2]:
```
loan=pd.read_csv('prosperLoanData.csv')
```

Now, lets go ahead and access our data, visually and programatically 💻

1. Lets take a random sample of 5 to see the structure of the Data

In [3]:
```
loan.sample(5)
```

Out[3]:

	ListingKey	ListingNumber	ListingCreationDate	CreditGrade	Term	Loan
63488	B29234100154252104B9022	267360	2008-01-17 19:47:34.063000000	A	36	Com
83877	1FF63589331915539C51225	898719	2013-09-11 07:18:29.247000000	NaN	60	C
75069	3A6F3550979242211184848	608541	2012-07-07 16:16:03.303000000	NaN	36	Char
58194	251F35313326281524DA9DD	538311	2011-11-09 18:25:48.973000000	NaN	36	Com
57366	F4933604395712050327160	1212506	2014-02-27 13:10:49.983000000	NaN	36	C

5 rows × 81 columns

On the first glance we can clearly see that, there are total of 81 columns, and clearly the CreditGrade Column has Null values. Lets check how many rows and column this data set has and the data types of all using the .info() command

In [4]:
```
loan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 81 columns):
ListingKey                           113937 non-null object
ListingNumber                        113937 non-null int64
ListingCreationDate                  113937 non-null object
CreditGrade                          28953 non-null object
Term                                 113937 non-null int64
LoanStatus                           113937 non-null object
ClosedDate                           55089 non-null object
BorrowerAPR                          113912 non-null float64
BorrowerRate                         113937 non-null float64
LenderYield                          113937 non-null float64
EstimatedEffectiveYield              84853 non-null float64
EstimatedLoss                        84853 non-null float64
EstimatedReturn                      84853 non-null float64
ProsperRating (numeric)              84853 non-null float64
ProsperRating (Alpha)                84853 non-null object
ProsperScore                         84853 non-null float64
ListingCategory (numeric)            113937 non-null int64
BorrowerState                        108422 non-null object
Occupation                           110349 non-null object
EmploymentStatus                     111682 non-null object
EmploymentStatusDuration             106312 non-null float64
IsBorrowerHomeowner                  113937 non-null bool
CurrentlyInGroup                     113937 non-null bool
GroupKey                             13341 non-null object
DateCreditPulled                     113937 non-null object
CreditScoreRangeLower                113346 non-null float64
CreditScoreRangeUpper                113346 non-null float64
FirstRecordedCreditLine              113240 non-null object
CurrentCreditLines                   106333 non-null float64
OpenCreditLines                      106333 non-null float64
TotalCreditLinespast7years           113240 non-null float64
OpenRevolvingAccounts                113937 non-null int64
OpenRevolvingMonthlyPayment          113937 non-null float64
InquiriesLast6Months                 113240 non-null float64
TotalInquiries                       112778 non-null float64
CurrentDelinquencies                 113240 non-null float64
AmountDelinquent                     106315 non-null float64
DelinquenciesLast7Years              112947 non-null float64
PublicRecordsLast10Years             113240 non-null float64
PublicRecordsLast12Months            106333 non-null float64
RevolvingCreditBalance               106333 non-null float64
BankcardUtilization                  106333 non-null float64
AvailableBankcardCredit              106393 non-null float64
TotalTrades                          106393 non-null float64
TradesNeverDelinquent (percentage)   106393 non-null float64
TradesOpenedLast6Months              106393 non-null float64
DebtToIncomeRatio                    105383 non-null float64
IncomeRange                          113937 non-null object
IncomeVerifiable                     113937 non-null bool
StatedMonthlyIncome                  113937 non-null float64
LoanKey                              113937 non-null object
TotalProsperLoans                    22085 non-null float64
TotalProsperPaymentsBilled           22085 non-null float64
OnTimeProsperPayments                22085 non-null float64
```

```
ProsperPaymentsLessThanOneMonthLate    22085 non-null float64
ProsperPaymentsOneMonthPlusLate        22085 non-null float64
ProsperPrincipalBorrowed               22085 non-null float64
ProsperPrincipalOutstanding            22085 non-null float64
ScorexChangeAtTimeOfListing            18928 non-null float64
LoanCurrentDaysDelinquent              113937 non-null int64
LoanFirstDefaultedCycleNumber          16952 non-null float64
LoanMonthsSinceOrigination             113937 non-null int64
LoanNumber                             113937 non-null int64
LoanOriginalAmount                     113937 non-null int64
LoanOriginationDate                    113937 non-null object
LoanOriginationQuarter                 113937 non-null object
MemberKey                              113937 non-null object
MonthlyLoanPayment                     113937 non-null float64
LP_CustomerPayments                    113937 non-null float64
LP_CustomerPrincipalPayments           113937 non-null float64
LP_InterestandFees                     113937 non-null float64
LP_ServiceFees                         113937 non-null float64
LP_CollectionFees                      113937 non-null float64
LP_GrossPrincipalLoss                  113937 non-null float64
LP_NetPrincipalLoss                    113937 non-null float64
LP_NonPrincipalRecoverypayments        113937 non-null float64
PercentFunded                          113937 non-null float64
Recommendations                        113937 non-null int64
InvestmentFromFriendsCount             113937 non-null int64
InvestmentFromFriendsAmount            113937 non-null float64
Investors                              113937 non-null int64
dtypes: bool(3), float64(50), int64(11), object(17)
memory usage: 68.1+ MB
```

In [5]:
```
loan.isnull().sum().head(60)
```

Out[5]:

ListingKey	0
ListingNumber	0
ListingCreationDate	0
CreditGrade	84984
Term	0
LoanStatus	0
ClosedDate	58848
BorrowerAPR	25
BorrowerRate	0
LenderYield	0
EstimatedEffectiveYield	29084
EstimatedLoss	29084
EstimatedReturn	29084
ProsperRating (numeric)	29084
ProsperRating (Alpha)	29084
ProsperScore	29084
ListingCategory (numeric)	0
BorrowerState	5515
Occupation	3588
EmploymentStatus	2255
EmploymentStatusDuration	7625
IsBorrowerHomeowner	0
CurrentlyInGroup	0
GroupKey	100596
DateCreditPulled	0
CreditScoreRangeLower	591
CreditScoreRangeUpper	591
FirstRecordedCreditLine	697
CurrentCreditLines	7604
OpenCreditLines	7604
TotalCreditLinespast7years	697
OpenRevolvingAccounts	0
OpenRevolvingMonthlyPayment	0
InquiriesLast6Months	697
TotalInquiries	1159
CurrentDelinquencies	697
AmountDelinquent	7622
DelinquenciesLast7Years	990
PublicRecordsLast10Years	697
PublicRecordsLast12Months	7604
RevolvingCreditBalance	7604
BankcardUtilization	7604
AvailableBankcardCredit	7544
TotalTrades	7544
TradesNeverDelinquent (percentage)	7544
TradesOpenedLast6Months	7544
DebtToIncomeRatio	8554
IncomeRange	0
IncomeVerifiable	0
StatedMonthlyIncome	0
LoanKey	0
TotalProsperLoans	91852
TotalProsperPaymentsBilled	91852
OnTimeProsperPayments	91852
ProsperPaymentsLessThanOneMonthLate	91852
ProsperPaymentsOneMonthPlusLate	91852
ProsperPrincipalBorrowed	91852

```
ProsperPrincipalOutstanding          91852
ScorexChangeAtTimeOfListing          95009
LoanCurrentDaysDelinquent                0
dtype: int64
```

Now, we can clearly see that there are so many columns that has null values in it. So i am going to choose from those column which are relevant and dont have Null values
To do so, we are going to first, store a list of columns that we are going to use like BorrowerAPR which means Annual Percentage Rate, Loan status which would mean the status of the Loan Given, and much more

In [6]:
```
columns= [ 'ListingNumber',
           'ListingCategory (numeric)',
          'Term',
          'LoanStatus',
           'BorrowerAPR',
          'BorrowerRate',
           'ProsperRating (Alpha)',
           'ProsperRating (numeric)',
           'Occupation',
           'EmploymentStatus',
           'EmploymentStatusDuration',
           'IsBorrowerHomeowner',
           'IncomeVerifiable',
           'StatedMonthlyIncome',
           'MonthlyLoanPayment',
           'Recommendations',
           'DebtToIncomeRatio',
           'LoanOriginalAmount' ,
           'PercentFunded',
           'IncomeRange',
           'Investors',
           'BorrowerState'
          ]
```

In [7]:
```
loan=loan[columns]
```

Now, lets check for Duplicate Data

In [8]:
```
row=loan.shape[0]
```

In [9]:
```
nrow=loan.drop_duplicates().shape[0]
```

In [10]:
```
(row-nrow)/row*100
```

Out[10]: 0.7644575511028024

Now, lets check for the data dtypes

In [11]:
```
loan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 22 columns):
ListingNumber                113937 non-null int64
ListingCategory (numeric)    113937 non-null int64
Term                         113937 non-null int64
LoanStatus                   113937 non-null object
BorrowerAPR                  113912 non-null float64
BorrowerRate                 113937 non-null float64
ProsperRating (Alpha)        84853 non-null object
ProsperRating (numeric)      84853 non-null float64
Occupation                   110349 non-null object
EmploymentStatus             111682 non-null object
EmploymentStatusDuration     106312 non-null float64
IsBorrowerHomeowner          113937 non-null bool
IncomeVerifiable             113937 non-null bool
StatedMonthlyIncome          113937 non-null float64
MonthlyLoanPayment           113937 non-null float64
Recommendations              113937 non-null int64
DebtToIncomeRatio            105383 non-null float64
LoanOriginalAmount           113937 non-null int64
PercentFunded                113937 non-null float64
IncomeRange                  113937 non-null object
Investors                    113937 non-null int64
BorrowerState                108422 non-null object
dtypes: bool(2), float64(8), int64(6), object(6)
memory usage: 17.6+ MB
```

Quality Issues

1. Incorrect Data TYpe for Loan Status,Prosper rating Alpha, Income Range, IsBorrowerHomeowner, Prosper rating numeric, Employment Status, Income Verifiable, Term.
2. Lots of Null values.
3. 0.76% of the Data is duplicated.

Cleaning the Data

Note: Before going ahead with the further steps we must always keep a copy of our data for safety

In [12]:
```
cleanloan=loan.copy()
```

Define

Droping Duplicates from the data frame.

Code

In [13]:
```
cleanloan.drop_duplicates(inplace=True)
```

Test

In [14]:
```
cleanloan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 113066 entries, 0 to 113936
Data columns (total 22 columns):
ListingNumber                113066 non-null int64
ListingCategory (numeric)    113066 non-null int64
Term                         113066 non-null int64
LoanStatus                   113066 non-null object
BorrowerAPR                  113041 non-null float64
BorrowerRate                 113066 non-null float64
ProsperRating (Alpha)        83982 non-null object
ProsperRating (numeric)      83982 non-null float64
Occupation                   109537 non-null object
EmploymentStatus             110811 non-null object
EmploymentStatusDuration     105441 non-null float64
IsBorrowerHomeowner          113066 non-null bool
IncomeVerifiable             113066 non-null bool
StatedMonthlyIncome          113066 non-null float64
MonthlyLoanPayment           113066 non-null float64
Recommendations              113066 non-null int64
DebtToIncomeRatio            104594 non-null float64
LoanOriginalAmount           113066 non-null int64
PercentFunded                113066 non-null float64
IncomeRange                  113066 non-null object
Investors                    113066 non-null int64
BorrowerState                107551 non-null object
dtypes: bool(2), float64(8), int64(6), object(6)
memory usage: 18.3+ MB
```

Define

Droping Null Values.

Code

In [15]:
```
cleanloan.dropna(inplace=True)
```

Test

In [16]:
```
cleanloan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 75486 entries, 1 to 113936
Data columns (total 22 columns):
ListingNumber                75486 non-null int64
ListingCategory (numeric)    75486 non-null int64
Term                         75486 non-null int64
LoanStatus                   75486 non-null object
BorrowerAPR                  75486 non-null float64
BorrowerRate                 75486 non-null float64
ProsperRating (Alpha)        75486 non-null object
ProsperRating (numeric)      75486 non-null float64
Occupation                   75486 non-null object
EmploymentStatus             75486 non-null object
EmploymentStatusDuration     75486 non-null float64
IsBorrowerHomeowner          75486 non-null bool
IncomeVerifiable             75486 non-null bool
StatedMonthlyIncome          75486 non-null float64
MonthlyLoanPayment           75486 non-null float64
Recommendations              75486 non-null int64
DebtToIncomeRatio            75486 non-null float64
LoanOriginalAmount           75486 non-null int64
PercentFunded                75486 non-null float64
IncomeRange                  75486 non-null object
Investors                    75486 non-null int64
BorrowerState                75486 non-null object
dtypes: bool(2), float64(8), int64(6), object(6)
memory usage: 12.2+ MB
```

Define.

Changing Datatypes

Code

In [17]:
```
cleanloan['ProsperRating (numeric)']=cleanloan['ProsperRating (numeric)'].asty
pe('category')
cleanloan.IncomeRange=cleanloan.IncomeRange.astype('category')
cleanloan.IsBorrowerHomeowner=cleanloan.IsBorrowerHomeowner.astype('bool')
```

Test

In [18]:
```
cleanloan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 75486 entries, 1 to 113936
Data columns (total 22 columns):
ListingNumber                75486 non-null int64
ListingCategory (numeric)    75486 non-null int64
Term                         75486 non-null int64
LoanStatus                   75486 non-null object
BorrowerAPR                  75486 non-null float64
BorrowerRate                 75486 non-null float64
ProsperRating (Alpha)        75486 non-null object
ProsperRating (numeric)      75486 non-null category
Occupation                   75486 non-null object
EmploymentStatus             75486 non-null object
EmploymentStatusDuration     75486 non-null float64
IsBorrowerHomeowner          75486 non-null bool
IncomeVerifiable             75486 non-null bool
StatedMonthlyIncome          75486 non-null float64
MonthlyLoanPayment           75486 non-null float64
Recommendations              75486 non-null int64
DebtToIncomeRatio            75486 non-null float64
LoanOriginalAmount           75486 non-null int64
PercentFunded                75486 non-null float64
IncomeRange                  75486 non-null category
Investors                    75486 non-null int64
BorrowerState                75486 non-null object
dtypes: bool(2), category(2), float64(7), int64(6), object(5)
memory usage: 11.2+ MB
```

In [19]:
```
cleanloan=cleanloan.drop('Recommendations',axis=1)
```

What is the structure of your dataset?

After cleaning my dataset, the structure of my dataset is reduced to 75486x20

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

I want to find out which people are able to pay loan.

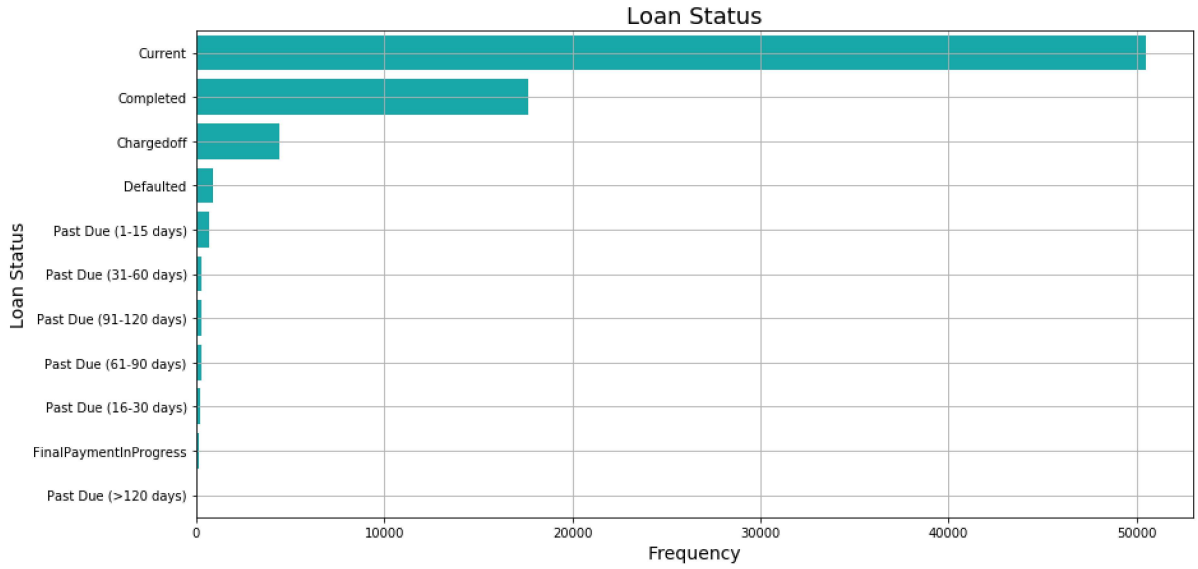
What features in the dataset do you think will help support your investigation into your feature(s) of interest?

I think many features such as ISBorrowerHomeowner, Occupation, Employment Status, Employment status Duration, Income range will help us to predict our outcome

Univariate Exploration

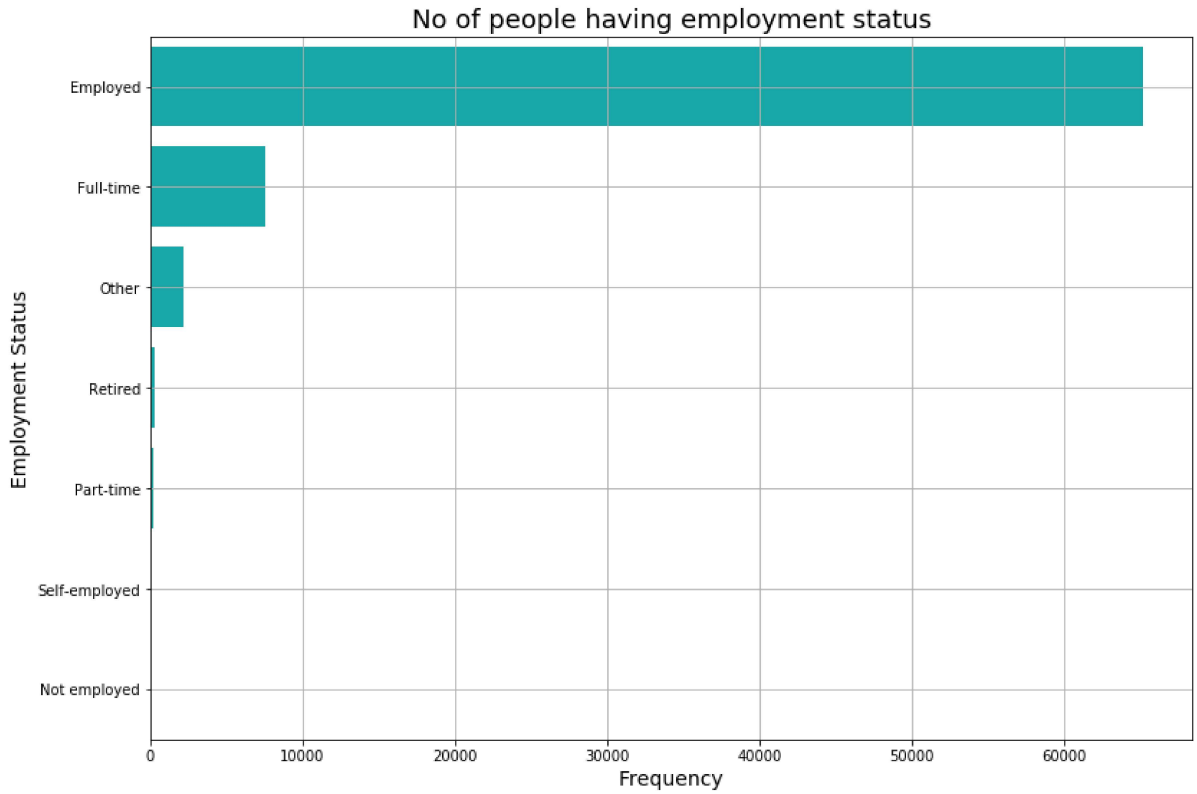
> In this section, Lets investigate individual variables so that we can notice some insights

In [20]:
```
plt.figure(figsize=[14,7])
countstatus=cleanloan.LoanStatus.value_counts().index
sb.countplot(data=cleanloan,y='LoanStatus',order=countstatus,color='c');
plt.xlabel('Frequency', fontsize=14);
plt.ylabel('Loan Status', fontsize=14)
plt.title('Loan Status', fontsize=18)
plt.grid();
```



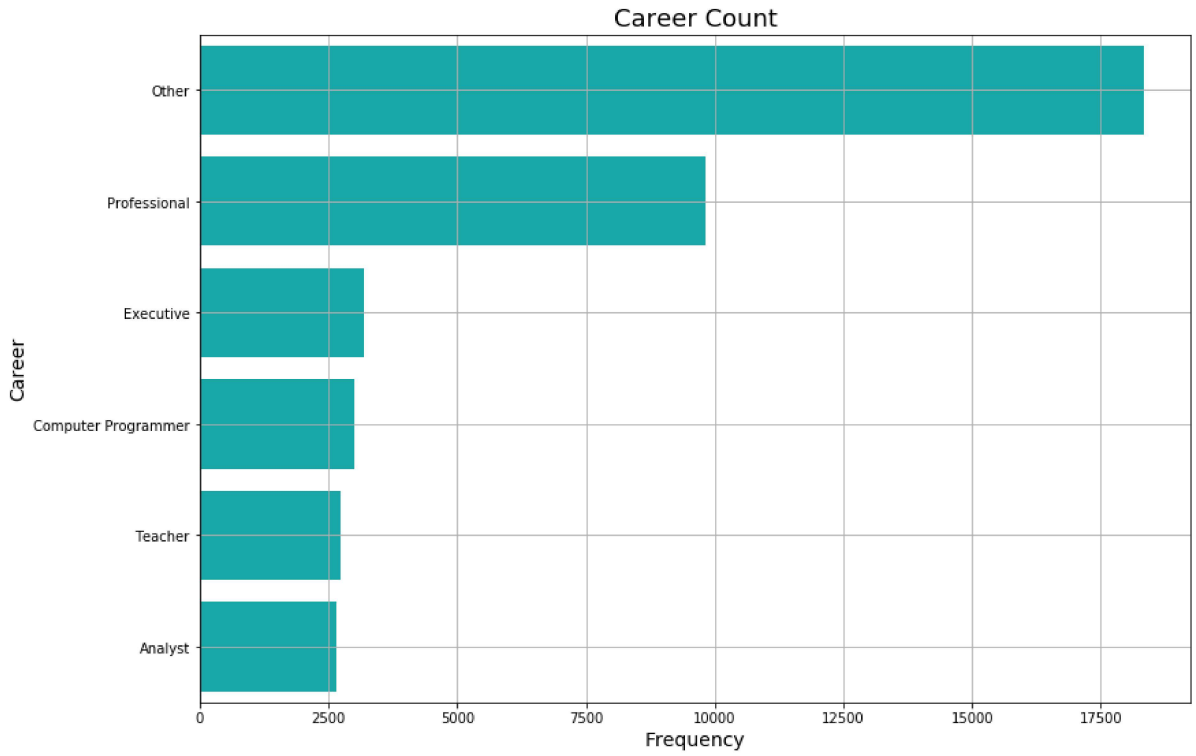
We can see most of the people have current loan status

In [21]:
```
plt.figure(figsize=[13,9])
sorted_counts = cleanloan['EmploymentStatus'].value_counts().index
sb.countplot(data = cleanloan, y = 'EmploymentStatus', color = 'c',order=sorte
d_counts);
plt.xlabel('Frequency', fontsize=14)
plt.ylabel('Employment Status', fontsize=14)
plt.title('No of people having employment status', fontsize=18);
plt.grid();
```



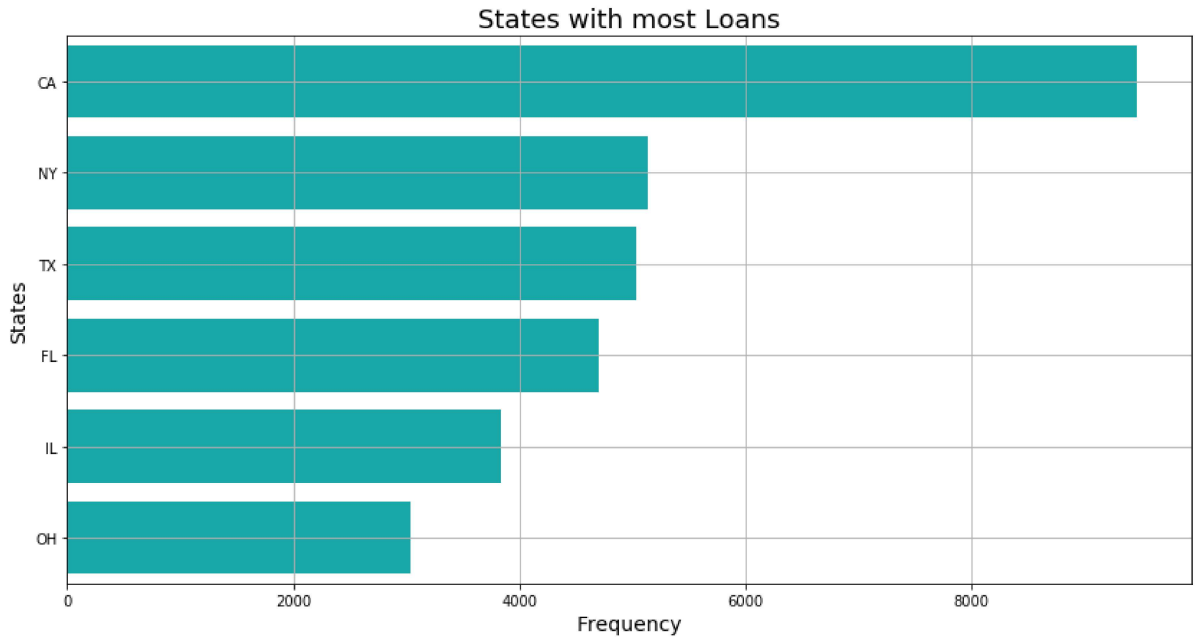
Clearly we can see that most of the people are employed and there is a very less fracticion who are not employed

In [22]:
```
plt.figure(figsize=[13,9])
sorted_occu = cleanloan['Occupation'].value_counts().head(6).index
sb.countplot(data = cleanloan, y = 'Occupation', color = 'c',order=sorted_occu
);
plt.xlabel('Frequency', fontsize=14)
plt.ylabel('Career', fontsize=14)
plt.title('Career Count', fontsize=18);
plt.grid();
```



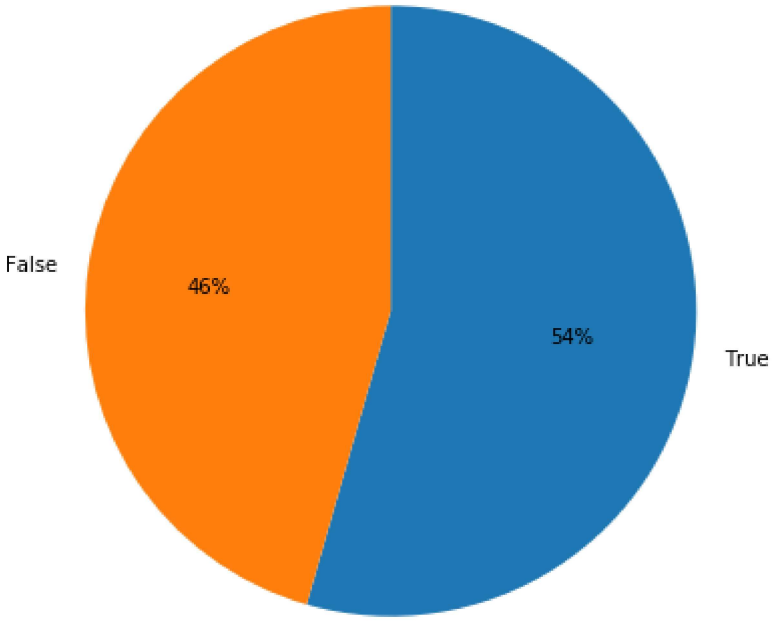
Most of the people belong to the Other Profession

In [23]:
```
plt.figure(figsize=[14,7])
sorted_state = cleanloan['BorrowerState'].value_counts().head(6).index
sb.countplot(data = cleanloan, y = 'BorrowerState', color = 'c',order=sorted_s
tate);
plt.xlabel('Frequency', fontsize=14)
plt.ylabel('States', fontsize=14)
plt.title('States with most Loans', fontsize=18);
plt.grid();
```



Most of the people taking loan belong to California which is the most economic state of US, followed by New York which is the second most economic city of USA

In [24]:
```
plt.figure(figsize=[14,7])
sorted_countshome = cleanloan['IsBorrowerHomeowner'].value_counts()
plt.pie(sorted_countshome, labels = sorted_countshome.index, startangle = 90,
        counterclock = False,autopct='%1.0f%%');
```



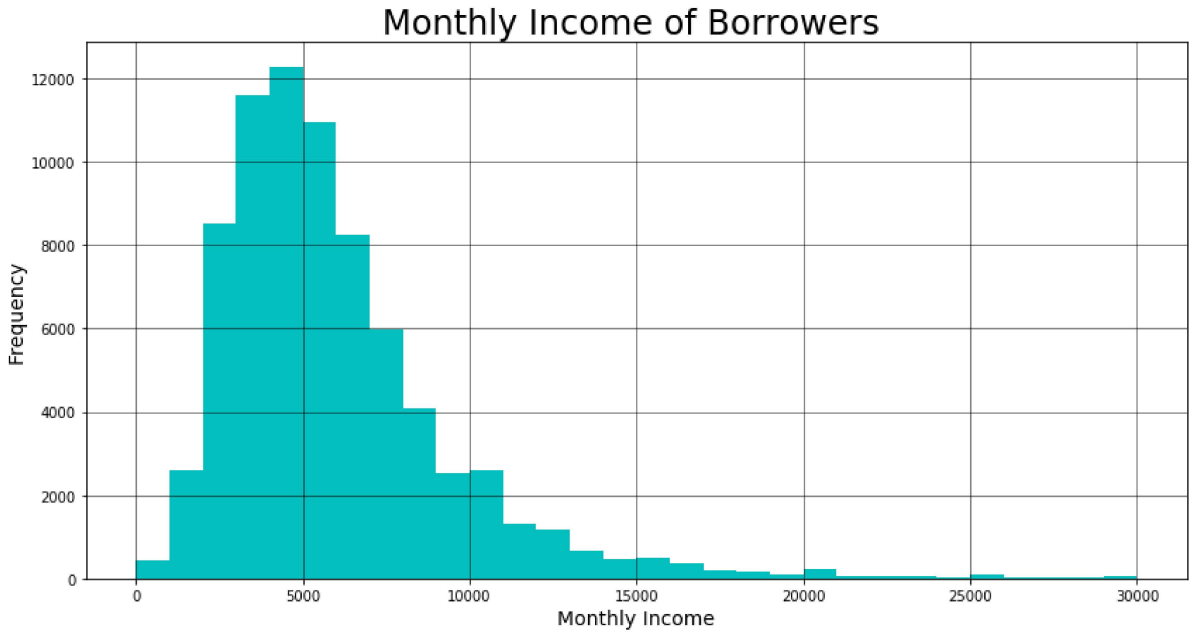
54% of people do have their own Home

Lets focus on finding about the people who have income less than 30000 and see what are the factors that influence the ratings

In [25]:
```
poor=cleanloan.query('StatedMonthlyIncome < 30000')
```

In [26]:
```
k=poor.MonthlyLoanPayment.max()
```

In [27]:
```
plt.figure(figsize=[14,7])
bins = np.arange(0, k+28000, 1000)
plt.hist(data = cleanloan, x = 'StatedMonthlyIncome', bins = bins, color='c');
plt.xlabel('Monthly Income', fontsize=14);
plt.ylabel('Frequency', fontsize=14);
plt.grid(color='black',alpha=0.6)
plt.title("Monthly Income of Borrowers ", fontsize=24);
```

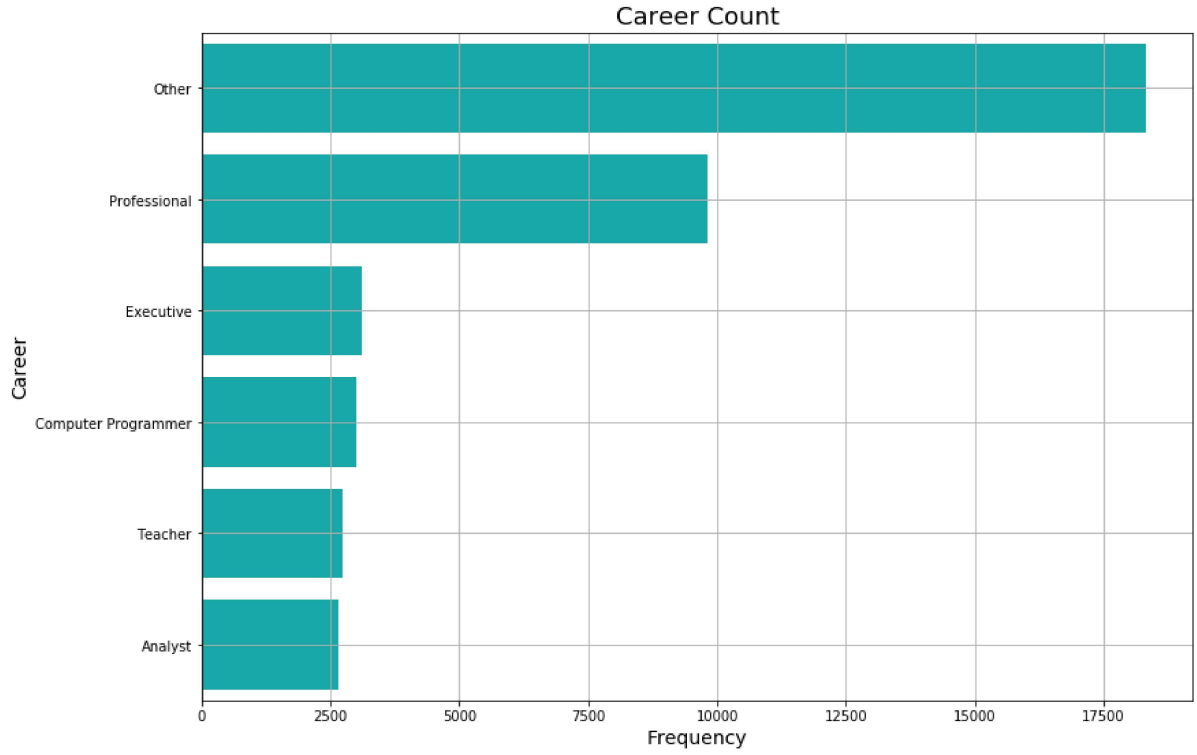


In [28]:
```
poor.StatedMonthlyIncome.describe()
```

Out[28]:
```
count    75283.000000
mean      5882.263464
std       3446.721934
min          0.250000
25%       3583.333333
50%       5000.000000
75%       7166.666667
max      29833.333333
Name: StatedMonthlyIncome, dtype: float64
```

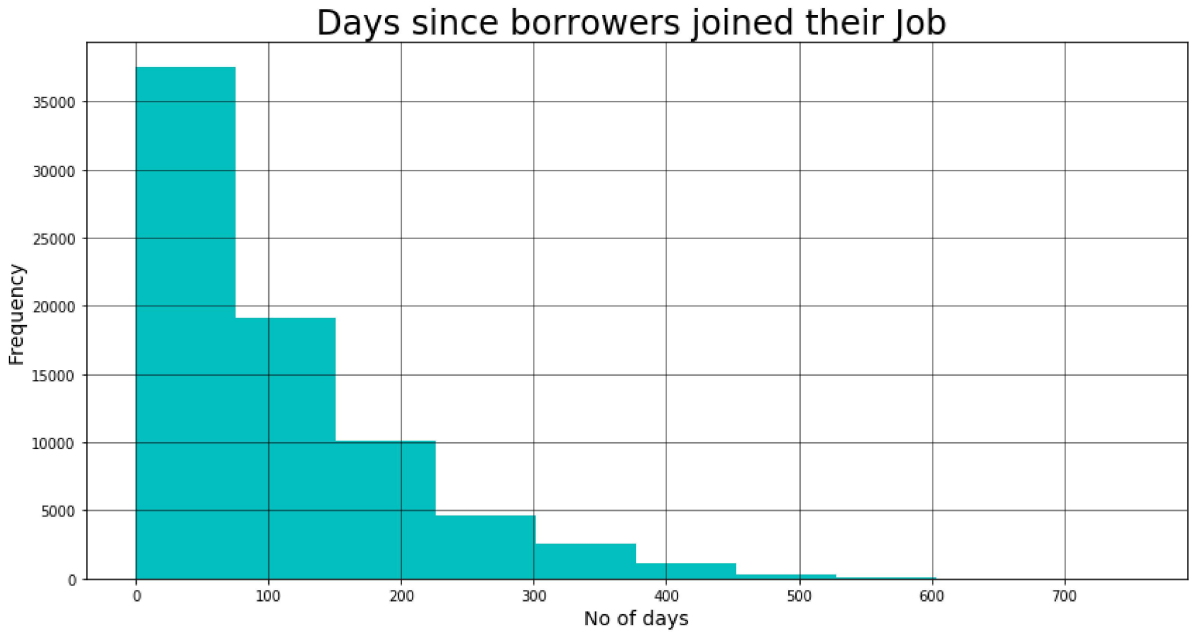
We can clearly see that most of the people take loan with the income of 5882$

In [29]:
```
plt.figure(figsize=[13,9])
sorted_occu = poor['Occupation'].value_counts().head(6).index
sb.countplot(data = poor, y = 'Occupation', color = 'c',order=sorted_occu);
plt.xlabel('Frequency', fontsize=14)
plt.ylabel('Career', fontsize=14)
plt.title('Career Count', fontsize=18);
plt.grid();
```



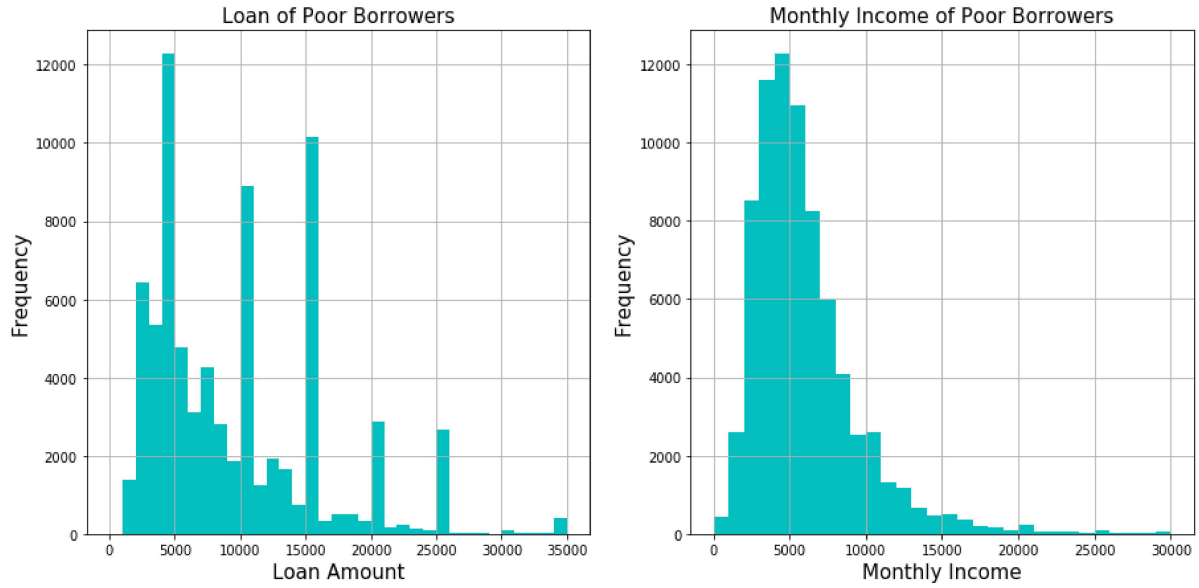
Even for poor people Other and Professional are the top choices of Employment

In [30]:
```
plt.figure(figsize=[14,7])
plt.hist(data = poor, x = 'EmploymentStatusDuration', color='c');
plt.xlabel('No of days', fontsize=14);
plt.ylabel('Frequency', fontsize=14);
plt.grid(color='black',alpha=0.6)
plt.title("Days since borrowers joined their Job ", fontsize=24);
```



We can see that most of the people are under 100 days of their jobs

In [31]:
```
plt.figure(figsize=[15,7])
plt.subplot(1, 2, 1)
ok=poor.LoanOriginalAmount.max()
bins=np.arange(0,ok+1000,1000)
plt.hist(data =poor, x = 'LoanOriginalAmount',bins=bins,color='c')
plt.xlabel('Loan Amount',fontsize=15)
plt.ylabel('Frequency',fontsize=15);
plt.title("Loan of Poor Borrowers",fontsize=15);
plt.grid()
plt.subplot(1, 2, 2)
bins = np.arange(0, k+28000, 1000)
plt.hist(data = cleanloan, x = 'StatedMonthlyIncome', bins = bins,color='c');
plt.xlabel('Monthly Income',fontsize=15);
plt.ylabel('Frequency',fontsize=15);
plt.title("Monthly Income of Poor Borrowers", fontsize=15);
plt.grid()
```



We can see that most of the people are taking loan in the range of 5000 just like the average of monthly income, but one thing to notice here is that people having 15,000 as a salary is less but still many are taking loan for that much big amount

In [32]:
```
c=poor['ListingCategory (numeric)']
c=c.value_counts()
c
```

Out[32]:
```
1     47918
7      8228
2      6249
3      3607
6      2027
13     1758
15     1368
14      783
18      770
20      718
19      706
16      288
5       201
11      198
8       187
9        83
10       82
17       49
12       44
0        19
Name: ListingCategory (numeric), dtype: int64
```

Insights

1. Most of the Poor people are taking Loans for Debt Consodilation, Home Improvemnt, Business and Household Expenses.
2. The states with highest Loans are California which is the economic Hub of US and then New York.
3. Mostly People with recent joinings upto 100 Days are taking Loans.
4. 46% of the people dont have their home.
5. There is a strange pattern to be noticed that is the average people taking loans has salary 5000 dollars, yet people are taking loans of 25000 with a high spike
6. The top choice for employment even is Others or Professional Career.
7. Most of them are currently on Loan

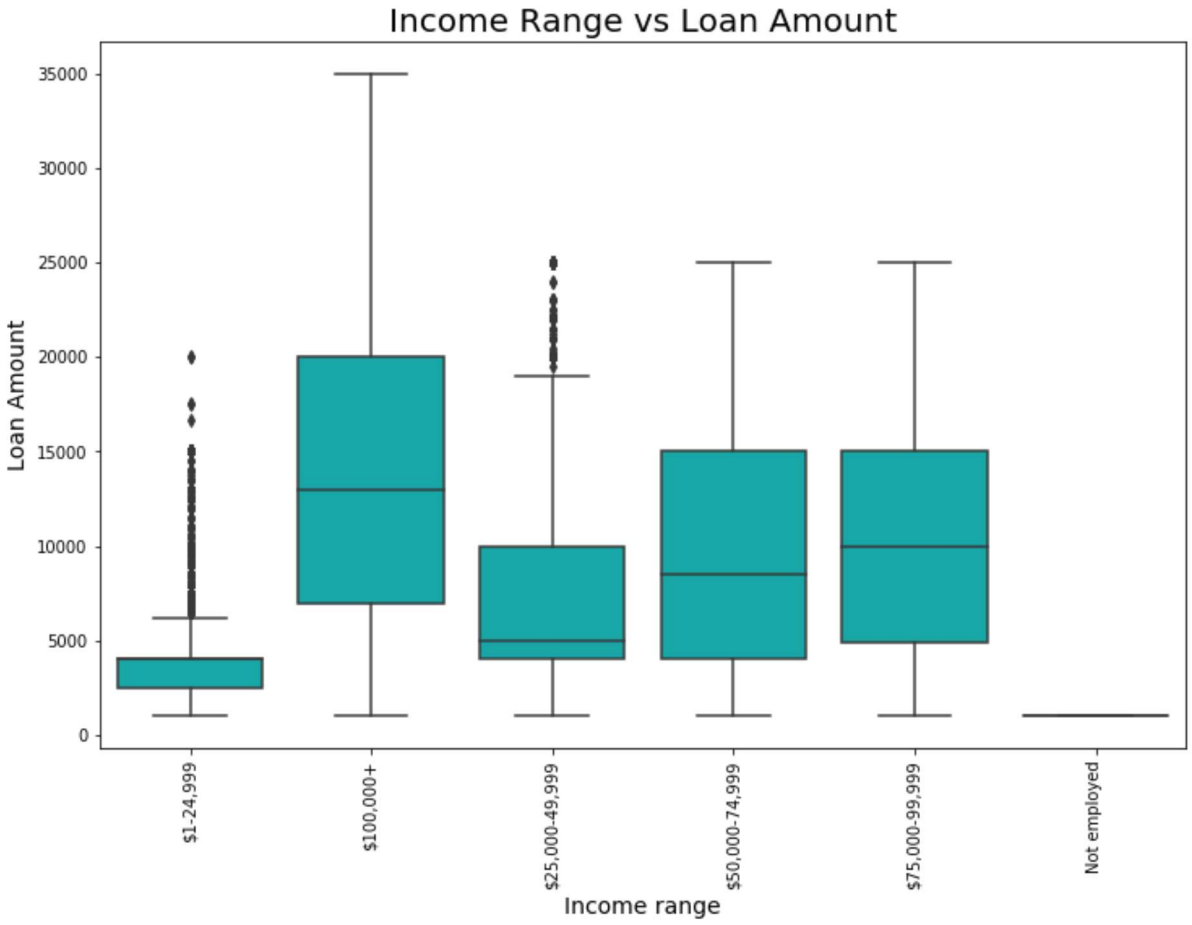
Bivariate Exploration

In this section, investigate relationships between pairs of variables in your data.

In [33]:
```
poor.IncomeRange.value_counts()
```

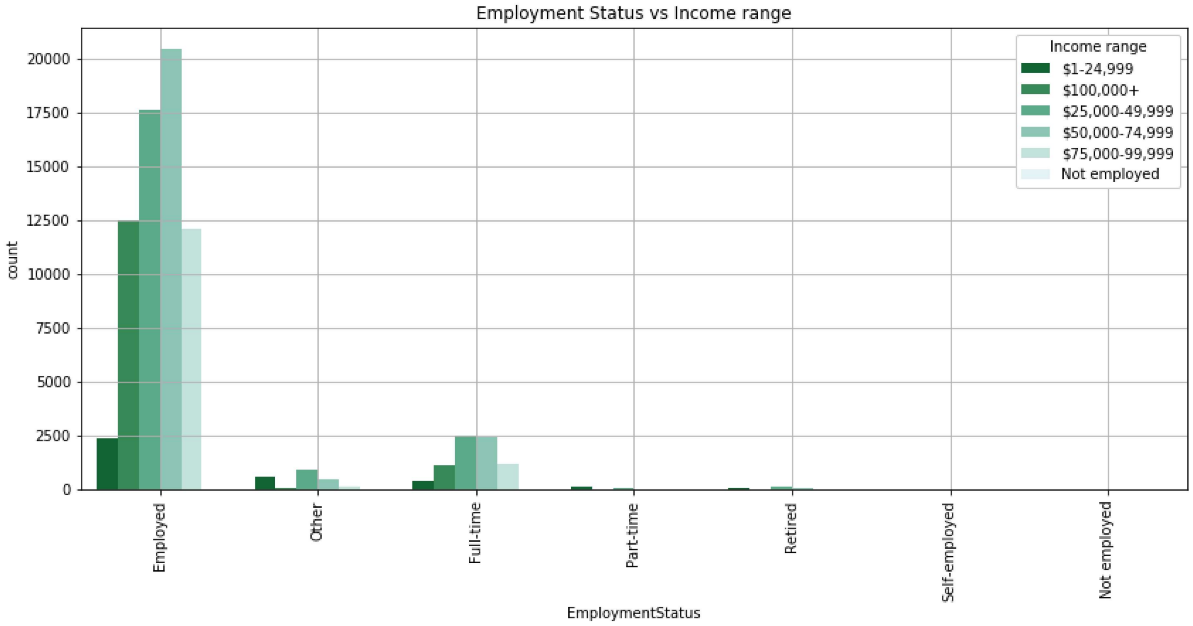
Out[33]:
```
$50,000-74,999    23432
$25,000-49,999    21219
$100,000+         13646
$75,000-99,999    13427
$1-24,999          3558
Not employed          1
Name: IncomeRange, dtype: int64
```

In [34]:
```
plt.figure(figsize=[12,8])
sb.boxplot(data = poor, x = 'IncomeRange', y = 'LoanOriginalAmount', color =
'c')
plt.xticks(rotation = 90);
plt.ylabel('Loan Amount', fontsize=14)
plt.xlabel('Income range', fontsize=14)
plt.title('Income Range vs Loan Amount', fontsize=20);
```



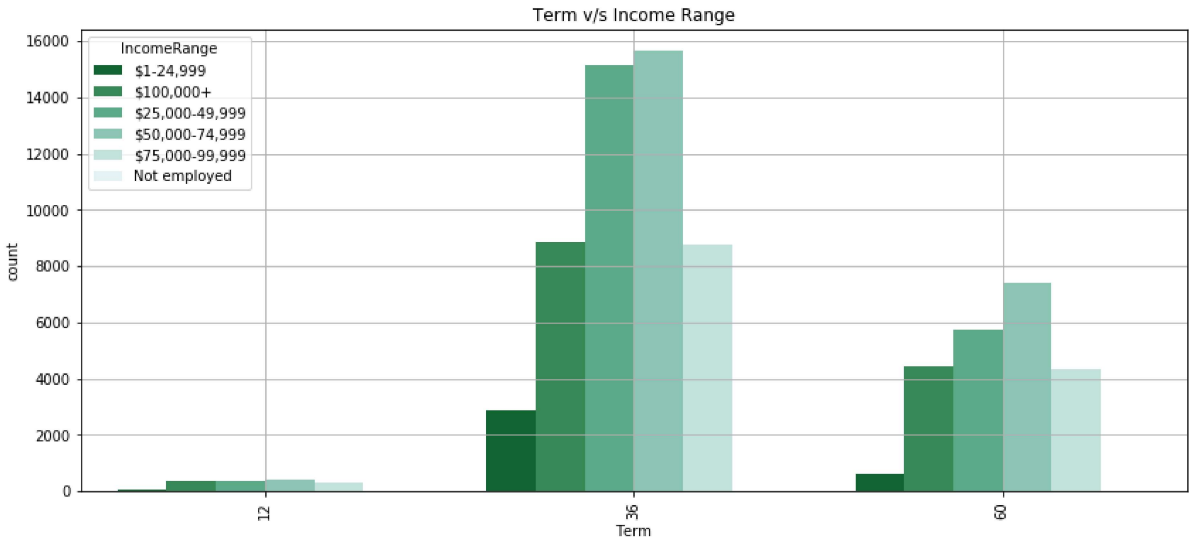
We can clearly see that the people with income higher than 100,000+ take the highest amount of Loan

In [35]:
```
plt.figure(figsize=[14,6])
sb.countplot(data = poor, x = 'EmploymentStatus', hue = 'IncomeRange', palette
= 'BuGn_r')
plt.legend(loc = 1, ncol = 1, framealpha = 1, title = 'Income range')
plt.xticks(rotation = 90)
plt.grid()
plt.title('Employment Status vs Income range');
```



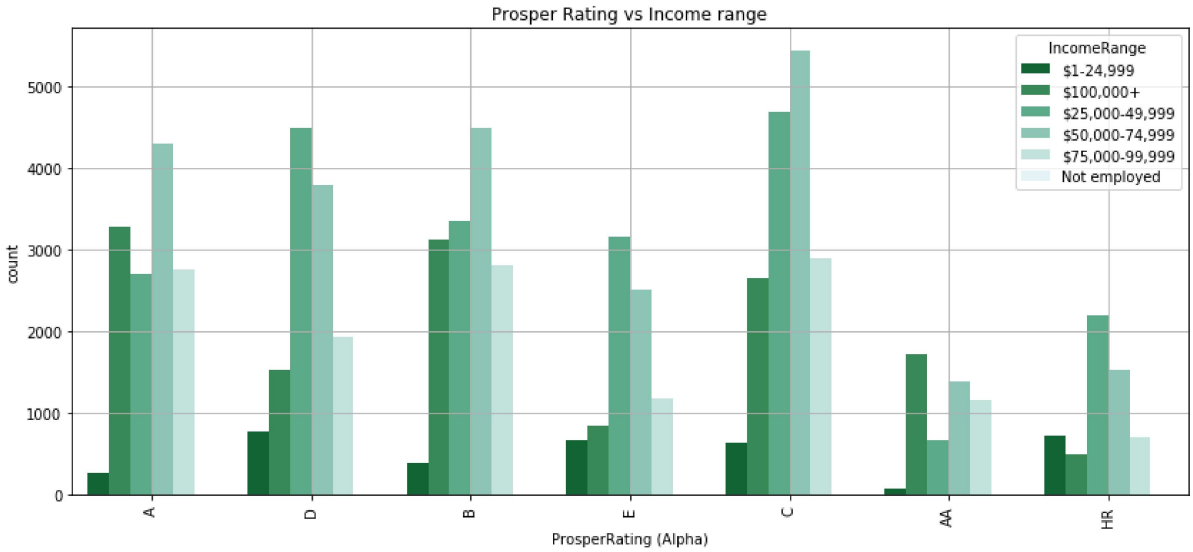
Most of the people who are employed with range of 50-75 are highest

In [36]:
```
plt.figure(figsize=[14,6])
sb.countplot(data = poor, x = 'Term', hue = 'IncomeRange', palette = 'BuGn_r')
plt.xticks(rotation = 90)
plt.grid()
plt.title('Term v/s Income Range');
```



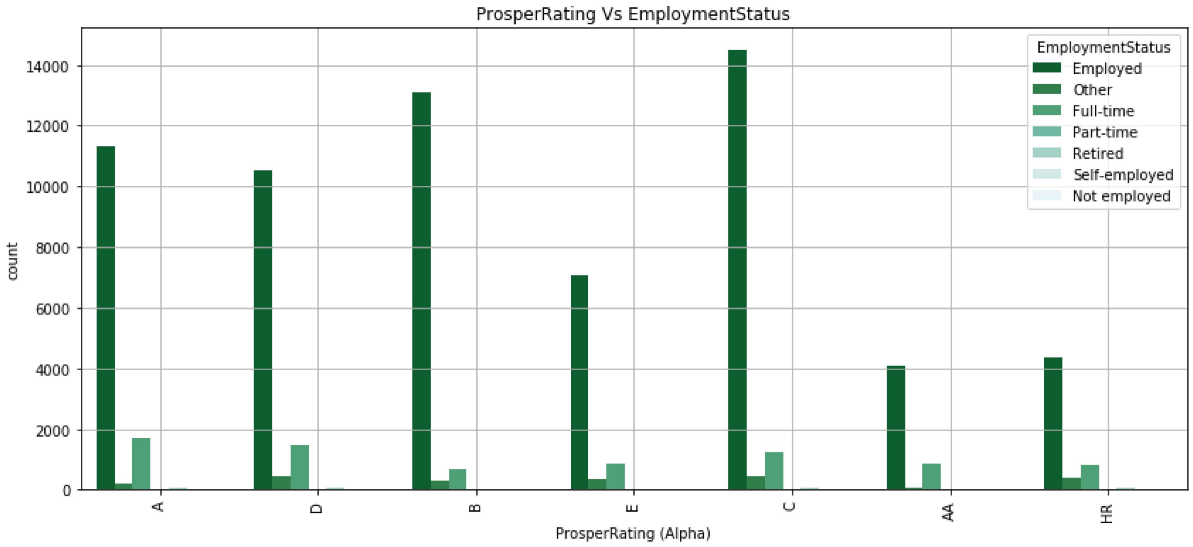
Most of the people are in the term 36, with 50-75 range

In [37]:
```
plt.figure(figsize=[14,6])
sb.countplot(data = poor, x ='ProsperRating (Alpha)' , hue = 'IncomeRange', pa
lette = 'BuGn_r')
plt.xticks(rotation = 90)
plt.grid()
plt.title('Prosper Rating vs Income range');
```

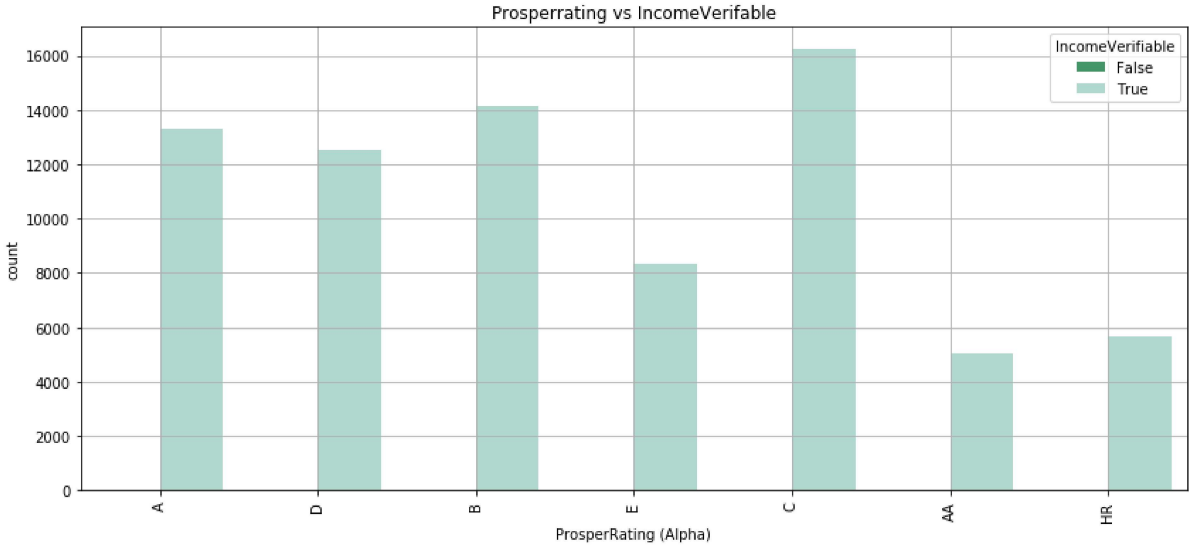


Minimum are from the rating AA

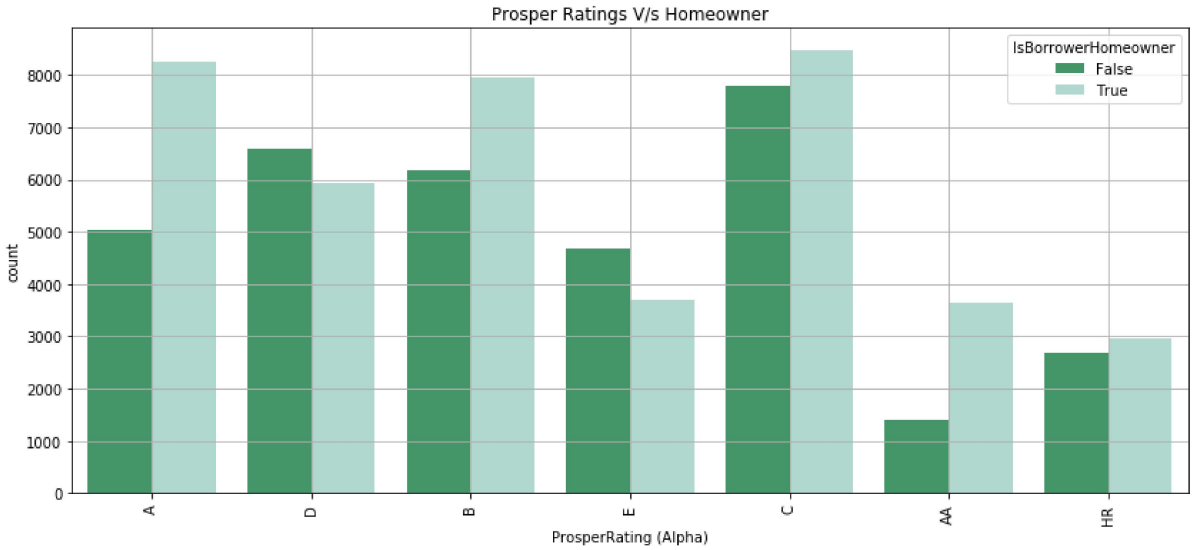
In [38]:
```
plt.figure(figsize=[14,6])
sb.countplot(data = poor, x ='ProsperRating (Alpha)' , hue = 'EmploymentStatu
s', palette = 'BuGn_r')
plt.xticks(rotation = 90)
plt.grid()
plt.title('ProsperRating Vs EmploymentStatus');
```



In [39]:
```
plt.figure(figsize=[14,6])
sb.countplot(data = poor, x ='ProsperRating (Alpha)' , hue = 'IncomeVerifiabl
e', palette = 'BuGn_r')
plt.xticks(rotation = 90)
plt.grid()
plt.title('Prosperrating vs IncomeVerifable');
```



In [40]:
```
plt.figure(figsize=[14,6])
sb.countplot(data = poor, x ='ProsperRating (Alpha)' , hue = 'IsBorrowerHomeow
ner', palette = 'BuGn_r')
plt.xticks(rotation = 90)
plt.grid()
plt.title('Prosper Ratings V/s Homeowner');
```



Majority of the D and E dont have any Home

Insights

1.LoanOriginalAmount is highest for A and B Prosper ratings, when compared with income range.

1. Majority are from the range 100+.
2. AA rating having more people not owning home compared to HR which is very strage.
3. Majority of the people are employed

Talk about some of the relationships you observed in this part of the investigation. How did the feature(s) of interest vary with other features in the dataset?

I saw that most of the borrowers with highest loan amount are taken by Employed, this is followed by others and fulltime employees and mostly it decides the rating of one.

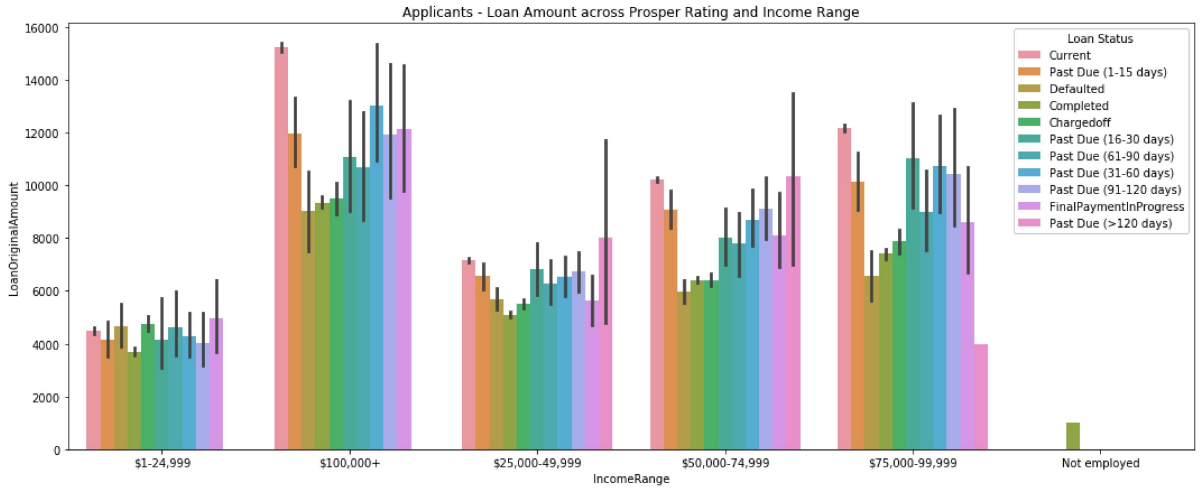
Did you observe any interesting relationships between the other features (not the main feature(s) of interest)?

Even people with 100,000+ income are in HR list which are employed and have a home

Multivariate Exploration

Does Loan Status Depends on Income?

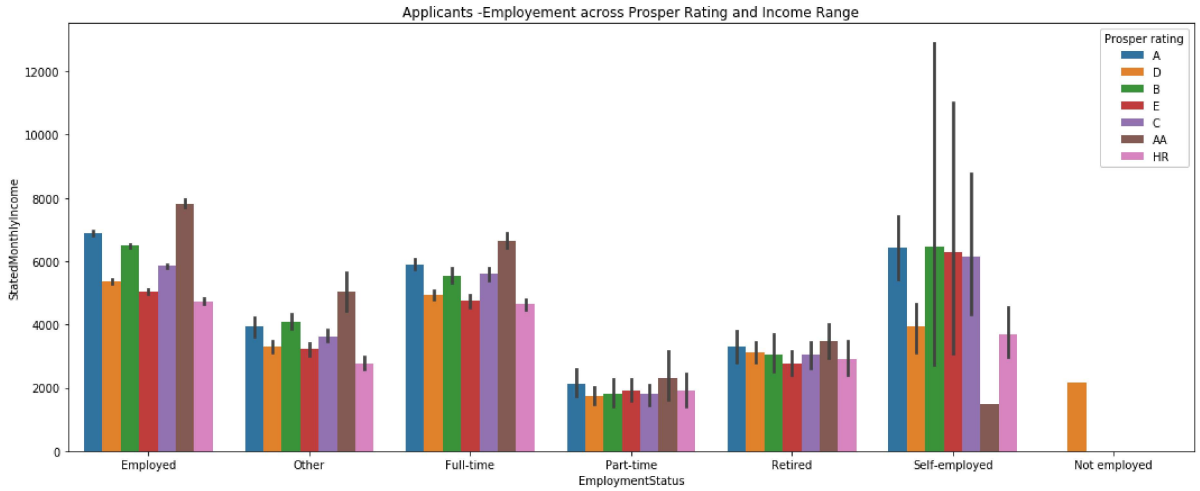
In [41]:
```
plt.figure(figsize = [18, 7])
ax = sb.barplot(data = poor, x = 'IncomeRange', y = 'LoanOriginalAmount', hue
= 'LoanStatus')
ax.legend(loc = 1, ncol = 1, framealpha = 1, title = 'Loan Status')
plt.title('Applicants - Loan Amount across Prosper Rating and Income Range');
```



Yes, clearly people with more income are having more people in current status

Does Salary decides the Ratings?

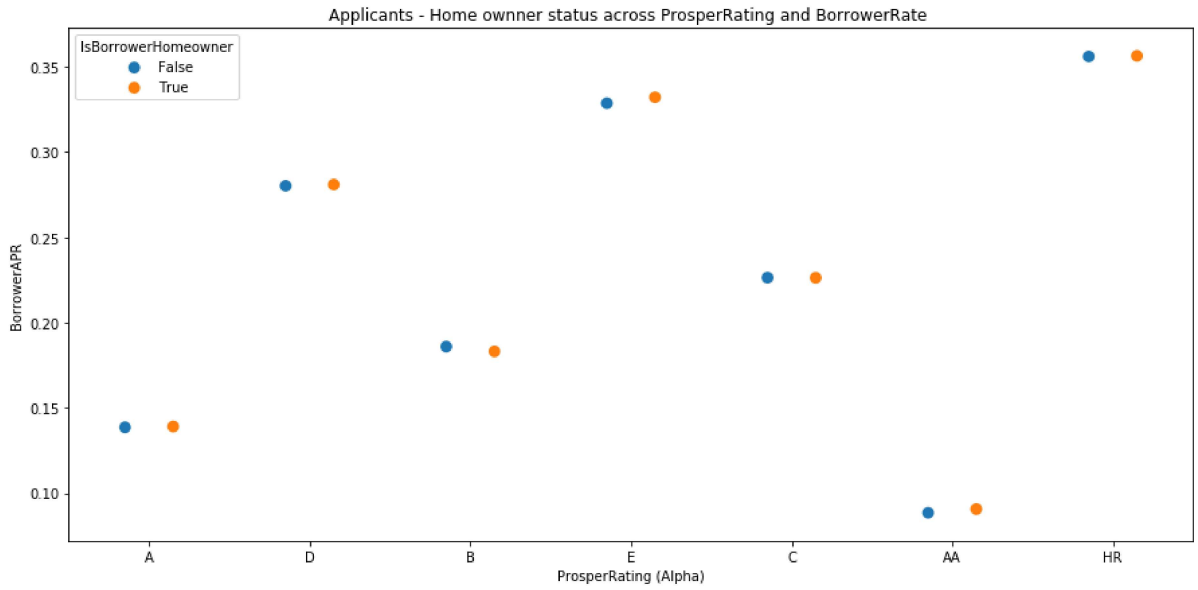
In [42]:
```
plt.figure(figsize = [18, 7])
ax = sb.barplot(data = poor, x = 'EmploymentStatus', y = 'StatedMonthlyIncome'
, hue = 'ProsperRating (Alpha)')
ax.legend(loc = 1, ncol = 1, framealpha = 1, title = 'Prosper rating')
plt.title('Applicants -Employement across Prosper Rating and Income Range');
```



Yes, but for Employed, Other and Full Time Employees, but not for others

Does APR varies with One having home and Prosper Rating?

In [43]:
```
plt.figure(figsize = [15, 7])
ax = sb.pointplot(data = poor, x = 'ProsperRating (Alpha)', y = 'BorrowerAPR',
hue = 'IsBorrowerHomeowner',
                  dodge = 0.3, linestyles = "");
plt.title('Applicants - Home ownner status across ProsperRating and BorrowerRa
te ');
```



Yes, AA has lowest APR while HR has the highest. One thing to notice that in Having a home doesnt impact rating that much

Conclusion

1. Most of the poor borrowers fall in prosper rating of B , irrespective of the income range
2. The monthly income of borrowers are having higher values for employed, other and full time employment status with the prosper rating of AA, A and B.
3. Having a home doesnt effect the interest. To conclude the analysis , I say that the loan approval status is dependent on the Income, Homeownerstatus and employment status.

In [ ]: