In [61]:

```
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
import mpld3
mpld3.enable_notebook()
```

In [47]:

```
import pandas as pd
import numpy as np
import random
import xgboost as xgb
import matplotlib.pyplot as plt
import pylab
import seaborn as sns
from pandas.tools.plotting import scatter_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score, auc,roc_curve
import pandas_profiling
from scipy import stats
```

In [7]:

```
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
pandas_profiling.ProfileReport(train)
```

```
Out[7]:
```

Overview

Dataset info

Number of variables	22
Number of observations	69713
Total Missing (%)	8.8%
Total size in memory	11.7 MiB
Average record size in memory	176.0 B

Variables types

Numeric	8
Categorical	12
Date	0
Text (Unique)	1
Rejected	1

Warnings

- `Approved` has 68693 / 98.5% zeros
- `City_Category` has 814 / 1.2% missing values **Missing**
- `City_Code` has 814 / 1.2% missing values **Missing**
- `City_Code` has a high cardinality: 679 distinct values **Warning**
- `Customer_Existing_Primary_Bank_Code` has 9391 / 13.5% missing values **Missing**
- `Customer_Existing_Primary_Bank_Code` has a high cardinality: 58 distinct values **Warning**
- `DOB` has a high cardinality: 10760 distinct values **Warning**
- `EMI` is highly correlated with `Loan_Amount` (ρ = 0.91683) **Rejected**
- `Employer_Category1` has 4018 / 5.8% missing values **Missing**
- `Employer_Category2` has 4298 / 6.2% missing values **Missing**
- `Employer_Code` has 4018 / 5.8% missing values **Missing**
- `Employer_Code` has a high cardinality: 36618 distinct values **Warning**
- `Existing_EMI` is highly skewed (γ1 = 194.93)
- `Existing_EMI` has 46621 / 66.9% zeros
- `Interest_Rate` has 47437 / 68.0% missing values **Missing**
- `Lead_Creation_Date` has a high cardinality: 92 distinct values **Warning**
- `Loan_Amount` has 27709 / 39.7% missing values **Missing**
- `Loan_Period` has 27709 / 39.7% missing values **Missing**
- `Monthly_Income` is highly skewed (γ1 = 168.42)
- `Primary_Bank_Type` has 9391 / 13.5% missing values **Missing**
- `Var1` has 23308 / 33.4% zeros

Variables

Approved

Numeric

Distinct count	2
Unique (%)	0.0%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0

Mean	0.014631
Minimum	0
Maximum	1
Zeros (%)	98.5%

Toggle details

City_Category

Categorical

Distinct count	4
Unique (%)	0.0%
Missing (%)	1.2%
Missing (n)	814

A	49885
C	11694
B	7320
(Missing)	814

Toggle details

City_Code

Categorical

Distinct count	679
Unique (%)	1.0%
Missing (%)	1.2%
Missing (n)	814

C10001	10007
C10002	8716
C10003	8666
Other values (675)	41510

Toggle details

Contacted
Categorical

Distinct count	2
Unique (%)	0.0%
Missing (%)	0.0%
Missing (n)	0

Y	45275
N	24438

Toggle details

Customer_Existing_Primary_Bank_Code
Categorical

Distinct count	58
Unique (%)	0.1%
Missing (%)	13.5%
Missing (n)	9391

B001	14197
B002	10880
B003	9515
Other values (54)	25730
(Missing)	9391

Toggle details

DOB
Categorical

Distinct count	10760
Unique (%)	15.4%
Missing (%)	0.0%
Missing (n)	15

11/01/82	253
04/03/71	183
03/03/71	121
Other values (10756)	69141

Toggle details

~~EMI~~
Highly correlated

This variable is highly correlated with `Loan_Amount` and should be ignored for analysis

Correlation	0.91683

Employer_Category1

Categorical

Distinct count	4
Unique (%)	0.0%
Missing (%)	5.8%
Missing (n)	4018

A	33336
B	18056
C	14303
(Missing)	4018

Toggle details

Employer_Category2

Numeric

Distinct count	5
Unique (%)	0.0%
Missing (%)	6.2%
Missing (n)	4298
Infinite (%)	0.0%
Infinite (n)	0

Mean	3.7202
Minimum	1
Maximum	4
Zeros (%)	0.0%

Toggle details

Employer_Code

Categorical

Distinct count	36618
Unique (%)	55.7%
Missing (%)	5.8%
Missing (n)	4018

COM0000002	457
COM0000003	324
COM0000004	262

Other values (36614)	64652
(Missing)	4018

Toggle details

Existing_EMI

Numeric

Distinct count	3246
Unique (%)	4.7%
Missing (%)	0.1%
Missing (n)	51
Infinite (%)	0.0%
Infinite (n)	0

Mean	360.93
Minimum	0
Maximum	545440
Zeros (%)	66.9%

Toggle details

Gender

Categorical

Distinct count	2
Unique (%)	0.0%
Missing (%)	0.0%
Missing (n)	0

Male	39949
Female	29764

Toggle details

ID

Categorical, Unique

First 3 values

APPC40121545438
APPQ30174685224
APPJ40284465127

Last 3 values

APPZ70610951333
APPB30147687132
APPP40371676716

Toggle details

Interest_Rate

Numeric

Distinct count	73
Unique (%)	0.3%
Missing (%)	68.0%
Missing (n)	47437
Infinite (%)	0.0%
Infinite (n)	0

Mean	19.214
Minimum	11.99
Maximum	37
Zeros (%)	0.0%

Toggle details

Lead_Creation_Date

Categorical

Distinct count	92
Unique (%)	0.1%
Missing (%)	0.0%
Missing (n)	0

02/09/16	1838
22/09/16	1629
29/09/16	1038
Other values (89)	65208

Toggle details

Loan_Amount

Numeric

Distinct count	197
Unique (%)	0.5%
Missing (%)	39.7%
Missing (n)	27709
Infinite (%)	0.0%
Infinite (n)	0

Mean	39430
Minimum	5000

Maximum	300000
Zeros (%)	0.0%

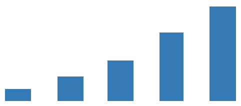
Toggle details

Loan_Period

Numeric

Distinct count	7
Unique (%)	0.0%
Missing (%)	39.7%
Missing (n)	27709
Infinite (%)	0.0%
Infinite (n)	0

Mean	3.8906
Minimum	1
Maximum	6
Zeros (%)	0.0%



Toggle details

Monthly_Income

Numeric

Distinct count	5010
Unique (%)	7.2%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0

Mean	5622.3
Minimum	0
Maximum	38384000
Zeros (%)	0.4%

Toggle details

Primary_Bank_Type
Categorical

Distinct count	3
Unique (%)	0.0%
Missing (%)	13.5%
Missing (n)	9391

P	39619
G	20703
(Missing)	9391

Toggle details

Source
Categorical

Distinct count	29
Unique (%)	0.0%
Missing (%)	0.0%
Missing (n)	0

S122	30941
S133	23877
S159	4474
Other values (26)	10421

Toggle details

Source_Category
Categorical

Distinct count	7
Unique (%)	0.0%
Missing (%)	0.0%
Missing (n)	0

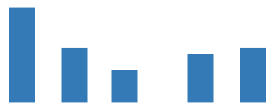
B	29812
G	26518
C	11374
Other values (4)	2009

Toggle details

Var1
Numeric

Distinct count	5
Unique (%)	0.0%

Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0
Mean	3.9484
Minimum	0
Maximum	10
Zeros (%)	33.4%



Toggle details

Sample

	ID	Gender	DOB	Lead_Creation_Date	City_Code
0	APPC90493171225	Female	23/07/79	15/07/16	C10001
1	APPD40611263344	Male	07/12/86	04/07/16	C10003
2	APPE70289249423	Male	10/12/82	19/07/16	C10125
3	APPF80273865537	Male	30/01/89	09/07/16	C10477
4	APPG60994436641	Male	19/04/85	20/07/16	C10002

In [6]:

```
pandas_profiling.ProfileReport(test)
```

Out[6]:

Overview

Dataset info

Number of variables	21
Number of observations	30037
Total Missing (%)	9.2%
Total size in memory	4.8 MiB
Average record size in memory	168.0 B

Variables types

Numeric	7
Categorical	12
Date	0
Text (Unique)	1
Rejected	1

Warnings

- `City_Category` has 314 / 1.0% missing values **Missing**
- `City_Code` has 314 / 1.0% missing values **Missing**
- `City_Code` has a high cardinality: 578 distinct values **Warning**
- `Customer_Existing_Primary_Bank_Code` has 4037 / 13.4% missing values **Missing**
- `Customer_Existing_Primary_Bank_Code` has a high cardinality: 57 distinct values **Warning**
- `DOB` has a high cardinality: 8511 distinct values **Warning**
- `EMI` is highly correlated with `Loan_Amount` (ρ = 0.93361) **Rejected**
- `Employer_Category1` has 1605 / 5.3% missing values **Missing**
- `Employer_Category2` has 1695 / 5.6% missing values **Missing**
- `Employer_Code` has 1605 / 5.3% missing values **Missing**
- `Employer_Code` has a high cardinality: 18656 distinct values **Warning**
- `Existing_EMI` has 20111 / 67.0% zeros
- `Interest_Rate` has 20385 / 67.9% missing values **Missing**
- `Lead_Creation_Date` has a high cardinality: 92 distinct values **Warning**
- `Loan_Amount` has 11871 / 39.5% missing values **Missing**
- `Loan_Period` has 11871 / 39.5% missing values **Missing**
- `Monthly_Income` is highly skewed (γ1 = 117.66)
- `Primary_Bank_Type` has 4037 / 13.4% missing values **Missing**
- `Var1` has 9937 / 33.1% zeros

Variables

City_Category
Categorical

Distinct count	4
Unique (%)	0.0%
Missing (%)	1.0%
Missing (n)	314

A	21498
C	5067
B	3158
(Missing)	314

Toggle details

City_Code
Categorical

Distinct count	578
Unique (%)	1.9%
Missing (%)	1.0%
Missing (n)	314

C10001	4306
C10002	3746
C10003	3684
Other values (574)	17987

Toggle details

Contacted
Categorical

Distinct count	2
Unique (%)	0.0%
Missing (%)	0.0%
Missing (n)	0

Y	19497
N	10540

Toggle details

Customer_Existing_Primary_Bank_Code
Categorical

Distinct count	57
Unique (%)	0.2%
Missing (%)	13.4%

Missing (n)	4037

B001	5958
B002	4665
B003	4167
Other values (53)	11210
(Missing)	4037

Toggle details

DOB
Categorical

Distinct count	8511
Unique (%)	28.3%
Missing (%)	0.0%
Missing (n)	3

11/01/82	92
04/03/71	84
03/03/91	51
Other values (8507)	29807

Toggle details

~~EMI~~
Highly correlated

This variable is highly correlated with `Loan_Amount` *and should be ignored for analysis*

Correlation 0.93361

Employer_Category1
Categorical

Distinct count	4
Unique (%)	0.0%
Missing (%)	5.3%
Missing (n)	1605

A	14469
B	7744
C	6219
(Missing)	1605

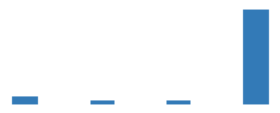
Toggle details

Employer_Category2

Numeric

Distinct count	5
Unique (%)	0.0%
Missing (%)	5.6%
Missing (n)	1695
Infinite (%)	0.0%
Infinite (n)	0

Mean	3.7282
Minimum	1
Maximum	4
Zeros (%)	0.0%



Toggle details

Employer_Code

Categorical

Distinct count	18656
Unique (%)	65.6%
Missing (%)	5.3%
Missing (n)	1605

COM0000002	165
COM0000004	130
COM0000003	124
Other values (18652)	28013
(Missing)	1605

Toggle details

Existing_EMI

Numeric

Distinct count	1805
Unique (%)	6.0%
Missing (%)	0.1%
Missing (n)	32
Infinite (%)	0.0%
Infinite (n)	0

Mean	348.91
Minimum	0
Maximum	43000
Zeros (%)	67.0%

Toggle details

Gender

Categorical

Distinct count	2
Unique (%)	0.0%
Missing (%)	0.0%
Missing (n)	0

Male	17204
Female	12833

Toggle details

ID

Categorical, Unique

First 3 values

APPS90177849642
APPF50232302911
APPN10004372935

Last 3 values

APPN90087131519
APPY10040885644
APPZ30412063529

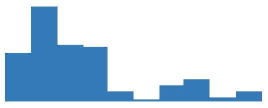
Toggle details

Interest_Rate

Numeric

Distinct count	71
Unique (%)	0.7%
Missing (%)	67.9%
Missing (n)	20385
Infinite (%)	0.0%
Infinite (n)	0

Mean	19.281
Minimum	11.99
Maximum	37
Zeros (%)	0.0%



Toggle details

Lead_Creation_Date
Categorical

Distinct count	92
Unique (%)	0.3%
Missing (%)	0.0%
Missing (n)	0

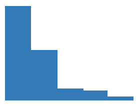
02/09/16	776
22/09/16	676
30/09/16	485
Other values (89)	28100

Toggle details

Loan_Amount
Numeric

Distinct count	169
Unique (%)	0.9%
Missing (%)	39.5%
Missing (n)	11871
Infinite (%)	0.0%
Infinite (n)	0

Mean	39483
Minimum	5000
Maximum	300000
Zeros (%)	0.0%



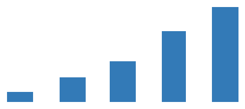
Toggle details

Loan_Period
Numeric

Distinct count	7
Unique (%)	0.0%
Missing (%)	39.5%
Missing (n)	11871
Infinite (%)	0.0%
Infinite (n)	0

Mean	3.9031
Minimum	1

Maximum	6
Zeros (%)	0.0%



Toggle details

Monthly_Income
Numeric

Distinct count	2825
Unique (%)	9.4%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0
Mean	3977.1
Minimum	0
Maximum	3500000
Zeros (%)	0.3%

Toggle details

Primary_Bank_Type
Categorical

Distinct count	3
Unique (%)	0.0%
Missing (%)	13.4%
Missing (n)	4037

P	16864
G	9136
(Missing)	4037

Toggle details

Source
Categorical

Distinct count	28
Unique (%)	0.1%
Missing (%)	0.0%
Missing (n)	0

S122	13272
S133	10313
S159	1949
Other values (25)	4503

Toggle details

Source_Category
Categorical

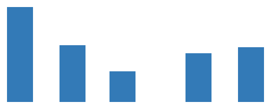
Distinct count	7
Unique (%)	0.0%
Missing (%)	0.0%
Missing (n)	0

B	12931
G	11385
C	4900
Other values (4)	821

Toggle details

Var1
Numeric

Distinct count	5
Unique (%)	0.0%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0
Mean	3.9623
Minimum	0
Maximum	10
Zeros (%)	33.1%



Toggle details

Sample

	ID	Gender	DOB	Lead_Creation_Date	City_Code
0	APPA70109647212	Male	03/06/88	05/07/16	C10028
1	APPB10687939341	Male	13/07/81	01/07/16	C10003
2	APPC80449411414	Female	19/11/90	01/07/16	C10009
3	APPD30665094501	Female	15/10/92	01/07/16	C10005
4	APPE80379821637	Male	21/09/88	01/07/16	C10005

Univariate Analysis

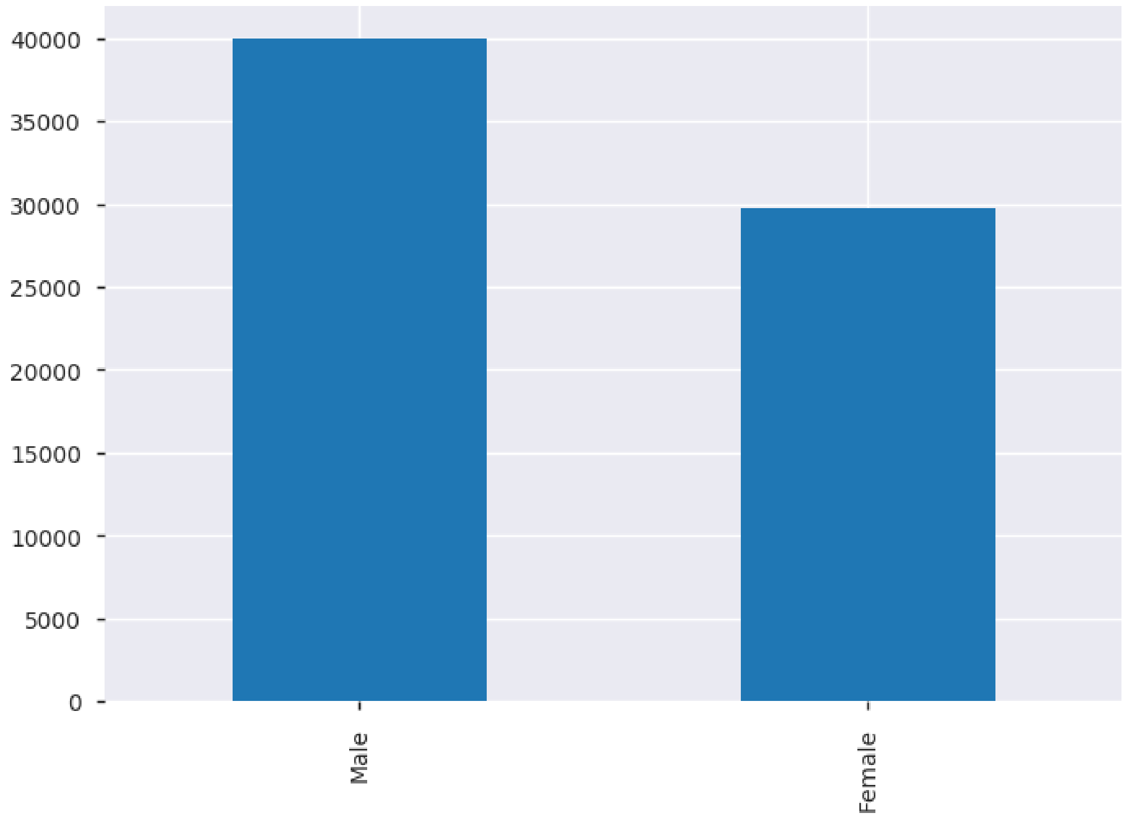
Bar charts of categorical data (Univariate Analysis)

In [18]:

```
train['Gender'].value_counts().head(100).plot.bar() #For top 100
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177d6ab0f0>
```

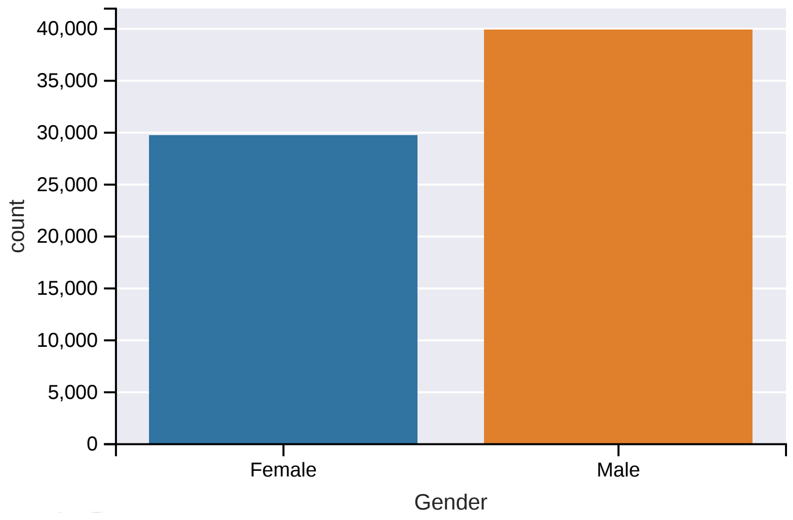


In [88]:

```
#pandas bar plot = seaborn count plot
sns.countplot(train['Gender'])
```

Out[88]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177a0ab198>
```

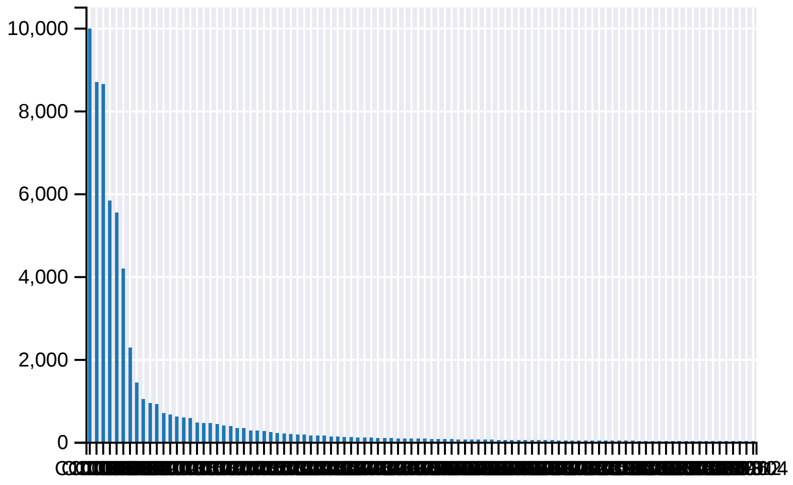


In [22]:

```
train['City_Code'].value_counts().head(100).plot.bar() #For top 100
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177d24f198>
```

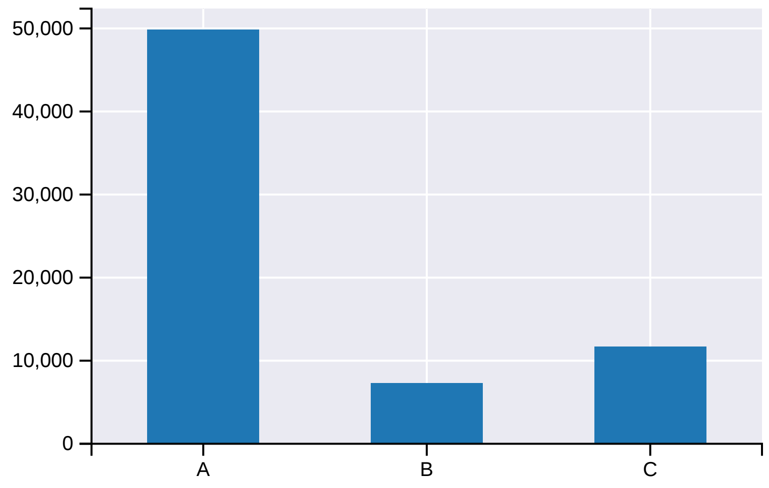


In [36]:

```
train['City_Category'].value_counts().sort_index().head(100).plot.bar() #For top
 100
```

Out[36]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177bd6fbe0>
```

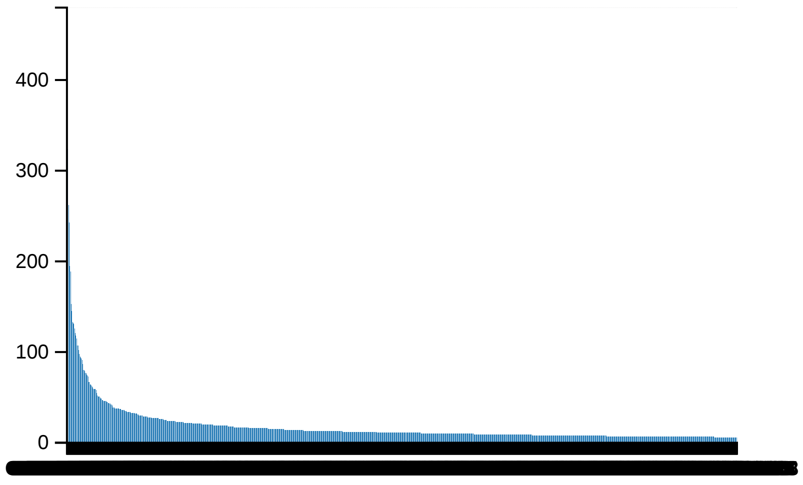


In [25]:

```
train['Employer_Code'].value_counts().head(1000).plot.bar() #For top 100
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177ce8ac50>
```

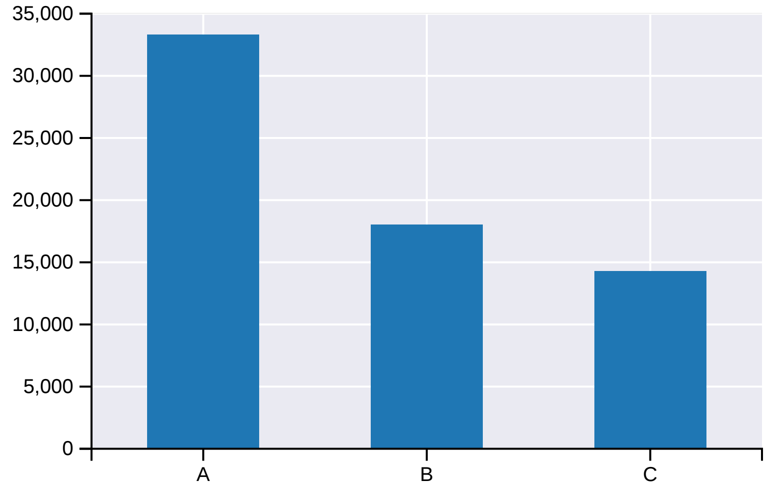


In [35]:

```
train['Employer_Category1'].value_counts().sort_index().head(1000).plot.bar() #F
or top 100
```

Out[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177bda1da0>
```

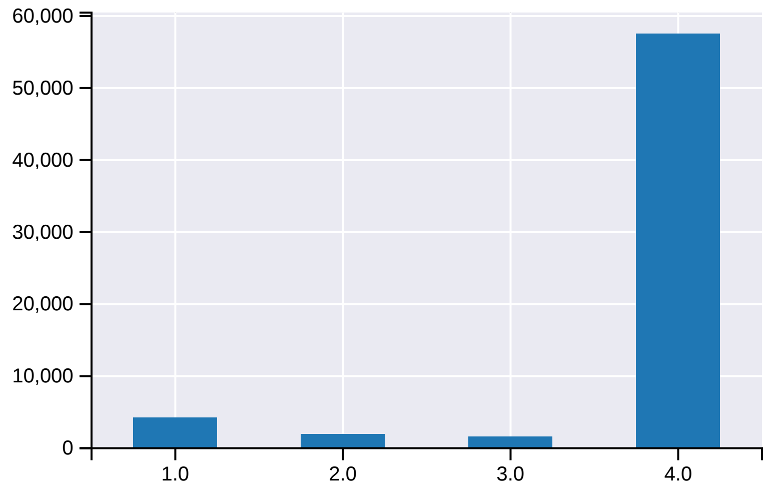


In [34]:

```
train['Employer_Category2'].value_counts().sort_index().head(1000).plot.bar() #F
or top 100
```

Out[34]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177bdeeba8>
```

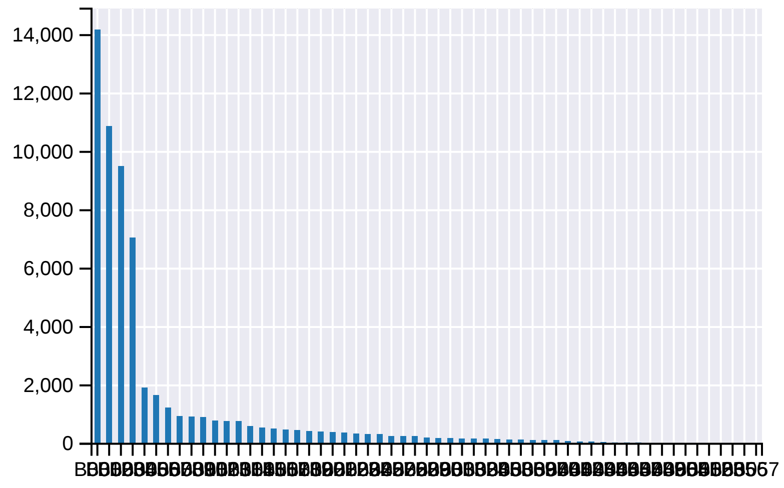


In [28]:

```
train['Customer_Existing_Primary_Bank_Code'].value_counts().head(1000).plot.bar
() #For top 100
```

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177c2c1a90>
```

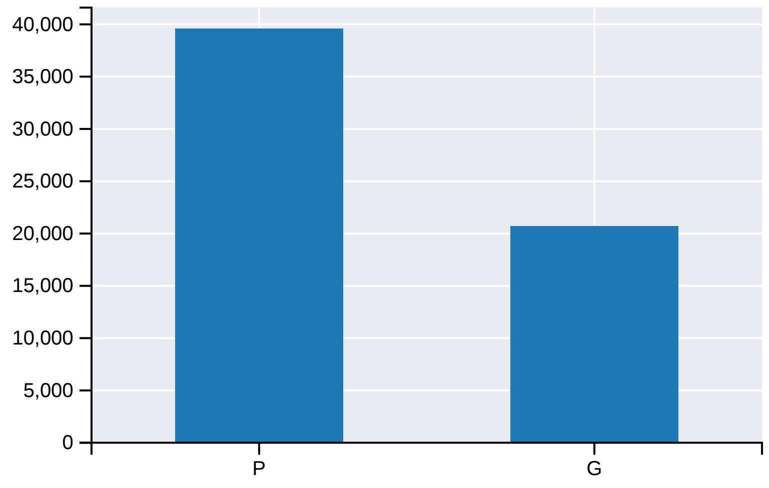


In [29]:

```
train['Primary_Bank_Type'].value_counts().head(1000).plot.bar() #For top 100
```

Out[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177bf46208>
```

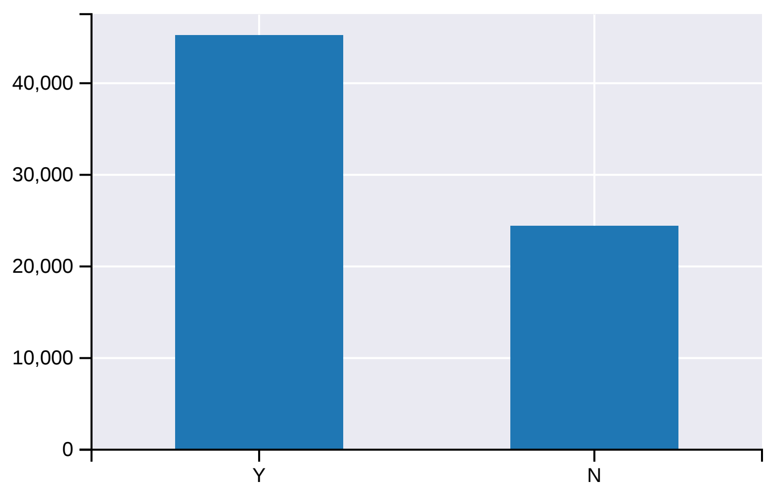


In [30]:

```
train['Contacted'].value_counts().head(1000).plot.bar() #For top 100
```

Out[30]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177bf820b8>
```

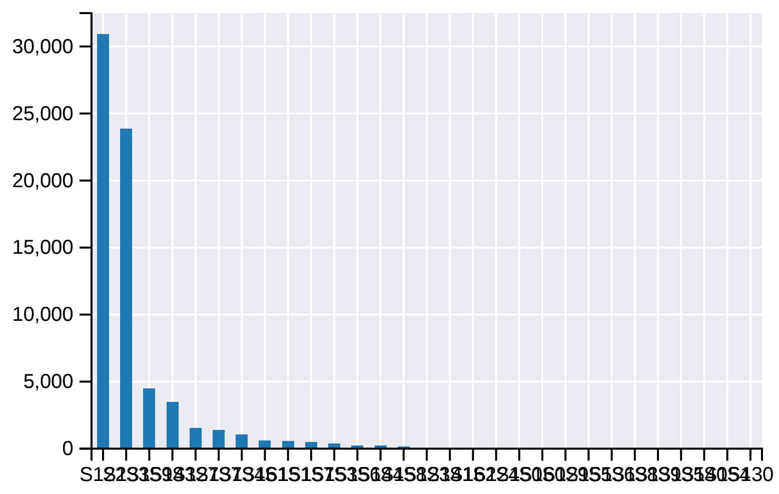


In [31]:

```
train['Source'].value_counts().head(1000).plot.bar() #For top 100
```

Out[31]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177bf3b358>
```

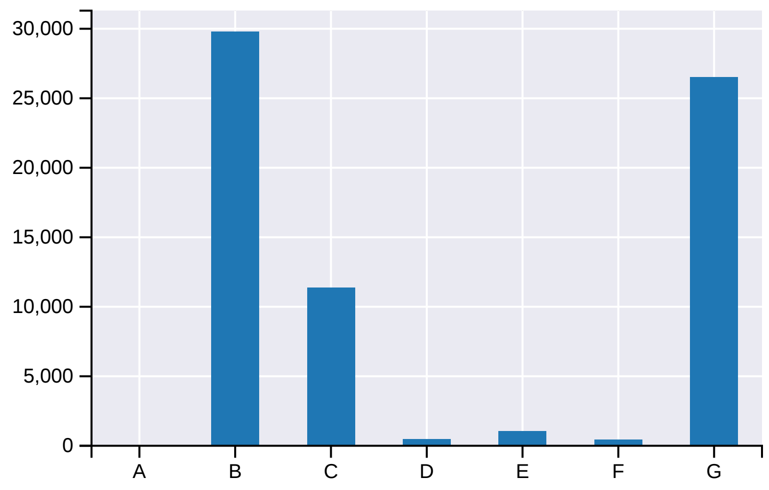


In [33]:

```
train['Source_Category'].value_counts().sort_index().head(1000).plot.bar() #For
 top 100
```

Out[33]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177be154a8>
```

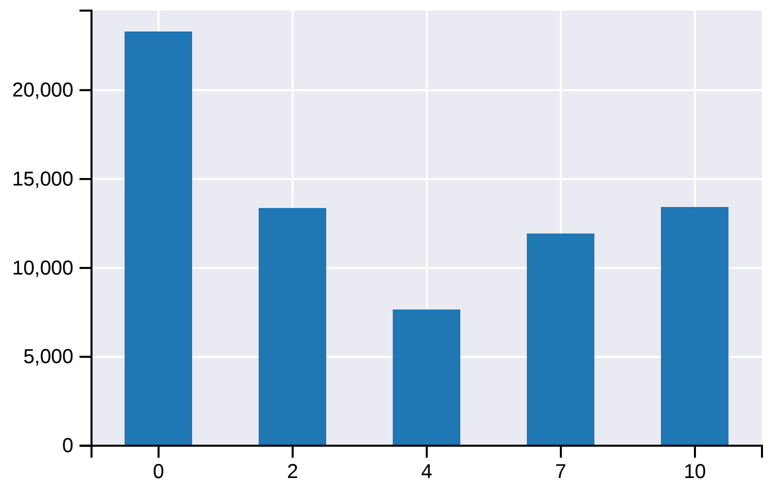


In [38]:

```
train['Var1'].value_counts().sort_index().head(1000).plot.bar()
```

Out[38]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177bce2240>
```



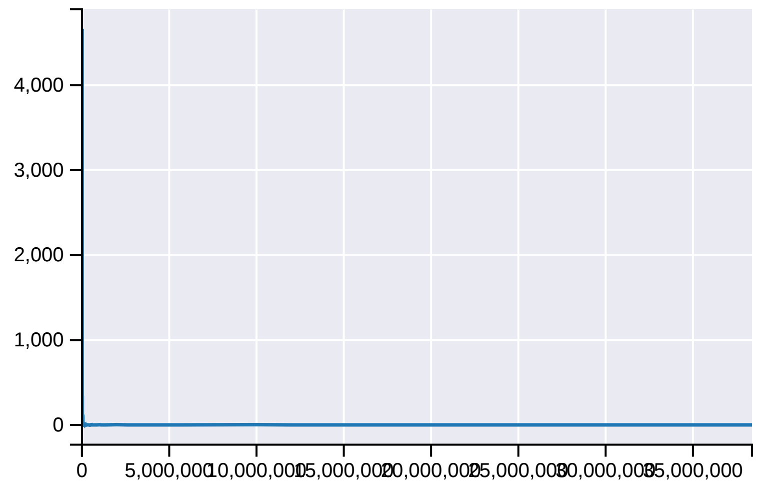
Line charts of numeric data (Univariate Analysis)

In [39]:

```
train['Monthly_Income'].value_counts().sort_index().plot.line()
```

Out[39]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177bcdd940>
```

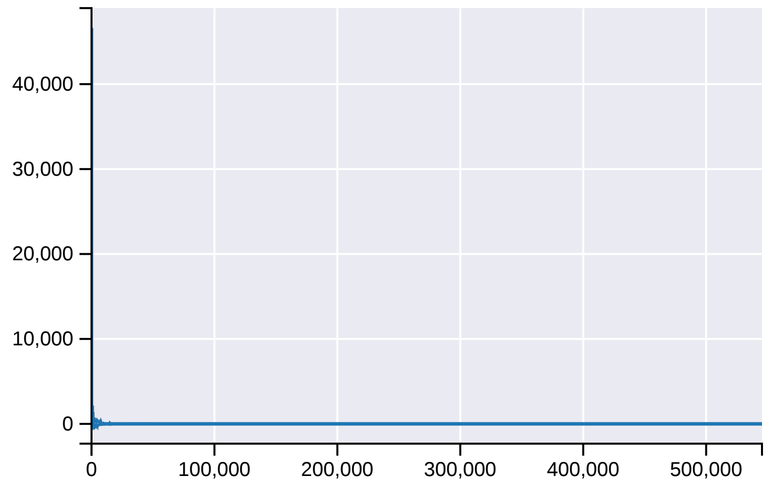


In [40]:

```
train['Existing_EMI'].value_counts().sort_index().plot.line()
```

Out[40]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177bac1f60>
```

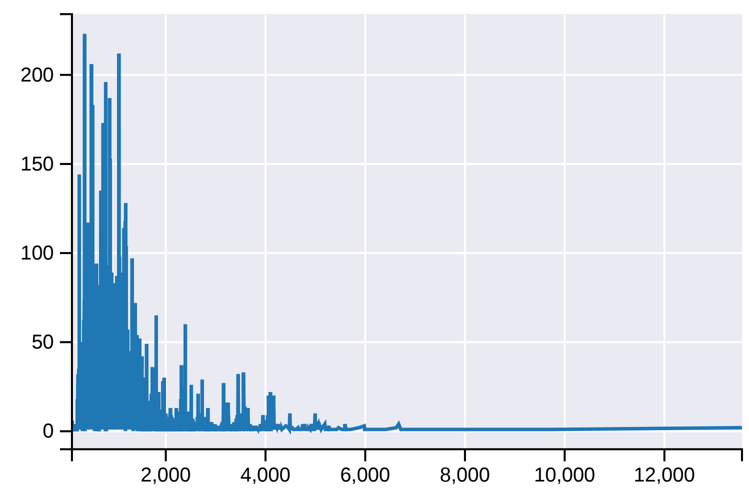


In [46]:

```
train['EMI'].value_counts().sort_index().plot.line()
```

Out[46]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177b9eb940>
```

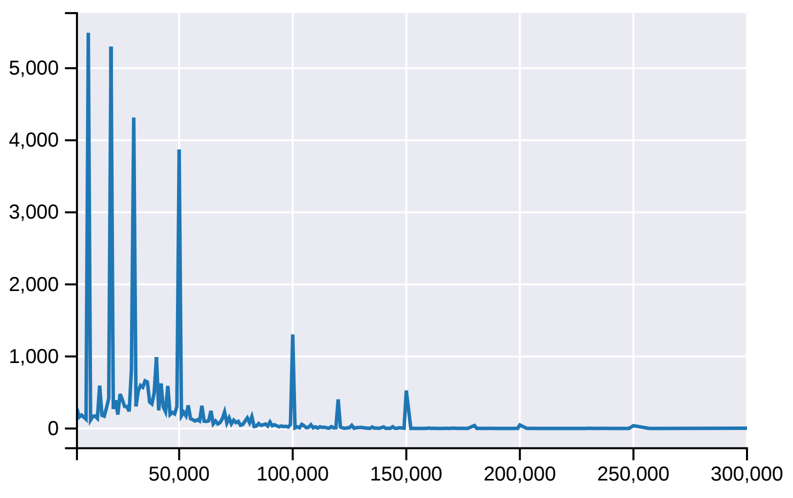


In [42]:

```
train['Loan_Amount'].value_counts().sort_index().plot.line()
```

Out[42]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177ba4dba8>
```

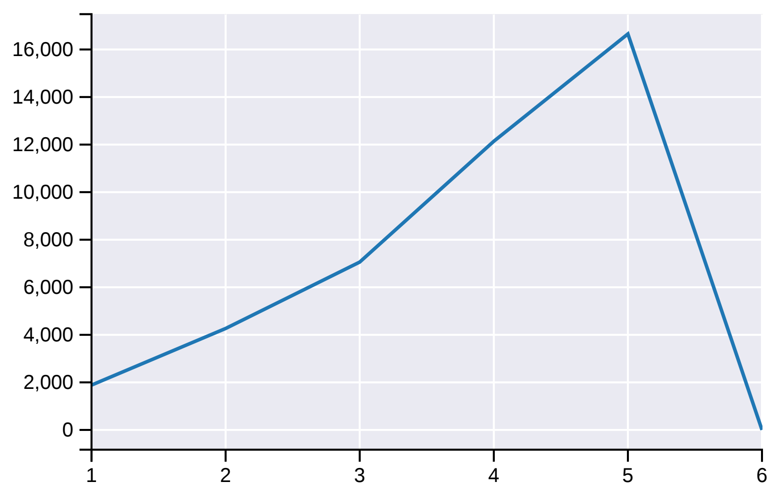


In [44]:

```
train['Loan_Period'].value_counts().sort_index().plot.line()
```

Out[44]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177ba5e898>
```

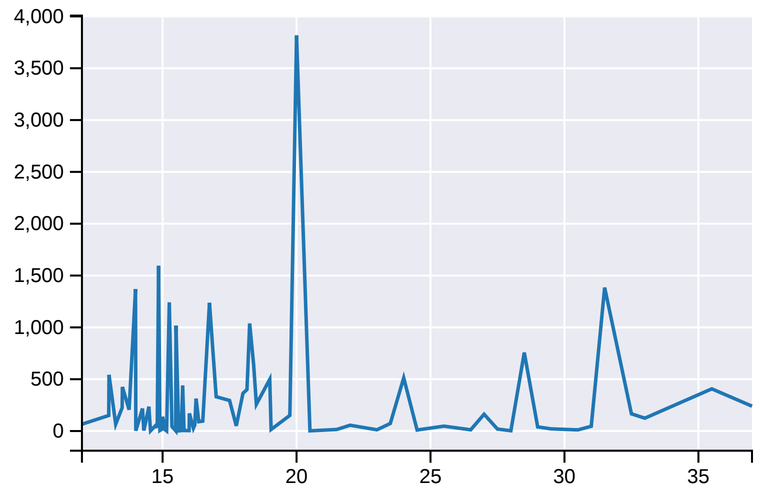


In [45]:

```
train['Interest_Rate'].value_counts().sort_index().plot.line()
```

Out[45]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177ba9b908>
```



In [ ]:

```
#Point To Remember : Area charts are line charts with bottom area shaded
#train['Interest_Rate'].value_counts().sort_index().plot.area()
```

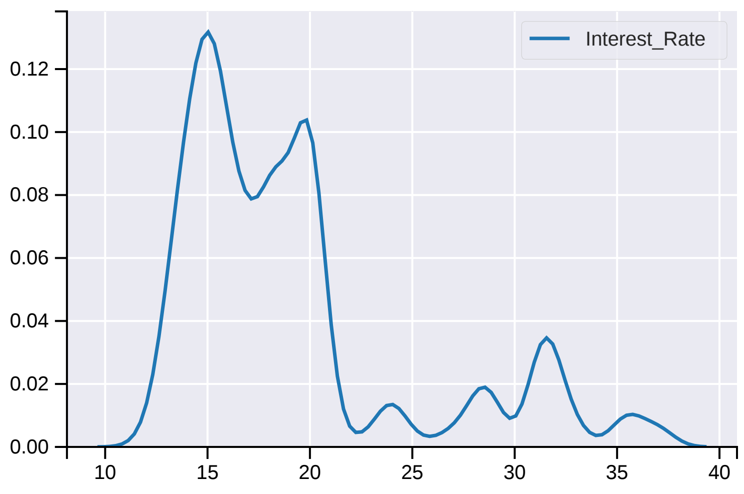
Kernel Density Estimate Plot (Univariate Analysis)

In [90]:

```
#Source: https://www.kaggle.com/residentmario/plotting-with-seaborn
'''
KDE, short for "kernel density estimate", is a statistical technique for smoothi
ng out data noise. It addresses an important fundamental weakness of a line char
t: it will buff out outlier or "in-betweener" values which would cause a line ch
art to suddenly dip.

For example, suppose that there was just one wine priced 19.93$, but several hun
dred prices 20.00$. If we were to plot the value counts in a line chart, our lin
e would dip very suddenly down to 1 and then back up to around 1000 again, creat
ing a strangely "jagged" line. The line chart with the same data, shown below fo
r the purposes of comparison, has exactly this problem!
'''
sns.kdeplot(train['Interest_Rate'].dropna())
```

Out[90]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177a001198>
```



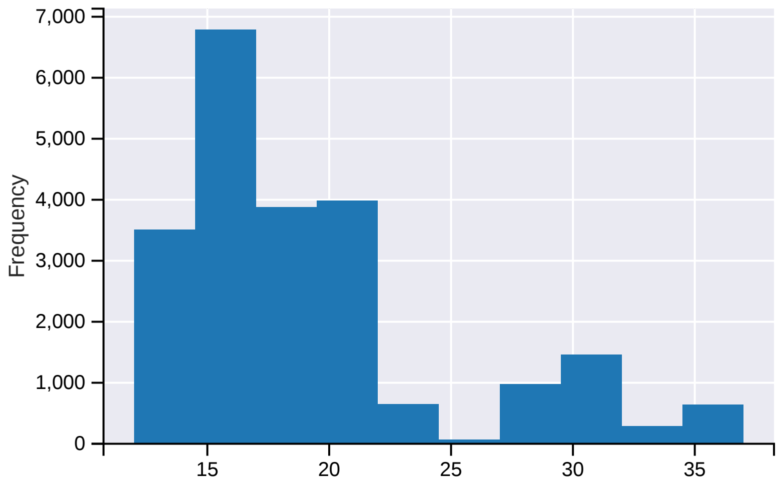
Histogram (Univariate Analysis) - For Numeric Data

In [51]:

```
#For Numeric
train['Interest_Rate'].plot.hist()
```

Out[51]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177b9b5208>
```

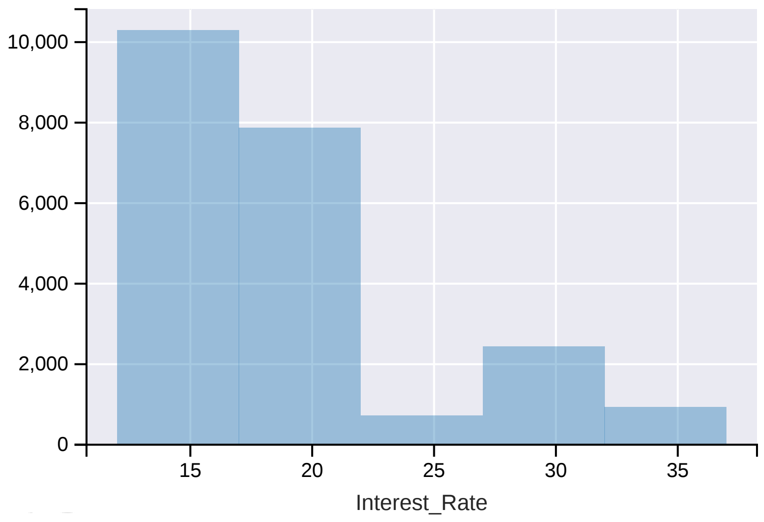


In [95]:

```
#"seaborn distplot" same as "pandas hist"
#bins = number of bins mention to make a clearer plot
#kde i.e smooting by default is True, make it False explicitly
sns.distplot(train['Interest_Rate'].dropna(), bins=5, kde=False)
```

Out[95]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177a18c978>
```



In [79]:

```
train['Interest_Rate'].value_counts().sort_index().plot.bar(figsize=(15, 10))
```

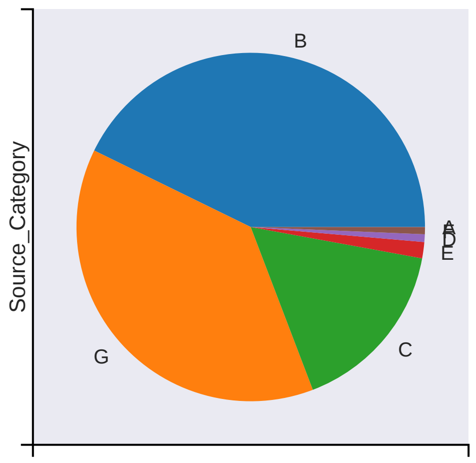
Out[79]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177a42ee10>
```

Pie Charts (Univariate Analysis)

In [55]:

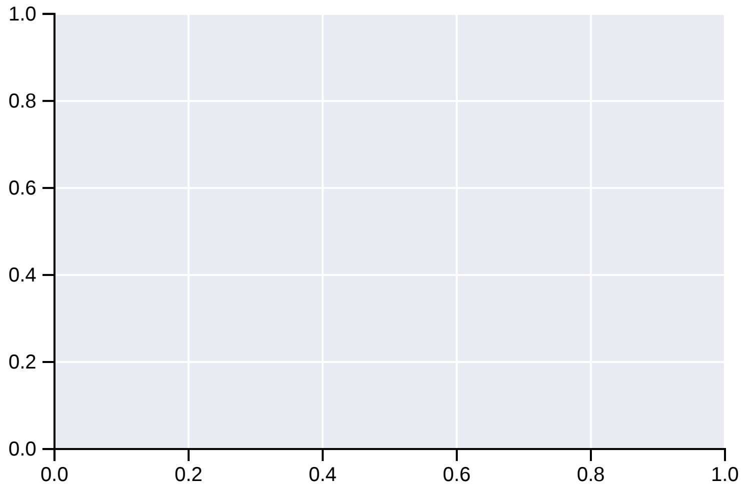
```
train['Source_Category'].value_counts().head(10).plot.pie()
plt.gca().set_aspect('equal')
```



In [102]:

```
sns.boxplot(x='Source_Category', y='Loan_Amount',data=train)
```

```
---------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-102-a222e2752bbc> in <module>()
----> 1 sns.boxplot(x='Source_Category',data=train)

/usr/local/lib/python3.5/site-packages/seaborn/categorical.py in box
plot(x, y, hue, data, order, hue_order, orient, color, palette, satu
ration, width, dodge, fliersize, linewidth, whis, notch, ax, **kwarg
s)
   2215     kwargs.update(dict(whis=whis, notch=notch))
   2216
-> 2217     plotter.plot(ax, kwargs)
   2218     return ax
   2219

/usr/local/lib/python3.5/site-packages/seaborn/categorical.py in plo
t(self, ax, boxplot_kws)
    542     def plot(self, ax, boxplot_kws):
    543         """Make the plot."""
--> 544         self.draw_boxplot(ax, boxplot_kws)
    545         self.annotate_axes(ax)
    546         if self.orient == "h":

/usr/local/lib/python3.5/site-packages/seaborn/categorical.py in dra
w_boxplot(self, ax, kws)
    479                                          positions=[i],
    480                                          widths=self.width,
--> 481                                          **kws)
    482                 color = self.colors[i]
    483                 self.restyle_boxplot(artist_dict, color, pro
ps)

/usr/local/lib/python3.5/site-packages/matplotlib/__init__.py in inn
er(ax, *args, **kwargs)
   1708                     warnings.warn(msg % (label_namer, func._
_name__),
   1709                                   RuntimeWarning, stacklevel
=2)
-> 1710             return func(ax, *args, **kwargs)
   1711         pre_doc = inner.__doc__
   1712         if pre_doc is None:

/usr/local/lib/python3.5/site-packages/matplotlib/axes/_axes.py in b
oxplot(self, x, notch, sym, vert, whis, positions, widths, patch_art
ist, bootstrap, usermedians, conf_intervals, meanline, showmeans, sh
owcaps, showbox, showfliers, boxprops, labels, flierprops, medianpro
ps, meanprops, capprops, whiskerprops, manage_xticks, autorange, zor
der)
   3330
   3331         bxpstats = cbook.boxplot_stats(x, whis=whis, bootstr
ap=bootstrap,
-> 3332                                        labels=labels, autora
nge=autorange)
   3333         if notch is None:
   3334             notch = rcParams['boxplot.notch']

/usr/local/lib/python3.5/site-packages/matplotlib/cbook/__init__.py
 in boxplot_stats(X, whis, bootstrap, labels, autorange)
   1823
   1824         # arithmetic mean
-> 1825         stats['mean'] = np.mean(x)
   1826
   1827         # medians and quartiles

/usr/local/lib/python3.5/site-packages/numpy/core/fromnumeric.py in
mean(a, axis, dtype, out, keepdims)
   2887
   2888     return _methods._mean(a, axis=axis, dtype=dtype,
-> 2889                           out=out, **kwargs)
   2890
   2891

/usr/local/lib/python3.5/site-packages/numpy/core/_methods.py in _me
an(a, axis, dtype, out, keepdims)
     68             is_float16_result = True
     69
---> 70     ret = umr_sum(arr, axis, dtype, out, keepdims)
     71     if isinstance(ret, mu.ndarray):
     72         ret = um.true_divide(

TypeError: cannot perform reduce with flexible type
```



Bivariate Analysis

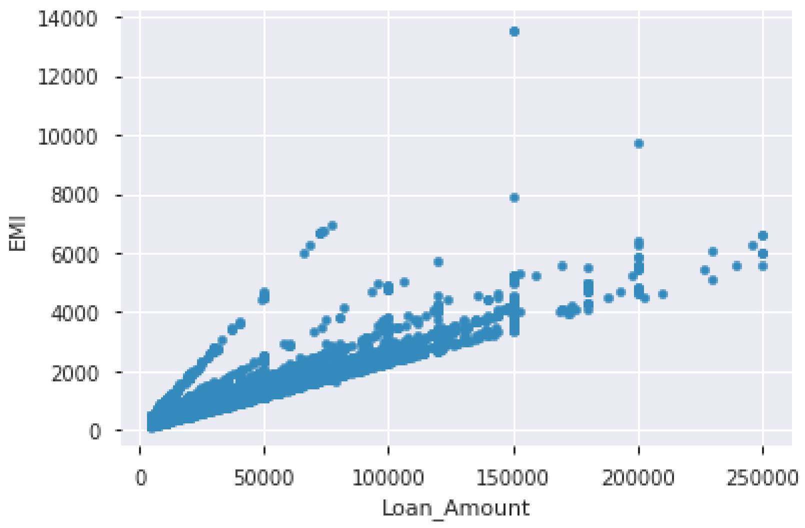
Scatter plot (Bivariate Analysis)

In [62]:

```
train.plot.scatter(x='Loan_Amount', y='EMI')
```

```
Out[62]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f177b2b19b0>
```

```
--------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
/usr/local/lib/python3.5/site-packages/IPython/core/formatters.py in
 __call__(self, obj)
    339                 pass
    340             else:
--> 341                 return printer(obj)
    342             # Finally look for special method names
    343             method = get_real_method(obj, self.print_method)

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in <lambda>
(fig, kwds)
    408     formatter = ip.display_formatter.formatters['text/html']
    409     formatter.for_type(Figure,
--> 410                        lambda fig, kwds=kwargs: fig_to_html
(fig, **kwds))
    411 
    412 

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in fig_to_h
tml(fig, d3_url, mpld3_url, no_extras, template_type, figid, use_htt
p, **kwargs)
    249                            d3_url=d3_url,
    250                            mpld3_url=mpld3_url,
--> 251                            figure_json=json.dumps(figure_jso
n, cls=NumpyEncoder),
    252                            extra_css=extra_css,
    253                            extra_js=extra_js)

/usr/local/lib/python3.5/json/__init__.py in dumps(obj, skipkeys, en
sure_ascii, check_circular, allow_nan, cls, indent, separators, defa
ult, sort_keys, **kw)
    235         check_circular=check_circular, allow_nan=allow_nan,
 indent=indent,
    236         separators=separators, default=default, sort_keys=so
rt_keys,
--> 237         **kw).encode(obj)
    238 
    239 

/usr/local/lib/python3.5/json/encoder.py in encode(self, o)
    196         # exceptions aren't as detailed.  The list call shou
ld be roughly
    197         # equivalent to the PySequence_Fast that ''.join() w
ould do.
--> 198         chunks = self.iterencode(o, _one_shot=True)
    199         if not isinstance(chunks, (list, tuple)):
    200             chunks = list(chunks)

/usr/local/lib/python3.5/json/encoder.py in iterencode(self, o, _one
_shot)
    254                 self.key_separator, self.item_separator, sel
f.sort_keys,
    255                 self.skipkeys, _one_shot)
--> 256         return _iterencode(o, 0)
    257 
    258 def _make_iterencode(markers, _default, _encoder, _indent, _
floatstr,
/usr/local/lib/python3.5/site-packages/mpld3/_display.py in default
(self, obj)
    136             numpy.float64)):
    137             return float(obj)
--> 138         return json.JSONEncoder.default(self, obj)
    139 
    140 

/usr/local/lib/python3.5/json/encoder.py in default(self, o)
    177 
    178         """
--> 179         raise TypeError(repr(o) + " is not JSON serializabl
e")
    180 
    181     def encode(self, o):

TypeError: array([ 0.3]) is not JSON serializable
```



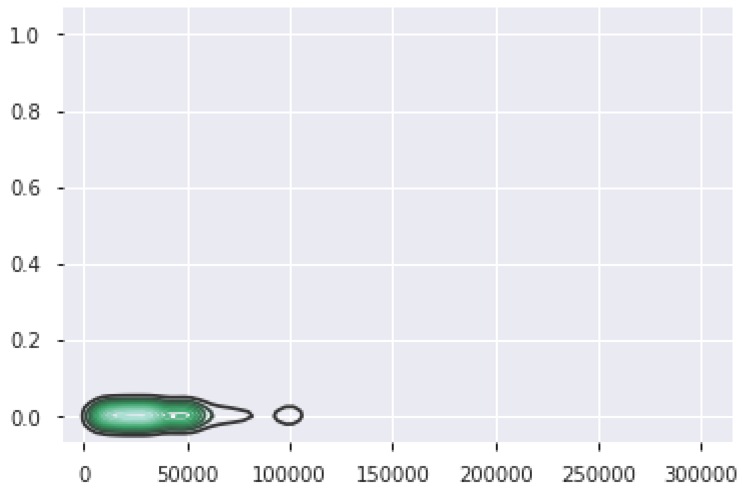
Kernel Density Estimate plot (Bivariate Analysis)

In [92]:

```
sns.kdeplot(train[['Loan_Amount','Approved']].dropna())
```

```
Out[92]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f1779f0a208>
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/usr/local/lib/python3.5/site-packages/IPython/core/formatters.py in __call__(self, obj)
    339                 pass
    340             else:
--> 341                 return printer(obj)
    342             # Finally look for special method names
    343             method = get_real_method(obj, self.print_method)

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in <lambda>(fig, kwds)
    408     formatter = ip.display_formatter.formatters['text/html']
    409     formatter.for_type(Figure,
--> 410                        lambda fig, kwds=kwargs: fig_to_html(fig, **kwds))
    411 
    412 

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in fig_to_html(fig, d3_url, mpld3_url, no_extras, template_type, figid, use_http, **kwargs)
    249                            d3_url=d3_url,
    250                            mpld3_url=mpld3_url,
--> 251                            figure_json=json.dumps(figure_json, cls=NumpyEncoder),
    252                            extra_css=extra_css,
    253                            extra_js=extra_js)

/usr/local/lib/python3.5/json/__init__.py in dumps(obj, skipkeys, ensure_ascii, check_circular, allow_nan, cls, indent, separators, default, sort_keys, **kw)
    235         check_circular=check_circular, allow_nan=allow_nan, indent=indent,
    236         separators=separators, default=default, sort_keys=sort_keys,
--> 237         **kw).encode(obj)
    238 
    239 

/usr/local/lib/python3.5/json/encoder.py in encode(self, o)
    196         # exceptions aren't as detailed.  The list call should be roughly
    197         # equivalent to the PySequence_Fast that ''.join() would do.
--> 198         chunks = self.iterencode(o, _one_shot=True)
    199         if not isinstance(chunks, (list, tuple)):
    200             chunks = list(chunks)

/usr/local/lib/python3.5/json/encoder.py in iterencode(self, o, _one_shot)
    254                 self.key_separator, self.item_separator, self.sort_keys,
    255                 self.skipkeys, _one_shot)
--> 256         return _iterencode(o, 0)
    257 
    258 def _make_iterencode(markers, _default, _encoder, _indent, _floatstr,
/usr/local/lib/python3.5/site-packages/mpld3/_display.py in default
(self, obj)
    136             numpy.float64)):
    137             return float(obj)
--> 138         return json.JSONEncoder.default(self, obj)
    139 
    140 

/usr/local/lib/python3.5/json/encoder.py in default(self, o)
    177 
    178         """
--> 179         raise TypeError(repr(o) + " is not JSON serializabl
e")
    180 
    181     def encode(self, o):

TypeError: array([ 1.75]) is not JSON serializable
```



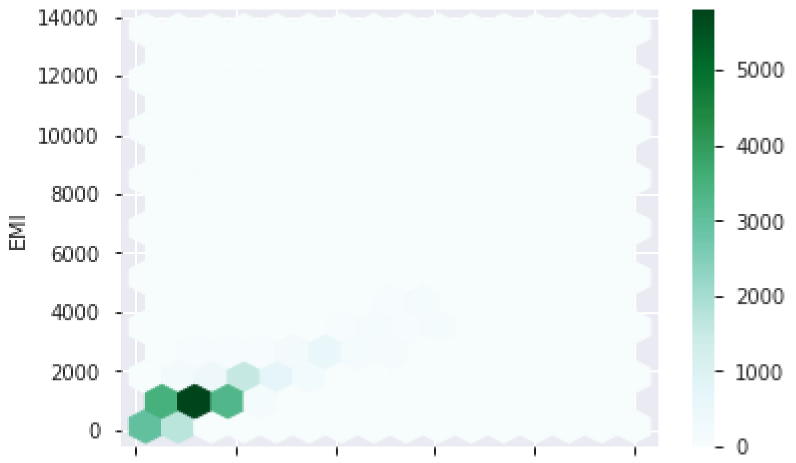
Hexplot (A hexplot aggregates points in space into hexagons, and then colorize those hexagons) (Bivariate Analysis)

In [63]:

```
train.plot.hexbin(x='Loan_Amount', y='EMI', gridsize=15)
```

```
Out[63]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f177b062f60>
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent cal
l last)
/usr/local/lib/python3.5/site-packages/IPython/core/formatters.py in
 __call__(self, obj)
    339                 pass
    340             else:
--> 341                 return printer(obj)
    342             # Finally look for special method names
    343             method = get_real_method(obj, self.print_method)

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in <lambda>
(fig, kwds)
    408     formatter = ip.display_formatter.formatters['text/html']
    409     formatter.for_type(Figure,
--> 410                        lambda fig, kwds=kwargs: fig_to_html
(fig, **kwds))
    411 
    412 

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in fig_to_h
tml(fig, d3_url, mpld3_url, no_extras, template_type, figid, use_htt
p, **kwargs)
    249                              d3_url=d3_url,
    250                              mpld3_url=mpld3_url,
--> 251                              figure_json=json.dumps(figure_jso
n, cls=NumpyEncoder),
    252                              extra_css=extra_css,
    253                              extra_js=extra_js)

/usr/local/lib/python3.5/json/__init__.py in dumps(obj, skipkeys, en
sure_ascii, check_circular, allow_nan, cls, indent, separators, defa
ult, sort_keys, **kw)
    235         check_circular=check_circular, allow_nan=allow_nan,
 indent=indent,
    236         separators=separators, default=default, sort_keys=so
rt_keys,
--> 237         **kw).encode(obj)
    238 
    239 

/usr/local/lib/python3.5/json/encoder.py in encode(self, o)
    196         # exceptions aren't as detailed.  The list call shou
ld be roughly
    197         # equivalent to the PySequence_Fast that ''.join() w
ould do.
--> 198         chunks = self.iterencode(o, _one_shot=True)
    199         if not isinstance(chunks, (list, tuple)):
    200             chunks = list(chunks)

/usr/local/lib/python3.5/json/encoder.py in iterencode(self, o, _one
_shot)
    254                 self.key_separator, self.item_separator, sel
f.sort_keys,
    255                 self.skipkeys, _one_shot)
--> 256         return _iterencode(o, 0)
    257 
    258 def _make_iterencode(markers, _default, _encoder, _indent, _
floatstr,
/usr/local/lib/python3.5/site-packages/mpld3/_display.py in default
(self, obj)
    136             numpy.float64)):
    137             return float(obj)
--> 138         return json.JSONEncoder.default(self, obj)
    139 
    140 

/usr/local/lib/python3.5/json/encoder.py in default(self, o)
    177 
    178         """
--> 179         raise TypeError(repr(o) + " is not JSON serializabl
e")
    180 
    181     def encode(self, o):

TypeError: array([ 1.]) is not JSON serializable
```



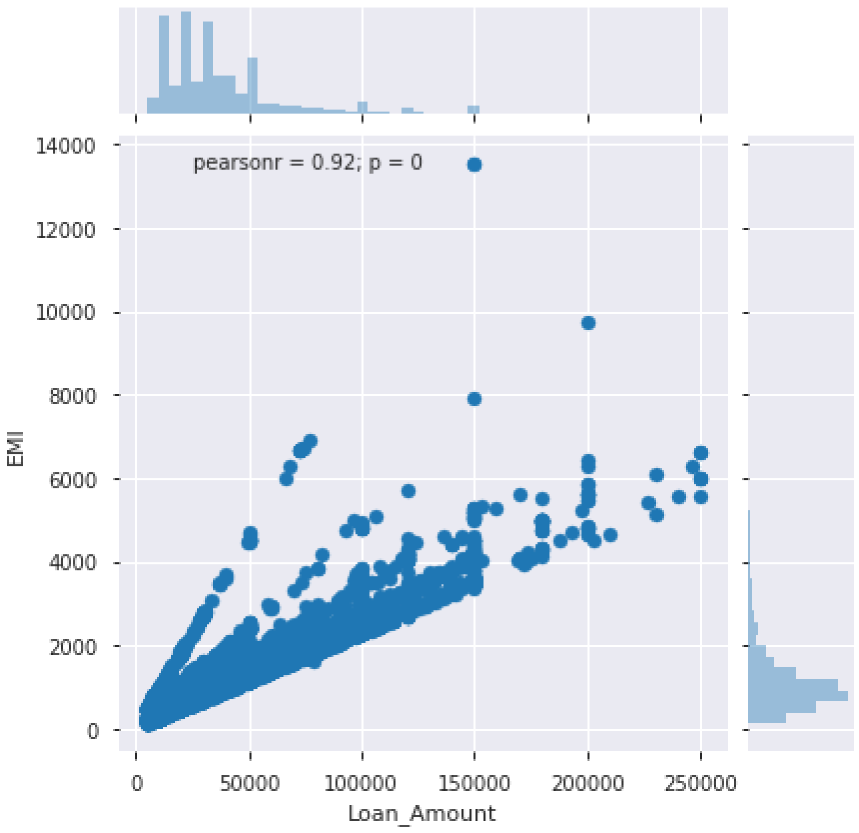
JointPlot - combine scatter and hexplot (Bivariate Analysis)

In [97]:

```
sns.jointplot(x='Loan_Amount', y='EMI', data=train[['Loan_Amount', 'EMI']].dropn
a())
```

```
Out[97]:
<seaborn.axisgrid.JointGrid at 0x7f177a1b64e0>
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent cal
l last)
/usr/local/lib/python3.5/site-packages/IPython/core/formatters.py in
 __call__(self, obj)
    339                 pass
    340             else:
--> 341                 return printer(obj)
    342             # Finally look for special method names
    343             method = get_real_method(obj, self.print_method)

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in <lambda>
(fig, kwds)
    408     formatter = ip.display_formatter.formatters['text/html']
    409     formatter.for_type(Figure,
--> 410                        lambda fig, kwds=kwargs: fig_to_html
(fig, **kwds))
    411 
    412 

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in fig_to_h
tml(fig, d3_url, mpld3_url, no_extras, template_type, figid, use_htt
p, **kwargs)
    249                            d3_url=d3_url,
    250                            mpld3_url=mpld3_url,
--> 251                            figure_json=json.dumps(figure_jso
n, cls=NumpyEncoder),
    252                            extra_css=extra_css,
    253                            extra_js=extra_js)

/usr/local/lib/python3.5/json/__init__.py in dumps(obj, skipkeys, en
sure_ascii, check_circular, allow_nan, cls, indent, separators, defa
ult, sort_keys, **kw)
    235         check_circular=check_circular, allow_nan=allow_nan,
 indent=indent,
    236         separators=separators, default=default, sort_keys=so
rt_keys,
--> 237         **kw).encode(obj)
    238 
    239 

/usr/local/lib/python3.5/json/encoder.py in encode(self, o)
    196         # exceptions aren't as detailed.  The list call shou
ld be roughly
    197         # equivalent to the PySequence_Fast that ''.join() w
ould do.
--> 198         chunks = self.iterencode(o, _one_shot=True)
    199         if not isinstance(chunks, (list, tuple)):
    200             chunks = list(chunks)

/usr/local/lib/python3.5/json/encoder.py in iterencode(self, o, _one
_shot)
    254                 self.key_separator, self.item_separator, sel
f.sort_keys,
    255                 self.skipkeys, _one_shot)
--> 256         return _iterencode(o, 0)
    257 
    258 def _make_iterencode(markers, _default, _encoder, _indent, _
floatstr,
/usr/local/lib/python3.5/site-packages/mpld3/_display.py in default
(self, obj)
    136             numpy.float64)):
    137             return float(obj)
--> 138         return json.JSONEncoder.default(self, obj)
    139
    140

/usr/local/lib/python3.5/json/encoder.py in default(self, o)
    177
    178         """
--> 179         raise TypeError(repr(o) + " is not JSON serializabl
e")
    180
    181     def encode(self, o):

TypeError: array([ 0.3]) is not JSON serializable
```

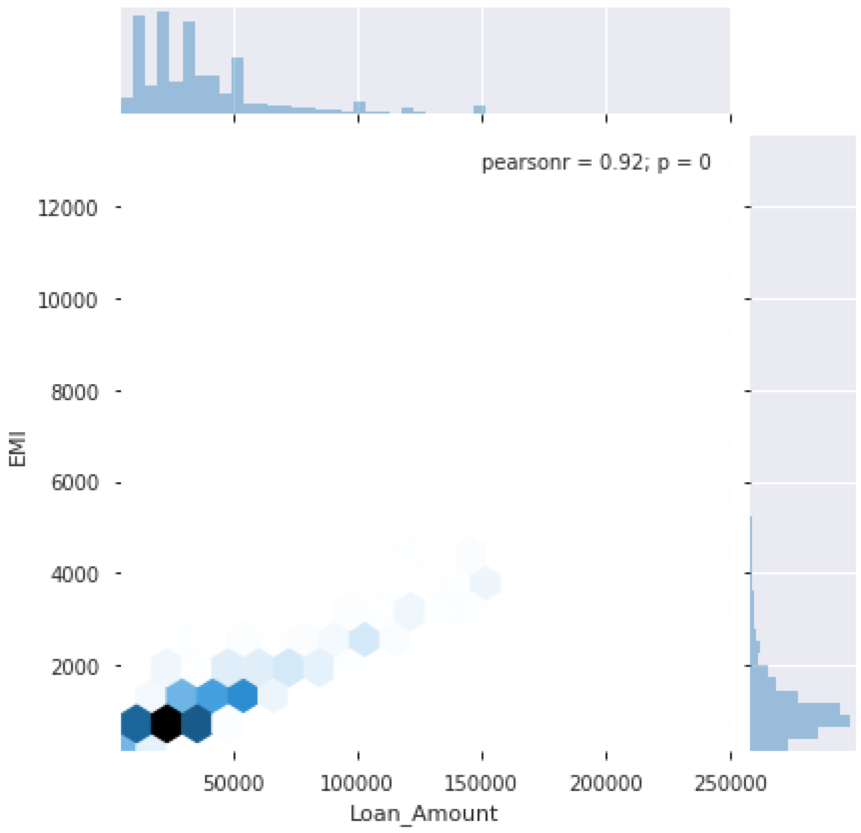


In [99]:

```
sns.jointplot(x='Loan_Amount', y='EMI', data=train[['Loan_Amount', 'EMI']].dropn
a(), kind='hex', gridsize=20)
```

```
Out[99]:

<seaborn.axisgrid.JointGrid at 0x7f177c503a58>
```

```
--------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
/usr/local/lib/python3.5/site-packages/IPython/core/formatters.py in
 __call__(self, obj)
    339                 pass
    340             else:
--> 341                 return printer(obj)
    342             # Finally look for special method names
    343             method = get_real_method(obj, self.print_method)

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in <lambda>
(fig, kwds)
    408     formatter = ip.display_formatter.formatters['text/html']
    409     formatter.for_type(Figure,
--> 410                        lambda fig, kwds=kwargs: fig_to_html
(fig, **kwds))
    411
    412

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in fig_to_h
tml(fig, d3_url, mpld3_url, no_extras, template_type, figid, use_htt
p, **kwargs)
    249                            d3_url=d3_url,
    250                            mpld3_url=mpld3_url,
--> 251                            figure_json=json.dumps(figure_jso
n, cls=NumpyEncoder),
    252                            extra_css=extra_css,
    253                            extra_js=extra_js)

/usr/local/lib/python3.5/json/__init__.py in dumps(obj, skipkeys, en
sure_ascii, check_circular, allow_nan, cls, indent, separators, defa
ult, sort_keys, **kw)
    235         check_circular=check_circular, allow_nan=allow_nan,
 indent=indent,
    236         separators=separators, default=default, sort_keys=so
rt_keys,
--> 237         **kw).encode(obj)
    238
    239

/usr/local/lib/python3.5/json/encoder.py in encode(self, o)
    196         # exceptions aren't as detailed.  The list call shou
ld be roughly
    197         # equivalent to the PySequence_Fast that ''.join() w
ould do.
--> 198         chunks = self.iterencode(o, _one_shot=True)
    199         if not isinstance(chunks, (list, tuple)):
    200             chunks = list(chunks)

/usr/local/lib/python3.5/json/encoder.py in iterencode(self, o, _one
_shot)
    254                 self.key_separator, self.item_separator, sel
f.sort_keys,
    255                 self.skipkeys, _one_shot)
--> 256         return _iterencode(o, 0)
    257
    258 def _make_iterencode(markers, _default, _encoder, _indent, _
floatstr,
/usr/local/lib/python3.5/site-packages/mpld3/_display.py in default
(self, obj)
    136             numpy.float64)):
    137             return float(obj)
--> 138         return json.JSONEncoder.default(self, obj)
    139 
    140 

/usr/local/lib/python3.5/json/encoder.py in default(self, o)
    177 
    178         """
--> 179         raise TypeError(repr(o) + " is not JSON serializabl
e")
    180 
    181     def encode(self, o):

TypeError: array([ 1.]) is not JSON serializable
```



Stacked Plots (Bivariate Analysis)

In [65]:

```
 train_stats_as_per_source_category = train.groupby('Source_Category').mean()[[
'Loan_Amount', 'Existing_EMI', 'EMI']]
```

```
In [68]:
```

```
train_stats_as_per_source_category.head()
```

```
Out[68]:
```

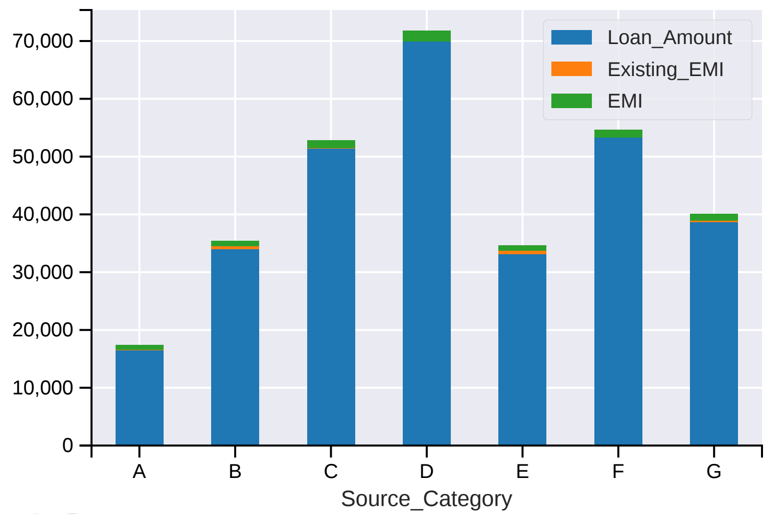
	Loan_Amount	Existing_EMI	EMI
Source_Category			
A	16500.000000	70.000000	848.000000
B	33986.532361	540.207652	927.607344
C	51448.958453	0.877440	1363.924179
D	69926.229508	0.000000	1851.180328
E	33110.248447	646.162381	909.305136

```
In [69]:
```

```
train_stats_as_per_source_category.plot.bar(stacked=True)
#Link https://www.kaggle.com/residentmario/bivariate-plotting-with-pandas/
```

```
Out[69]:
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177ac738d0>
```

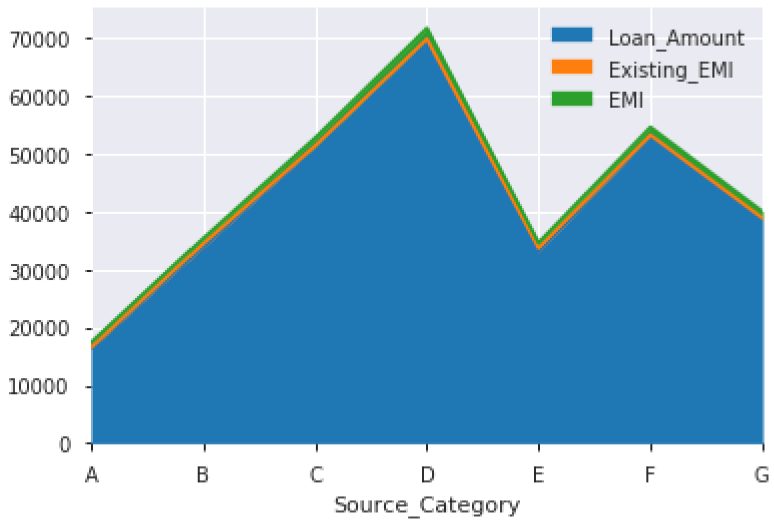


In [70]:

```
train_stats_as_per_source_category.plot.area()
```

```
Out[70]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f177ac07278>
```

```
--------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
/usr/local/lib/python3.5/site-packages/IPython/core/formatters.py in
 __call__(self, obj)
    339                 pass
    340             else:
--> 341                 return printer(obj)
    342             # Finally look for special method names
    343             method = get_real_method(obj, self.print_method)

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in <lambda>
(fig, kwds)
    408     formatter = ip.display_formatter.formatters['text/html']
    409     formatter.for_type(Figure,
--> 410                        lambda fig, kwds=kwargs: fig_to_html
(fig, **kwds))
    411 
    412 

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in fig_to_h
tml(fig, d3_url, mpld3_url, no_extras, template_type, figid, use_htt
p, **kwargs)
    249                            d3_url=d3_url,
    250                            mpld3_url=mpld3_url,
--> 251                            figure_json=json.dumps(figure_jso
n, cls=NumpyEncoder),
    252                            extra_css=extra_css,
    253                            extra_js=extra_js)

/usr/local/lib/python3.5/json/__init__.py in dumps(obj, skipkeys, en
sure_ascii, check_circular, allow_nan, cls, indent, separators, defa
ult, sort_keys, **kw)
    235         check_circular=check_circular, allow_nan=allow_nan,
 indent=indent,
    236         separators=separators, default=default, sort_keys=so
rt_keys,
--> 237         **kw).encode(obj)
    238 
    239 

/usr/local/lib/python3.5/json/encoder.py in encode(self, o)
    196         # exceptions aren't as detailed.  The list call shou
ld be roughly
    197         # equivalent to the PySequence_Fast that ''.join() w
ould do.
--> 198         chunks = self.iterencode(o, _one_shot=True)
    199         if not isinstance(chunks, (list, tuple)):
    200             chunks = list(chunks)

/usr/local/lib/python3.5/json/encoder.py in iterencode(self, o, _one
_shot)
    254                 self.key_separator, self.item_separator, sel
f.sort_keys,
    255                 self.skipkeys, _one_shot)
--> 256         return _iterencode(o, 0)
    257 
    258 def _make_iterencode(markers, _default, _encoder, _indent, _
floatstr,
/usr/local/lib/python3.5/site-packages/mpld3/_display.py in default
(self, obj)
    136             numpy.float64)):
    137             return float(obj)
--> 138         return json.JSONEncoder.default(self, obj)
    139 
    140 

/usr/local/lib/python3.5/json/encoder.py in default(self, o)
    177 
    178         """
--> 179         raise TypeError(repr(o) + " is not JSON serializabl
e")
    180 
    181     def encode(self, o):

TypeError: array([ 0.3]) is not JSON serializable
```



In [71]:

```
train_stats_as_per_source_category.plot.line()
```

Out[71]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177ab45ac8>
```

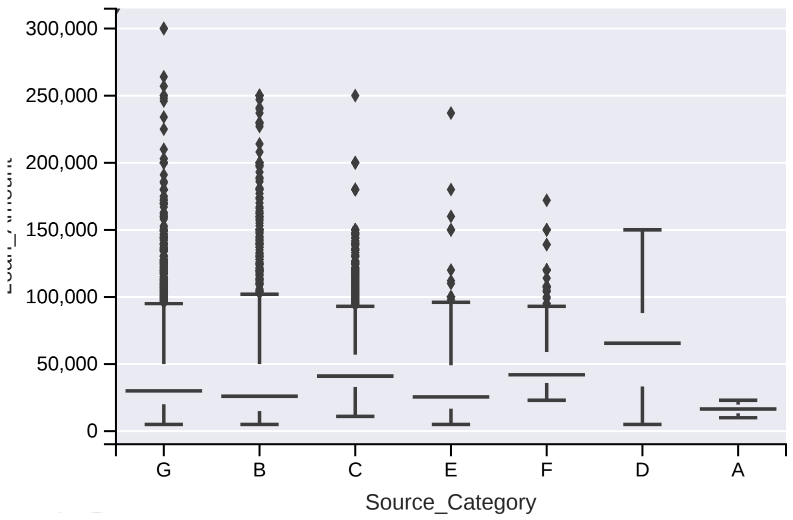
Box Plot (Bivariate Analysis)

```
In [103]:
```

```
sns.boxplot(x='Source_Category', y='Loan_Amount',data=train)
```

```
Out[103]:
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f177e8b5fd0>
```



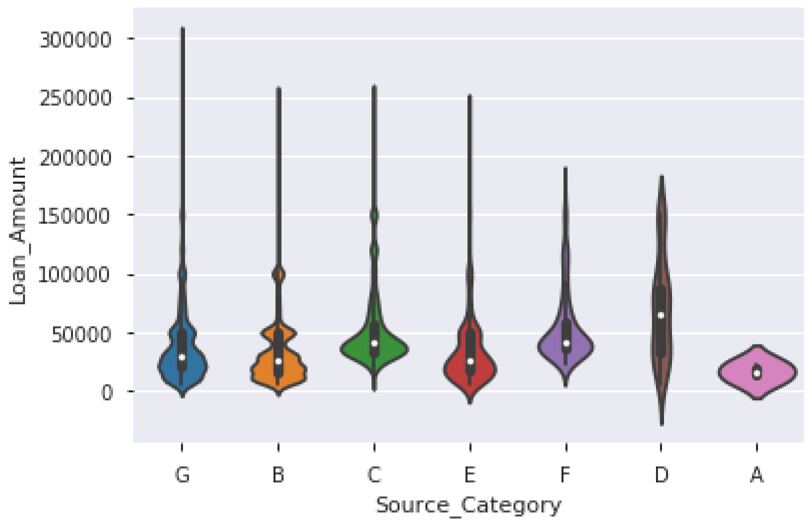
Violin Plot (Bivariate Analysis)

In [104]:

```
sns.violinplot(x='Source_Category', y='Loan_Amount',data=train)
```

```
Out[104]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f177e7818d0>
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent cal
l last)
/usr/local/lib/python3.5/site-packages/IPython/core/formatters.py in
 __call__(self, obj)
    339                 pass
    340             else:
--> 341                 return printer(obj)
    342             # Finally look for special method names
    343             method = get_real_method(obj, self.print_method)

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in <lambda>
(fig, kwds)
    408     formatter = ip.display_formatter.formatters['text/html']
    409     formatter.for_type(Figure,
--> 410                        lambda fig, kwds=kwargs: fig_to_html
(fig, **kwds))
    411
    412

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in fig_to_h
tml(fig, d3_url, mpld3_url, no_extras, template_type, figid, use_htt
p, **kwargs)
    249                            d3_url=d3_url,
    250                            mpld3_url=mpld3_url,
--> 251                            figure_json=json.dumps(figure_jso
n, cls=NumpyEncoder),
    252                            extra_css=extra_css,
    253                            extra_js=extra_js)

/usr/local/lib/python3.5/json/__init__.py in dumps(obj, skipkeys, en
sure_ascii, check_circular, allow_nan, cls, indent, separators, defa
ult, sort_keys, **kw)
    235         check_circular=check_circular, allow_nan=allow_nan,
 indent=indent,
    236         separators=separators, default=default, sort_keys=so
rt_keys,
--> 237         **kw).encode(obj)
    238
    239

/usr/local/lib/python3.5/json/encoder.py in encode(self, o)
    196         # exceptions aren't as detailed.  The list call shou
ld be roughly
    197         # equivalent to the PySequence_Fast that ''.join() w
ould do.
--> 198         chunks = self.iterencode(o, _one_shot=True)
    199         if not isinstance(chunks, (list, tuple)):
    200             chunks = list(chunks)

/usr/local/lib/python3.5/json/encoder.py in iterencode(self, o, _one
_shot)
    254                 self.key_separator, self.item_separator, sel
f.sort_keys,
    255                 self.skipkeys, _one_shot)
--> 256         return _iterencode(o, 0)
    257
    258 def _make_iterencode(markers, _default, _encoder, _indent, _
floatstr,
/usr/local/lib/python3.5/site-packages/mpld3/_display.py in default
(self, obj)
    136                 numpy.float64)):
    137                 return float(obj)
--> 138             return json.JSONEncoder.default(self, obj)
    139
    140

/usr/local/lib/python3.5/json/encoder.py in default(self, o)
    177
    178             """
--> 179             raise TypeError(repr(o) + " is not JSON serializabl
e")
    180
    181         def encode(self, o):

TypeError: array([ 1.75]) is not JSON serializable
```



Multivariate Plots

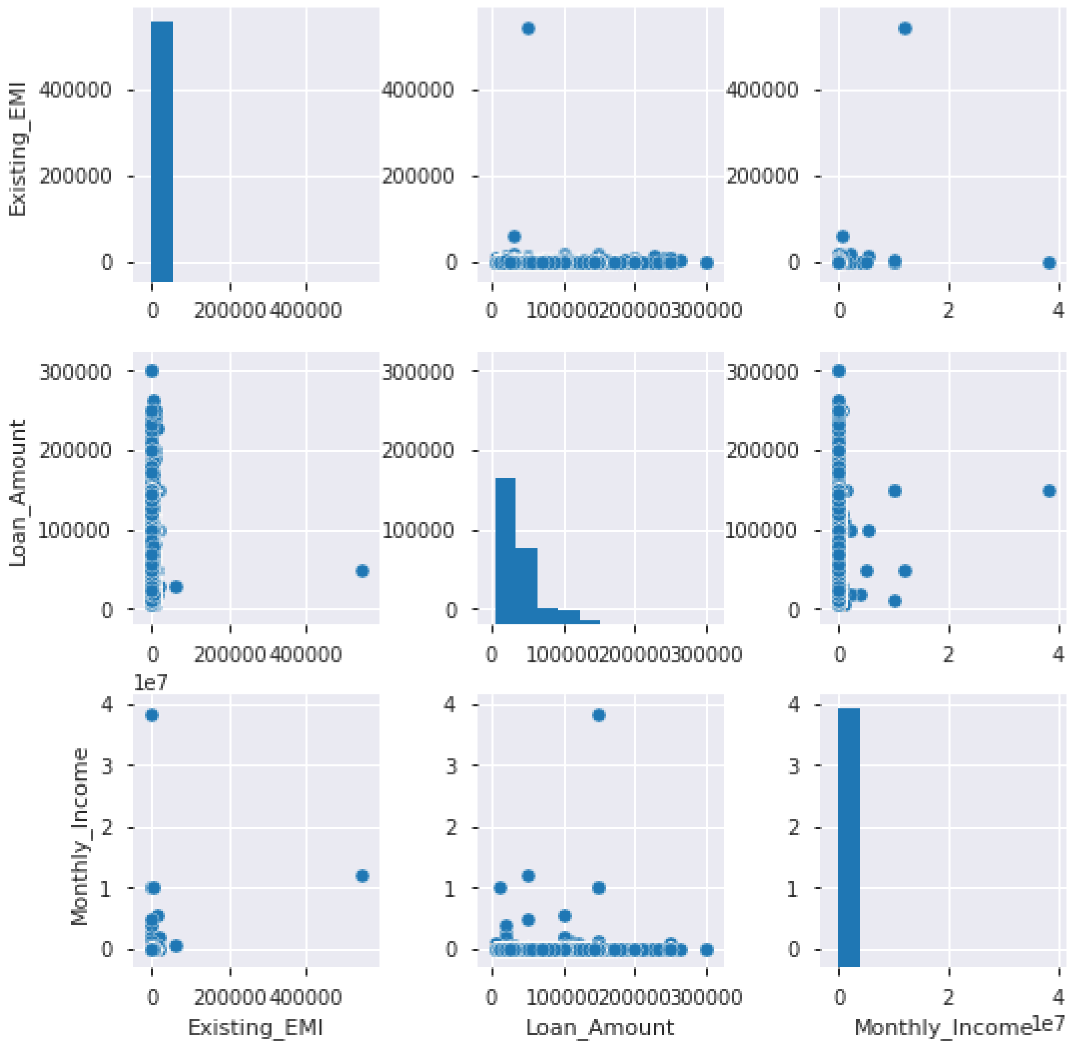
Pairplot (Multivariate plots)

In [106]:

```
sns.pairplot(train[['Existing_EMI', 'Loan_Amount', 'Monthly_Income']].dropna())
```

```
Out[106]:

<seaborn.axisgrid.PairGrid at 0x7f177e7dfe80>
```

```
--------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
/usr/local/lib/python3.5/site-packages/IPython/core/formatters.py in
 __call__(self, obj)
    339                 pass
    340             else:
--> 341                 return printer(obj)
    342             # Finally look for special method names
    343             method = get_real_method(obj, self.print_method)

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in <lambda>
(fig, kwds)
    408     formatter = ip.display_formatter.formatters['text/html']
    409     formatter.for_type(Figure,
--> 410                        lambda fig, kwds=kwargs: fig_to_html
(fig, **kwds))
    411 
    412 

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in fig_to_h
tml(fig, d3_url, mpld3_url, no_extras, template_type, figid, use_htt
p, **kwargs)
    249                            d3_url=d3_url,
    250                            mpld3_url=mpld3_url,
--> 251                            figure_json=json.dumps(figure_jso
n, cls=NumpyEncoder),
    252                            extra_css=extra_css,
    253                            extra_js=extra_js)

/usr/local/lib/python3.5/json/__init__.py in dumps(obj, skipkeys, en
sure_ascii, check_circular, allow_nan, cls, indent, separators, defa
ult, sort_keys, **kw)
    235         check_circular=check_circular, allow_nan=allow_nan,
 indent=indent,
    236         separators=separators, default=default, sort_keys=so
rt_keys,
--> 237         **kw).encode(obj)
    238 
    239 

/usr/local/lib/python3.5/json/encoder.py in encode(self, o)
    196         # exceptions aren't as detailed.  The list call shou
ld be roughly
    197         # equivalent to the PySequence_Fast that ''.join() w
ould do.
--> 198         chunks = self.iterencode(o, _one_shot=True)
    199         if not isinstance(chunks, (list, tuple)):
    200             chunks = list(chunks)

/usr/local/lib/python3.5/json/encoder.py in iterencode(self, o, _one
_shot)
    254                 self.key_separator, self.item_separator, sel
f.sort_keys,
    255                 self.skipkeys, _one_shot)
--> 256         return _iterencode(o, 0)
    257 
    258 def _make_iterencode(markers, _default, _encoder, _indent, _
floatstr,
/usr/local/lib/python3.5/site-packages/mpld3/_display.py in default
(self, obj)
    136             numpy.float64)):
    137             return float(obj)
--> 138         return json.JSONEncoder.default(self, obj)
    139 
    140 

/usr/local/lib/python3.5/json/encoder.py in default(self, o)
    177 
    178         """
--> 179         raise TypeError(repr(o) + " is not JSON serializabl
e")
    180 
    181     def encode(self, o):

TypeError: array([ 0.3]) is not JSON serializable
```



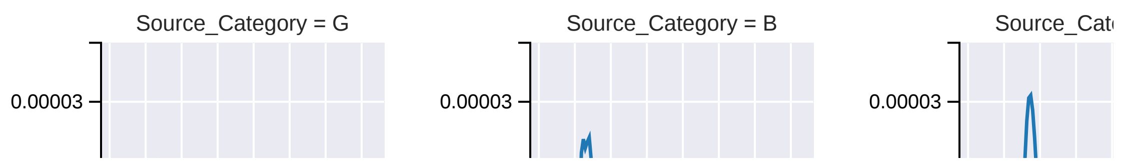
Facet Grid (Multivariate plots)

In [109]:

```
#A FacetGrid is an object which stores some information on how you want to break
 up your data visualization.
g = sns.FacetGrid(train, col="Source_Category")
g.map(sns.kdeplot, "Loan_Amount")
```

Out[109]:

```
<seaborn.axisgrid.FacetGrid at 0x7f177e5a3390>
```



Sub Plots (Multivariate plots)

In [ ]:

```
#fig, axarr = plt.subplots(<number_of_rows>, <number_of_columns>, figsize=(<alon
g_x_axis>, <along_y_axis>))
fig, axarr = plt.subplots(2, 2, figsize=(12, 8))
```

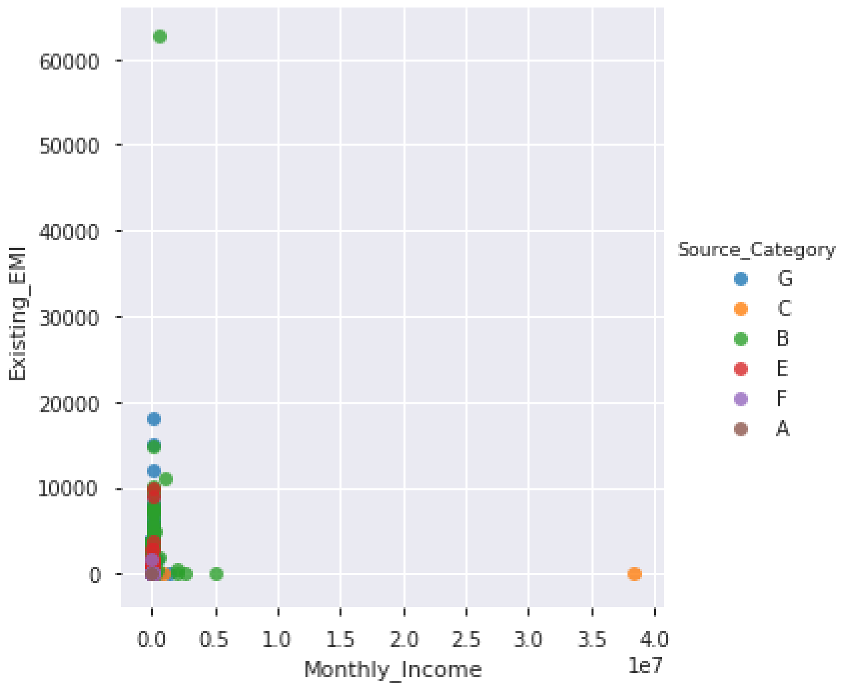
Scatter Plots (Multivariate plots)

In [110]:

```
sns.lmplot(x='Monthly_Income', y='Existing_EMI', hue='Source_Category', data=tra
in.dropna(), fit_reg=False)
```

```
Out[110]:

<seaborn.axisgrid.FacetGrid at 0x7f177e8a26d8>
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent cal
l last)
/usr/local/lib/python3.5/site-packages/IPython/core/formatters.py in
 __call__(self, obj)
    339                 pass
    340             else:
--> 341                 return printer(obj)
    342             # Finally look for special method names
    343             method = get_real_method(obj, self.print_method)

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in <lambda>
(fig, kwds)
    408     formatter = ip.display_formatter.formatters['text/html']
    409     formatter.for_type(Figure,
--> 410                        lambda fig, kwds=kwargs: fig_to_html
(fig, **kwds))
    411
    412

/usr/local/lib/python3.5/site-packages/mpld3/_display.py in fig_to_h
tml(fig, d3_url, mpld3_url, no_extras, template_type, figid, use_htt
p, **kwargs)
    249                            d3_url=d3_url,
    250                            mpld3_url=mpld3_url,
--> 251                            figure_json=json.dumps(figure_jso
n, cls=NumpyEncoder),
    252                            extra_css=extra_css,
    253                            extra_js=extra_js)

/usr/local/lib/python3.5/json/__init__.py in dumps(obj, skipkeys, en
sure_ascii, check_circular, allow_nan, cls, indent, separators, defa
ult, sort_keys, **kw)
    235         check_circular=check_circular, allow_nan=allow_nan,
 indent=indent,
    236         separators=separators, default=default, sort_keys=so
rt_keys,
--> 237         **kw).encode(obj)
    238
    239

/usr/local/lib/python3.5/json/encoder.py in encode(self, o)
    196         # exceptions aren't as detailed.  The list call shou
ld be roughly
    197         # equivalent to the PySequence_Fast that ''.join() w
ould do.
--> 198         chunks = self.iterencode(o, _one_shot=True)
    199         if not isinstance(chunks, (list, tuple)):
    200             chunks = list(chunks)

/usr/local/lib/python3.5/json/encoder.py in iterencode(self, o, _one
_shot)
    254                 self.key_separator, self.item_separator, sel
f.sort_keys,
    255                 self.skipkeys, _one_shot)
--> 256         return _iterencode(o, 0)
    257
    258 def _make_iterencode(markers, _default, _encoder, _indent, _
floatstr,
/usr/local/lib/python3.5/site-packages/mpld3/_display.py in default
(self, obj)
    136             numpy.float64)):
    137             return float(obj)
--> 138         return json.JSONEncoder.default(self, obj)
    139 
    140 

/usr/local/lib/python3.5/json/encoder.py in default(self, o)
    177 
    178         """
--> 179         raise TypeError(repr(o) + " is not JSON serializabl
e")
    180 
    181     def encode(self, o):

TypeError: array([ 0.]) is not JSON serializable
```



Heat Map (Multivariate Plots)

In [111]:

```
sns.heatmap(train[['Monthly_Income', 'EMI', 'Existing_EMI']].corr(),annot=True)
```

```
Out[111]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f1793701a90>
```