schulzdLab1

December 6, 2020

1 Lab 1: Data Cleaning

David Schulz

1.1 Introduction

In this lab, we are going to inspect and clean a data set of real estate transactions from California. We will determine when variables should be represented categorically, clarify what the variables represent, and filter variables down to make them more useful to our analysis.

1.2 1. Loading the Data and Initial Assessment

[1]:
```
import pandas as pd

data = pd.read_csv("Sacramentorealestatetransactions.csv")

print(data.head())
print()
print(data.info())
```

```
             street        city    zip state  beds  baths  sq__ft  \
0      3526 HIGH ST  SACRAMENTO  95838    CA     2      1     836
1       51 OMAHA CT  SACRAMENTO  95823    CA     3      1    1167
2    2796 BRANCH ST  SACRAMENTO  95815    CA     2      1     796
3  2805 JANETTE WAY  SACRAMENTO  95815    CA     2      1     852
4   6001 MCMAHON DR  SACRAMENTO  95824    CA     2      1     797

          type                     sale_date  price   latitude   longitude
0  Residential  Wed May 21 00:00:00 EDT 2008  59222  38.631913 -121.434879
1  Residential  Wed May 21 00:00:00 EDT 2008  68212  38.478902 -121.431028
2  Residential  Wed May 21 00:00:00 EDT 2008  68880  38.618305 -121.443839
3  Residential  Wed May 21 00:00:00 EDT 2008  69307  38.616835 -121.439146
4  Residential  Wed May 21 00:00:00 EDT 2008  81900  38.519470 -121.435768

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 985 entries, 0 to 984
```

```
Data columns (total 12 columns):
street       985 non-null object
city         985 non-null object
zip          985 non-null int64
state        985 non-null object
beds         985 non-null int64
baths        985 non-null int64
sq__ft       985 non-null int64
type         985 non-null object
sale_date    985 non-null object
price        985 non-null int64
latitude     985 non-null float64
longitude    985 non-null float64
dtypes: float64(2), int64(5), object(5)
memory usage: 92.4+ KB
None
```

Street is an object, city is an object, zip is an int64, state is an object, beds is an int64, baths is an int64, sq__ft is an int64, type is an object, sale_date is an object, price is an int64, latitude is a float64, and longitude is a float64. None of the columns have null values because they're all supposed to be non-null.

1.3 2. Representing Categorical Variables

[2]:
```
print(data['street'].nunique())
print(data['zip'].nunique())
print(data['beds'].nunique())
```

```
981
68
8
```

Streets: 981

Zip codes: 68

Beds: 8

It's probably more appropriate to represent them as categorical variables because

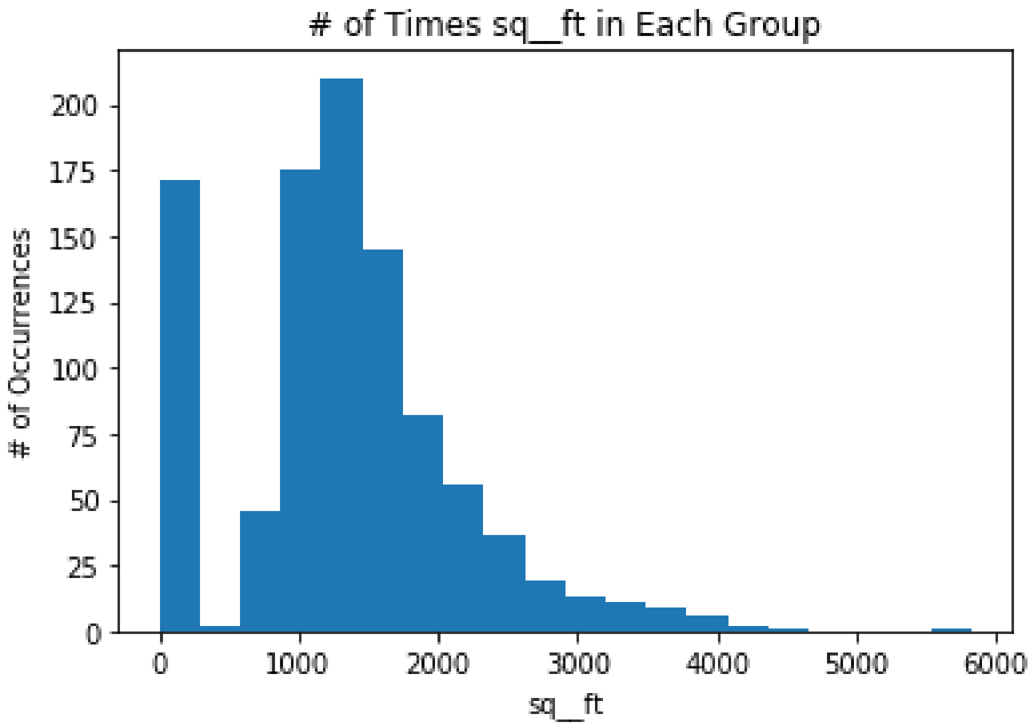
[3]:
```
data['city'] = data['city'].astype('category')
data['state'] = data['state'].astype('category')
data['zip'] = data['zip'].astype('category')
data['beds'] = data['beds'].astype('category')
data['baths'] = data['baths'].astype('category')
data['type'] = data['type'].astype('category')

print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 985 entries, 0 to 984
Data columns (total 12 columns):
street       985 non-null object
city         985 non-null category
zip          985 non-null category
state        985 non-null category
beds         985 non-null category
baths        985 non-null category
sq__ft       985 non-null int64
type         985 non-null category
sale_date    985 non-null object
price        985 non-null int64
latitude     985 non-null float64
longitude    985 non-null float64
dtypes: category(6), float64(2), int64(2), object(2)
memory usage: 57.5+ KB
None
```

1.4 3. Cleaning Continuous Variables

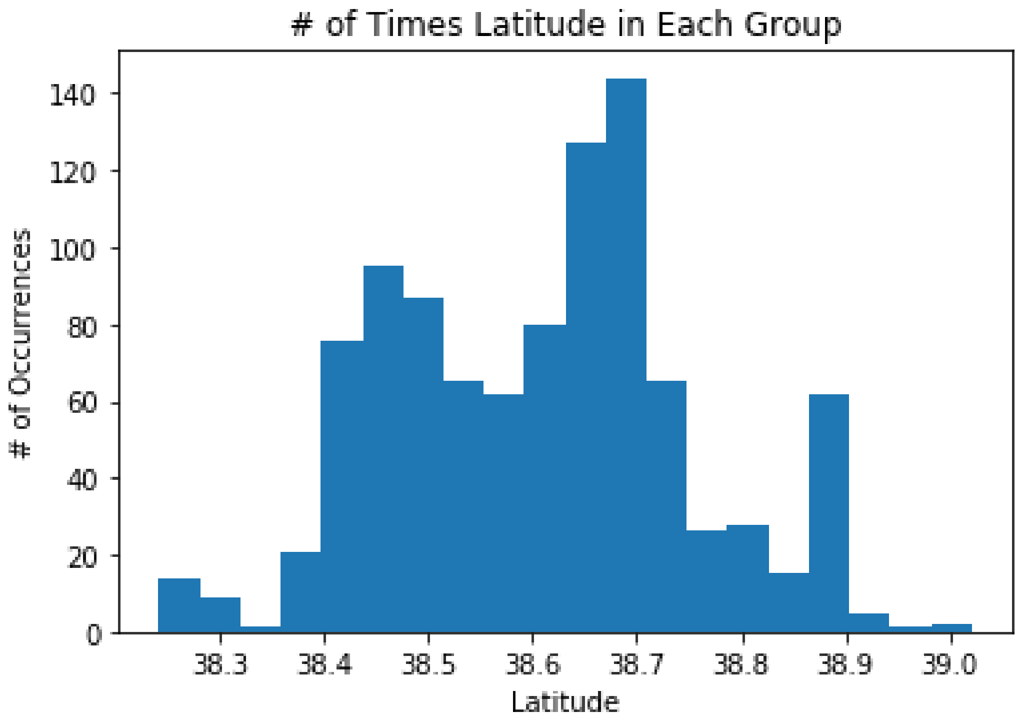
[4]:
```
import matplotlib.pyplot as plt

plt.title('# of Times sq__ft in Each Group')
plt.xlabel('sq__ft')
plt.ylabel('# of Occurrences')
plt.hist(data['sq__ft'], bins=20)
```

[4]:
```
(array([171.,   2.,  46., 175., 210., 145.,  82.,  56.,  36.,  19.,  13.,
         11.,   9.,   6.,   2.,   1.,   0.,   0.,   0.,   1.]),
 array([   0. ,  291.1,  582.2,  873.3, 1164.4, 1455.5, 1746.6, 2037.7,
        2328.8, 2619.9, 2911. , 3202.1, 3493.2, 3784.3, 4075.4, 4366.5,
        4657.6, 4948.7, 5239.8, 5530.9, 5822. ]),
 <a list of 20 Patch objects>)
```



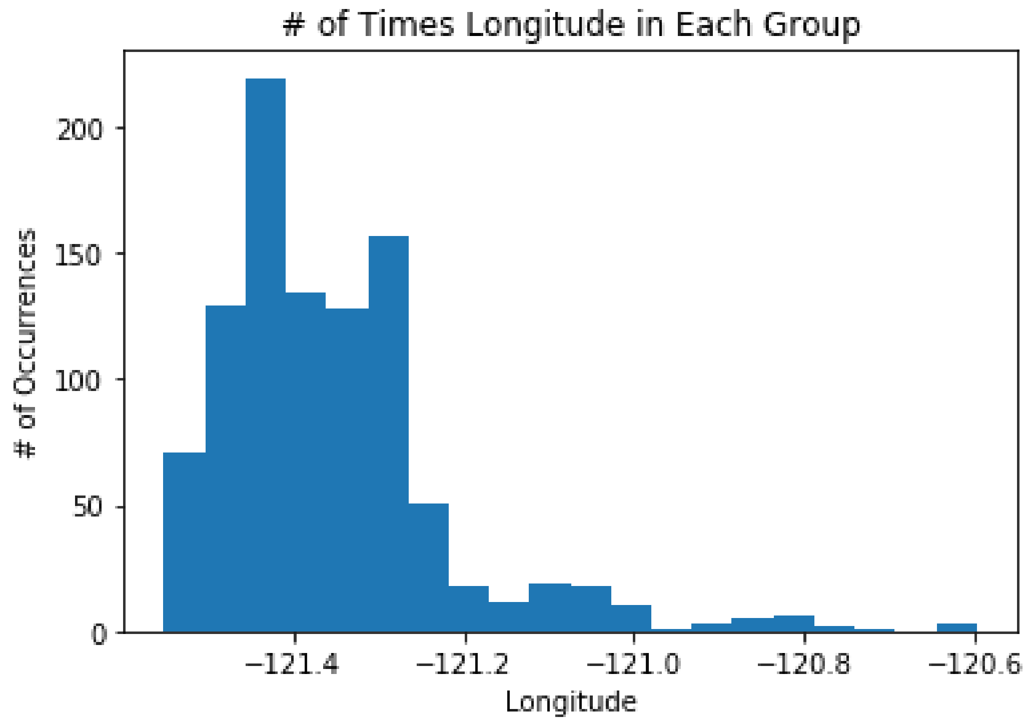
[5]:
```
plt.title('# of Times Latitude in Each Group')
plt.xlabel('Latitude')
plt.ylabel('# of Occurrences')
plt.hist(data['latitude'], bins=20)
```

[5]:
```
(array([ 14.,   9.,   1.,  21.,  76.,  95.,  87.,  65.,  62.,  80., 127.,
        144.,  65.,  26.,  28.,  15.,  62.,   5.,   1.,   2.]),
 array([38.241514 , 38.2804787, 38.3194434, 38.3584081, 38.3973728,
        38.4363375, 38.4753022, 38.5142669, 38.5532316, 38.5921963,
        38.631161 , 38.6701257, 38.7090904, 38.7480551, 38.7870198,
        38.8259845, 38.8649492, 38.9039139, 38.9428786, 38.9818433,
        39.020808 ]),
 <a list of 20 Patch objects>)
```




```
[6]: plt.title('# of Times Longitude in Each Group')
     plt.xlabel('Longitude')
     plt.ylabel('# of Occurrences')
     plt.hist(data['longitude'], bins=20)
```

```
[6]: (array([ 71., 129., 219., 134., 128., 156.,  51.,  18.,  11.,  19.,  18.,
              10.,   1.,   3.,   5.,   6.,   2.,   1.,   0.,   3.]),
      array([-121.551704  , -121.50399875, -121.4562935 , -121.40858825,
             -121.360883  , -121.31317775, -121.2654725 , -121.21776725,
             -121.170062  , -121.12235675, -121.0746515 , -121.02694625,
             -120.979241  , -120.93153575, -120.8838305 , -120.83612525,
             -120.78842   , -120.74071475, -120.6930095 , -120.64530425,
             -120.597599  ]),
      <a list of 20 Patch objects>)
```



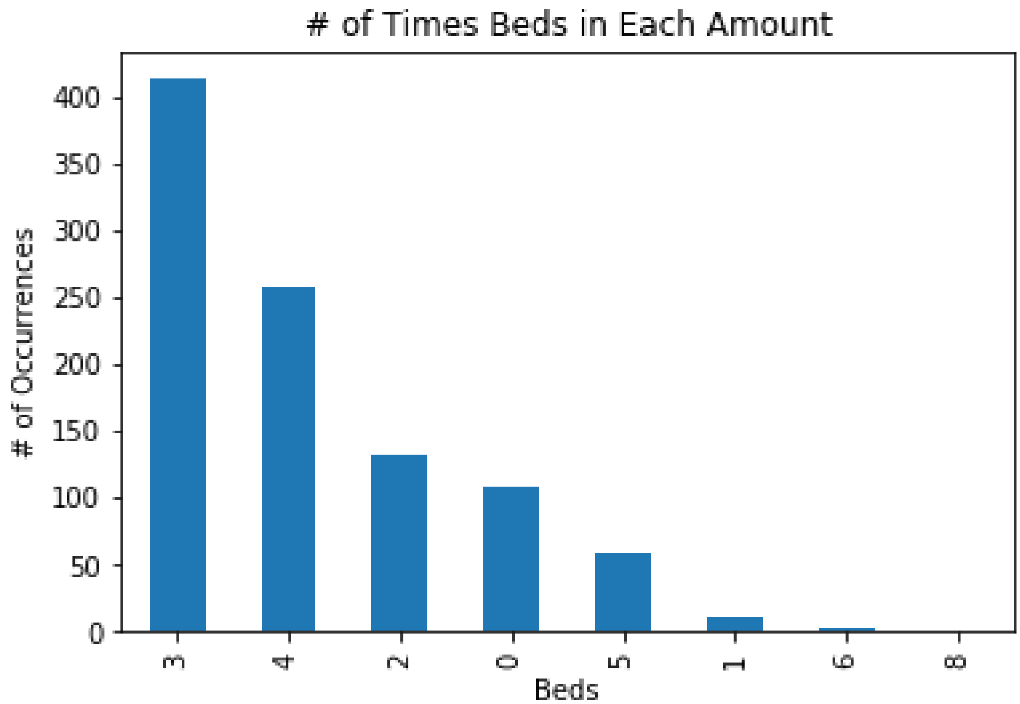
A histogram with a sufficient number of bins to show the number of entries in a small group accurately enough is appropriate.

The fact that almost 175 of the entries have a square footage of zero is very noticeable. Since it's impossible to have a square footage of zero, I'm pretty sure they are artifacts.

1.5 4. Cleaning Categorical Variables

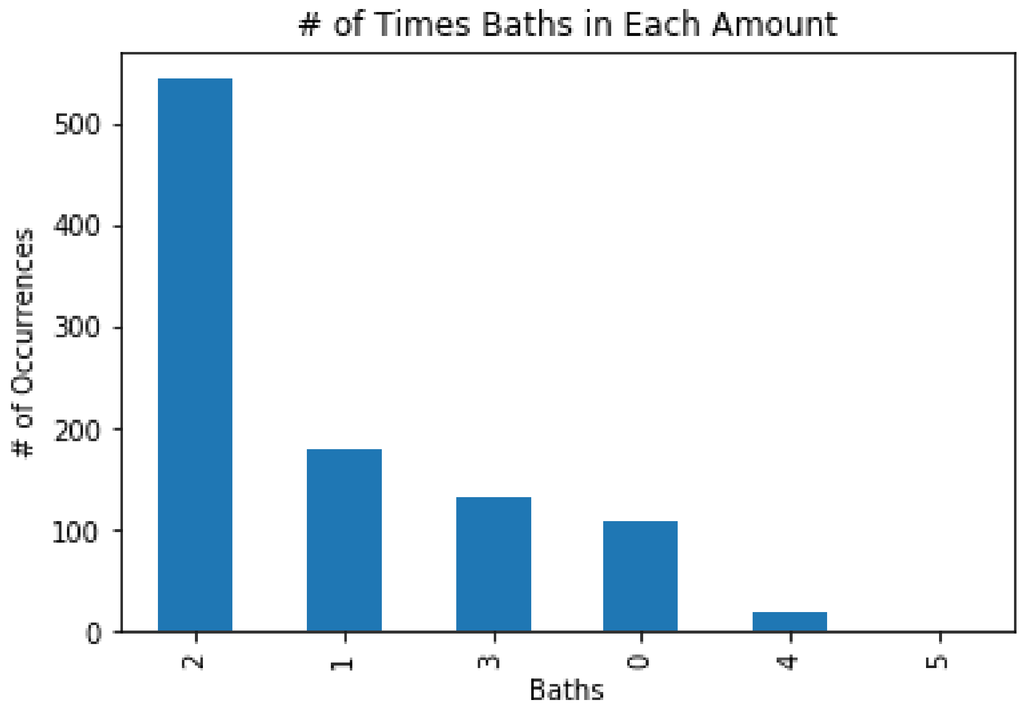
[7]:
```
plt.title('# of Times Beds in Each Amount')
plt.xlabel('Beds')
plt.ylabel('# of Occurrences')
data['beds'].value_counts().plot(kind='bar')
```

[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f01808aaad0>
```



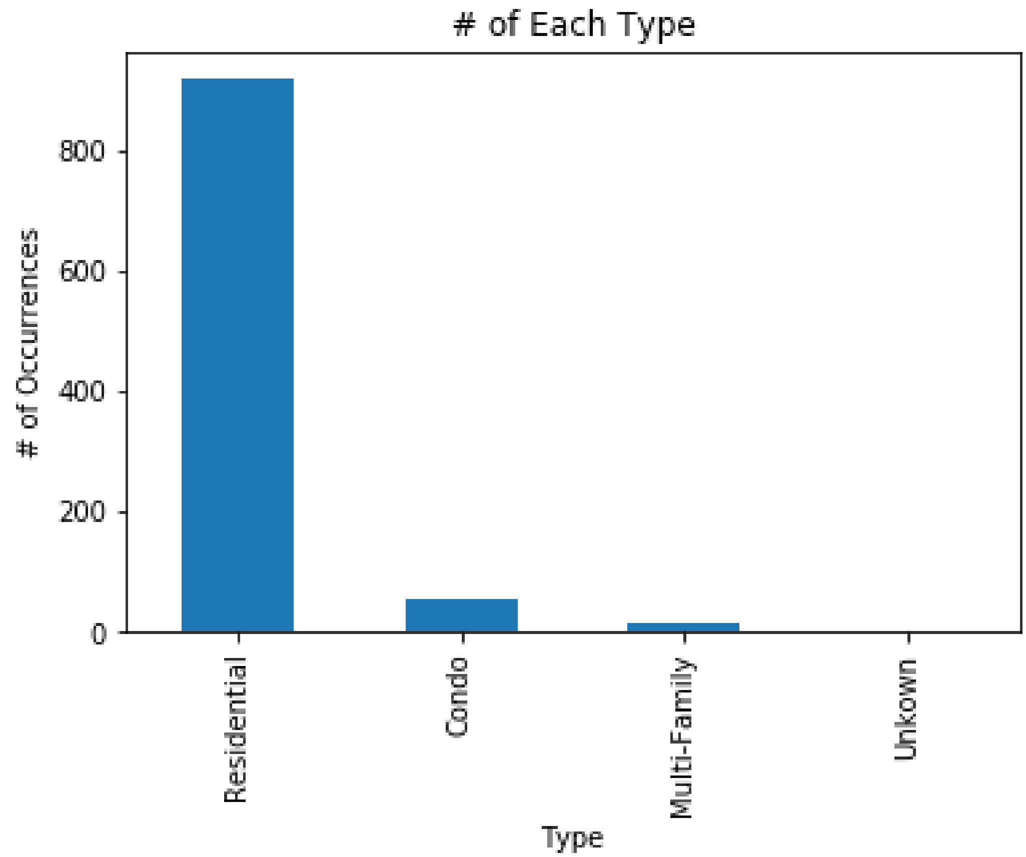
[8]:
```
plt.title('# of Times Baths in Each Amount')
plt.xlabel('Baths')
plt.ylabel('# of Occurrences')
data['baths'].value_counts().plot(kind='bar')
```

[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7f01807db510>



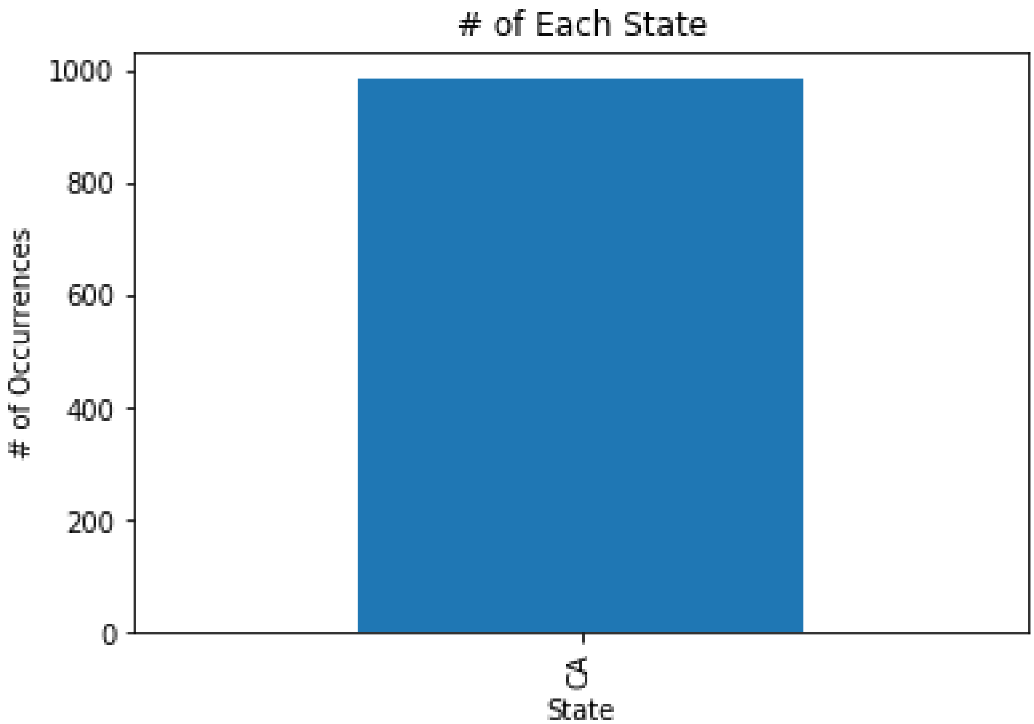
[9]:
```
plt.title('# of Each Type')
plt.xlabel('Type')
plt.ylabel('# of Occurrences')
data['type'].value_counts().plot(kind='bar')
```

[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7f0180748e90>



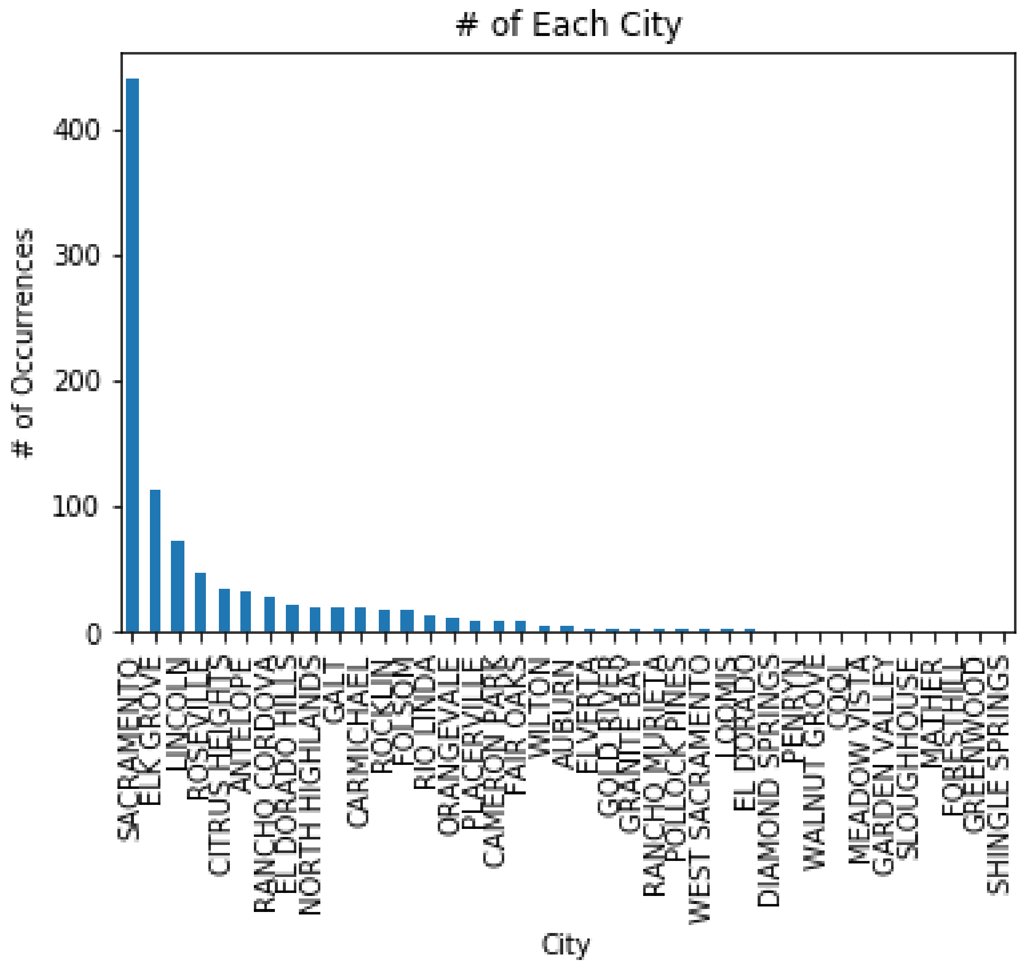
[10]:
```
plt.title('# of Each State')
plt.xlabel('State')
plt.ylabel('# of Occurrences')
data['state'].value_counts().plot(kind='bar')
```

[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7f0180726b90>



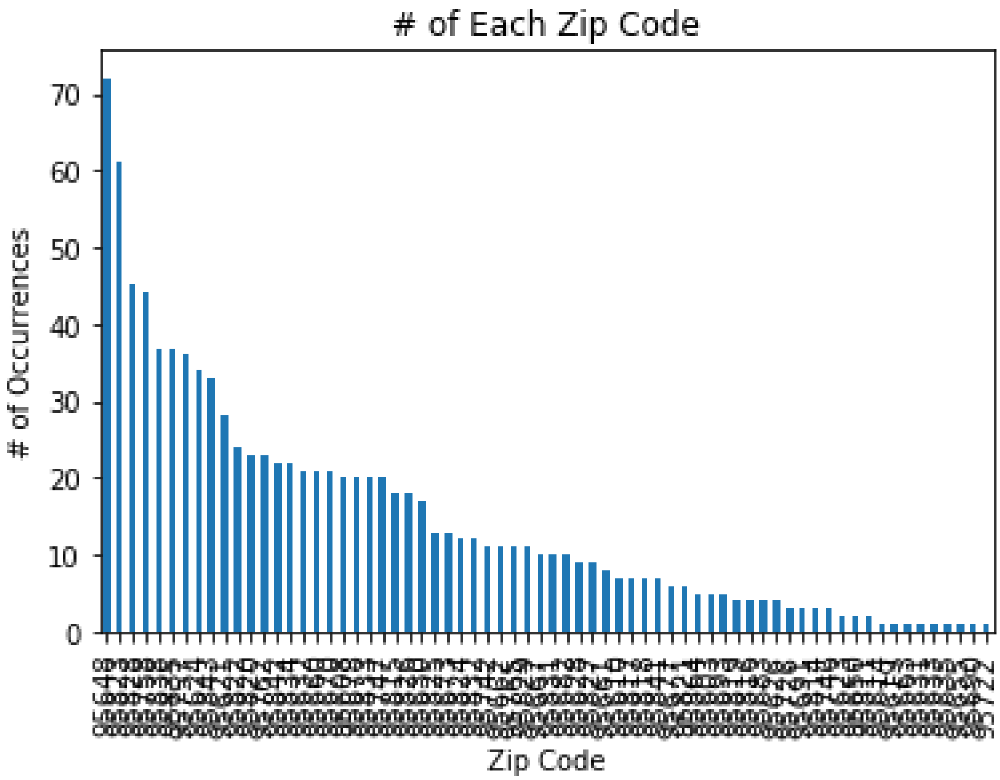

```
[11]: plt.title('# of Each City')
      plt.xlabel('City')
      plt.ylabel('# of Occurrences')
      data['city'].value_counts().plot(kind='bar')
```

```
[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7f01807322d0>
```



[12]:
```
plt.title('# of Each Zip Code')
plt.xlabel('Zip Code')
plt.ylabel('# of Occurrences')
data['zip'].value_counts().plot(kind='bar')
```

[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0180689950>
```

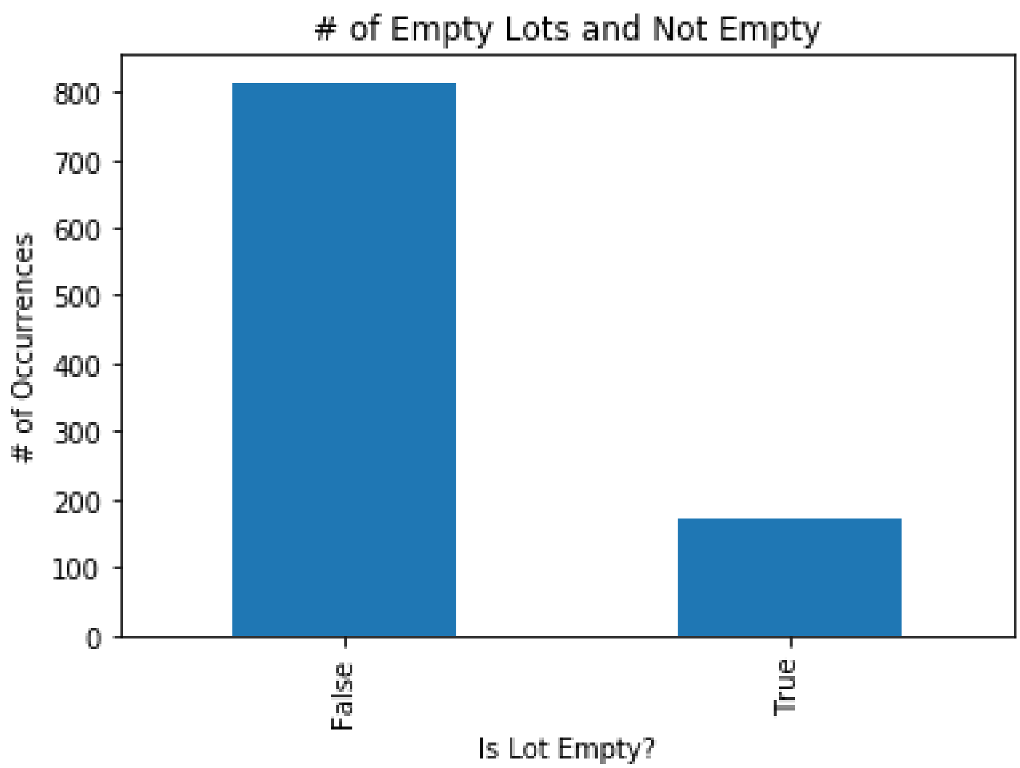


There are properties that have 0 bedrooms and/or 0 bathrooms. They could be empty lots that have yet to be built on.

1.6 5. Engineering New Variables - Part I

[13]:
```
import numpy as np

data['empty_lot'] = np.where(data['sq__ft'] == 0, True, False)

plt.title('# of Empty Lots and Not Empty')
plt.xlabel('Is Lot Empty?')
plt.ylabel('# of Occurrences')
data['empty_lot'].value_counts().plot(kind='bar')
```

[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f01803ed990>
```



1.7 6. Engineering New Variables - Part II

[14]:
```
print(data['street'].nunique())
```

```
981
```

It's not useful for analysis or ML in its current form.

[15]:
```
data.head(n=20)
```

[15]:

	street	city	zip	state	beds	baths	\
0	3526 HIGH ST	SACRAMENTO	95838	CA	2	1	
1	51 OMAHA CT	SACRAMENTO	95823	CA	3	1	
2	2796 BRANCH ST	SACRAMENTO	95815	CA	2	1	
3	2805 JANETTE WAY	SACRAMENTO	95815	CA	2	1	
4	6001 MCMAHON DR	SACRAMENTO	95824	CA	2	1	
5	5828 PEPPERMILL CT	SACRAMENTO	95841	CA	3	1	
6	6048 OGDEN NASH WAY	SACRAMENTO	95842	CA	3	2	
7	2561 19TH AVE	SACRAMENTO	95820	CA	3	1	
8	11150 TRINITY RIVER DR Unit 114	RANCHO CORDOVA	95670	CA	2	2	
9	7325 10TH ST	RIO LINDA	95673	CA	3	2	
10	645 MORRISON AVE	SACRAMENTO	95838	CA	3	2	

```
11               4085 FAWN CIR      SACRAMENTO  95823    CA    3     2
12             2930 LA ROSA RD      SACRAMENTO  95815    CA    1     1
13               2113 KIRK WAY      SACRAMENTO  95822    CA    3     1
14         4533 LOCH HAVEN WAY      SACRAMENTO  95842    CA    2     2
15              7340 HAMDEN PL      SACRAMENTO  95842    CA    2     2
16                 6715 6TH ST       RIO LINDA  95673    CA    2     1
17     6236 LONGFORD DR Unit 1  CITRUS HEIGHTS  95621    CA    2     1
18             250 PERALTA AVE      SACRAMENTO  95833    CA    2     1
19             113 LEEWILL AVE       RIO LINDA  95673    CA    3     2

   sq__ft         type                     sale_date   price   latitude  \
0     836  Residential  Wed May 21 00:00:00 EDT 2008   59222  38.631913
1    1167  Residential  Wed May 21 00:00:00 EDT 2008   68212  38.478902
2     796  Residential  Wed May 21 00:00:00 EDT 2008   68880  38.618305
3     852  Residential  Wed May 21 00:00:00 EDT 2008   69307  38.616835
4     797  Residential  Wed May 21 00:00:00 EDT 2008   81900  38.519470
5    1122        Condo  Wed May 21 00:00:00 EDT 2008   89921  38.662595
6    1104  Residential  Wed May 21 00:00:00 EDT 2008   90895  38.681659
7    1177  Residential  Wed May 21 00:00:00 EDT 2008   91002  38.535092
8     941        Condo  Wed May 21 00:00:00 EDT 2008   94905  38.621188
9    1146  Residential  Wed May 21 00:00:00 EDT 2008   98937  38.700909
10    909  Residential  Wed May 21 00:00:00 EDT 2008  100309  38.637663
11   1289  Residential  Wed May 21 00:00:00 EDT 2008  106250  38.470746
12    871  Residential  Wed May 21 00:00:00 EDT 2008  106852  38.618698
13   1020  Residential  Wed May 21 00:00:00 EDT 2008  107502  38.482215
14   1022  Residential  Wed May 21 00:00:00 EDT 2008  108750  38.672914
15   1134        Condo  Wed May 21 00:00:00 EDT 2008  110700  38.700051
16    844  Residential  Wed May 21 00:00:00 EDT 2008  113263  38.689591
17    795        Condo  Wed May 21 00:00:00 EDT 2008  116250  38.679776
18    588  Residential  Wed May 21 00:00:00 EDT 2008  120000  38.612099
19   1356  Residential  Wed May 21 00:00:00 EDT 2008  121630  38.689999

     longitude  empty_lot
0  -121.434879      False
1  -121.431028      False
2  -121.443839      False
3  -121.439146      False
4  -121.435768      False
5  -121.327813      False
6  -121.351705      False
7  -121.481367      False
8  -121.270555      False
9  -121.442979      False
10 -121.451520      False
11 -121.458918      False
12 -121.435833      False
13 -121.492603      False
```

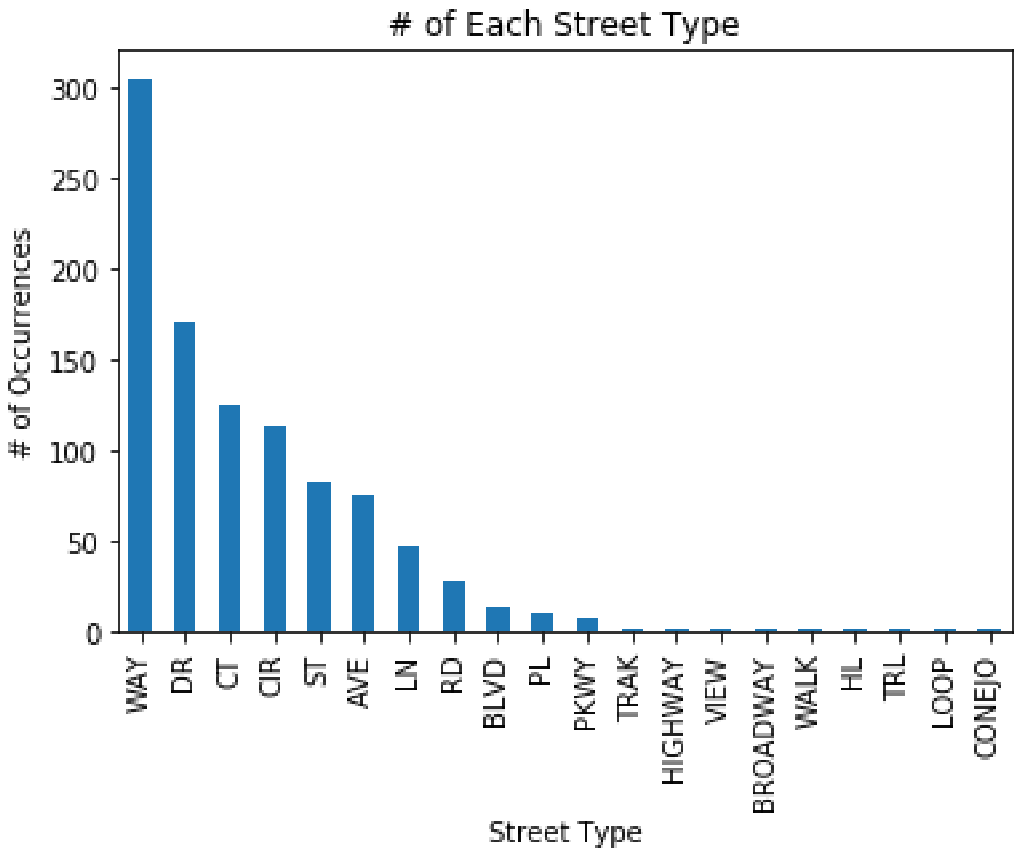
```
14 -121.359340      False
15 -121.351278      False
16 -121.452239      False
17 -121.314089      False
18 -121.469095      False
19 -121.463220      False
```

The street type is almost always the last word of the string, unless it's followed by "Unit".

[16]:
```python
def get_street_type(address) -> str:
    words = address.split()
    if "Unit" in address:
        return words[-3]
    elif "HIGHWAY" in address:
        return words[-2]
    elif "VIA" in address:
        return "WAY"
    elif "AVENIDA" in address:
        return "AVE"
    elif "MADERA" in address:
        return "VIEW"
    else:
        return words[-1]
```

[17]:
```python
data['street_type'] = data['street'].map(get_street_type)

plt.title('# of Each Street Type')
plt.xlabel('Street Type')
plt.ylabel('# of Occurrences')
data['street_type'].value_counts().plot(kind='bar')
```

[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0180408050>
```



1.8 7. Identifying Potential Dependent Variables

- Variables that are continuous, such as floats, are appropriate for regression.
- Variables that are not continuous and have specific values from a given list of possible choices, such as strings or integers, are appropriate for classification.
- The average speed, in MPH, of a vehicle, which would be a float, would be a good dependent variable for a regression problem.
- The make of a vehicle, which would be a string chosen from a small list of possible strings, would be a good dependent variable for a classification problem.

1.9 8. Save the Cleaned Data Set

[18]:
```
data.to_csv("cleaned_Sacramentorealestatetransactions.csv", index=False)
```