

# Midterm 2\_8

February 8, 2021

## 0.1 Midterm Project: Tracking Water Bill Debt across California

### 0.1.1 Load Libraries and Data

The first type of data I'm working with is a combination of 2 datasets (which has previously been merged by zip code) \* Water bill debt information \* American Community Survey Data

These two datasets, together, will help me better understand where drinking water debt is distributed across the state as well as which demographic factors coincide with higher or lower debt levels.

The second piece of my data is the spatial component. It is a GeoJson file of all zip code boundaries in the state of California.

First, I am going to load my libraries. I want to be able to read and map my data, so I've selected the 5 following libraries.

```
[1]: import pandas as pd
import geopandas as gpd
import contextily as ctx
import matplotlib.pyplot as plt
import urllib.request, json
import plotly.express as px
from ipywidgets import interact
```

Now that I can read my data with pandas, I am going to upload the water bill data and zip code boundary file.

```
[2]: acs = pd.read_csv('Data/URBNPL206A Dataset - Sheet1.csv')

zip = gpd.read_file('Data/ca_california_zip_codes_geo.min.json')
```

### 0.1.2 Explore the Data

Now that my data are uploaded, I need to get a sense of how they look, if any data are missing, etc. For now, I am going to focus on my water bill debt/demographic data since it is in .csv. Later, I will merge it with the GeoJson file.

```
[3]: acs.shape
```

```
[3]: (1073, 52)
```

This dataset has 1067 entries and 52 columns - it's big!

I also want to get a sense of the data itself: missing data, data type, etc.

```
[4]: acs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1073 entries, 0 to 1072
Data columns (total 52 columns):
 #   Column                                     Non-Null Count
Dtype  -----
---  -
0    Zip Codes                                1073 non-null
object
1    Count of Zip Code                        1072 non-null
float64
2    Sum of Less than $100                    934 non-null
float64
3    Sum of $100-$200                          965 non-null
float64
4    Sum of $200-$300                          965 non-null
float64
5    Sum of $300-$400                          954 non-null
float64
6    Sum of $400-$500                          941 non-null
float64
7    Sum of $500-$600                          917 non-null
float64
8    Sum of $600-$700                          914 non-null
float64
9    Sum of $700-$800                          906 non-null
float64
10   Sum of $800-$900                          895 non-null
float64
11   Sum of $900-$1000                         894 non-null
float64
12   Sum of More than $1000                     936 non-null
float64
13   Sum of Total number of delinquent residential accounts 1063 non-null
float64
14   pop                                         1030 non-null
float64
15   nhw                                         1030 non-null
float64
16   black                                       1030 non-null
float64
17   hisp                                        1030 non-null
```

float64		
18	asian	1030 non-null
float64		
19	noncitizen	1030 non-null
float64		
20	immigrants	1030 non-null
float64		
21	ohu	1030 non-null
float64		
22	lep_hh	1030 non-null
float64		
23	dpov	1030 non-null
float64		
24	npov	1030 non-null
float64		
25	mhhi	1030 non-null
float64		
26	overcrowded	1030 non-null
float64		
27	no_veh_hh	1030 non-null
float64		
28	w_broadband	1030 non-null
float64		
29	pop_19_64	1030 non-null
float64		
30	uninsured_19_64	1030 non-null
float64		
31	pct_nhw	1030 non-null
float64		
32	pct_black	1030 non-null
float64		
33	pct_hisp	1030 non-null
float64		
34	pct_asian	1030 non-null
float64		
35	pct_noncitizen	1030 non-null
float64		
36	pct_immigrants	1030 non-null
float64		
37	pct_lep_hh	1030 non-null
object		
38	pct_povt	1030 non-null
object		
39	pct_overcrowded	1030 non-null
object		
40	pct_no_veh_hh	1030 non-null
object		
41	pct_broadband	1030 non-null

```

object
  42  pct_no_broadband                                1030 non-null
object
  43  pct_uninsured_19_64                             1030 non-null
float64
  44  aggveh                                           1030 non-null
float64
  45  pct_no_hins                                       1030 non-null
float64
  46  veh_person                                       1030 non-null
float64
  47  Total Population in Occupied Housing Units: Renter Occupied 1030 non-null
float64
  48  Owner Occupied Pop                               1030 non-null
float64
  49  % Renter Pop                                     1030 non-null
float64
  50  % Owner Pop                                       1030 non-null
float64
  51  Households                                       1031 non-null
float64
dtypes: float64(45), object(7)
memory usage: 436.0+ KB

```

So here we can see that there are fewer datapoints here than total zip codes in California. There are over 1,700 zips in the state. This can largely be attributed to the fact that the water bill debt data was conducted via a survey distributed by the California State Water Resources Control Board. Survey responses have their limitations in that they are completed on a voluntary basis. It is worth keeping in mind as I continue with my analysis that this is not a complete dataset of all zip codes in the state. Hopefully, this dataset is complete enough though to draw some conclusions about water bill debt trends and demographics.

```
[5]: acs.head()
```

```

[5]:   Zip Codes  Count of Zip Code  Sum of Less than $100  Sum of $100-$200  \
0      90001              3.0          5726.0          4937.0
1      90002              3.0          3130.0          2152.0
2      90003              2.0          1829.0          1880.0
3      90004              1.0          2199.0          1659.0
4      90005              1.0          1712.0          1107.0

      Sum of $200-$300  Sum of $300-$400  Sum of $400-$500  Sum of $500-$600  \
0          1582.0          684.0          395.0          283.0
1          1052.0          708.0          493.0          385.0
2          1562.0         1169.0          941.0          782.0
3          1119.0          735.0          502.0          387.0
4           598.0          401.0          281.0          175.0

```

	Sum of \$600-\$700	Sum of \$700-\$800	...	pct_no_broadband \
0	220.0	137.0	...	0.2991675715
1	343.0	281.0	...	0.3798992602
2	673.0	513.0	...	0.3209552169
3	279.0	223.0	...	0.2220199354
4	146.0	95.0	...	0.3019705632

	pct_uninsured_19_64	aggveh	pct_no_hins	veh_person \
0	0.243428	24689.0	0.168425	0.418635
1	0.255720	21749.0	0.177007	0.409501
2	0.261466	28622.0	0.192422	0.393478
3	0.247983	29013.0	0.195940	0.471097
4	0.328389	16170.0	0.256415	0.409585

	Total Population in Occupied Housing Units: Renter Occupied \
0	38172.0
1	33035.0
2	50340.0
3	49509.0
4	36035.0

	Owner Occupied Pop	% Renter Pop	% Owner Pop	Households
0	20803.0	0.647257	0.352743	14174.0
1	20076.0	0.621999	0.378001	13546.0
2	22401.0	0.692044	0.307956	18523.0
3	12077.0	0.803900	0.196100	25192.0
4	3444.0	0.912764	0.087236	18291.0

[5 rows x 52 columns]

I actually want to over-write this feature so I can see all 52 columns, pick those of interest, and then remove the others.

```
[6]: pd.set_option('display.max_columns', None)
```

And now I'll check my work just to make sure....

```
[7]: acs.head()
```

```
[7]: Zip Codes Count of Zip Code Sum of Less than $100 Sum of $100-$200 \
0 90001 3.0 5726.0 4937.0
1 90002 3.0 3130.0 2152.0
2 90003 2.0 1829.0 1880.0
3 90004 1.0 2199.0 1659.0
4 90005 1.0 1712.0 1107.0

Sum of $200-$300 Sum of $300-$400 Sum of $400-$500 Sum of $500-$600 \
0 1582.0 684.0 395.0 283.0
```

1	1052.0	708.0	493.0	385.0
2	1562.0	1169.0	941.0	782.0
3	1119.0	735.0	502.0	387.0
4	598.0	401.0	281.0	175.0

	Sum of \$600-\$700	Sum of \$700-\$800	Sum of \$800-\$900	Sum of \$900-\$1000 \
0	220.0	137.0	132.0	97.0
1	343.0	281.0	231.0	201.0
2	673.0	513.0	482.0	401.0
3	279.0	223.0	206.0	164.0
4	146.0	95.0	95.0	58.0

	Sum of More than \$1000 \
0	709.0
1	1779.0
2	3437.0
3	1116.0
4	426.0

	Sum of Total number of delinquent residential accounts	pop	nhw \
0	14902.0	58975.0	413.0
1	10755.0	53111.0	223.0
2	13669.0	72741.0	392.0
3	8589.0	61586.0	10820.0
4	5094.0	39479.0	3020.0

	black	hisp	asian	noncitizen	immigrants	ohu	lep_hh \
0	5228.0	53086.0	129.0	17099.0	24040.0	13815.0	2876.0
1	10354.0	41673.0	322.0	13519.0	18628.0	12706.0	2203.0
2	16136.0	56050.0	257.0	19895.0	27310.0	17127.0	2829.0
3	2393.0	31479.0	15485.0	18353.0	30512.0	21971.0	5650.0
4	2423.0	19437.0	13849.0	15352.0	23346.0	16442.0	6799.0

	dpov	npov	mhhi	overcrowded	no_veh_hh	w_broadband	pop_19_64 \
0	58816.0	16911.0	38521.0	1902.0	1619.0	9682.0	37128.0
1	52845.0	17365.0	35410.0	832.0	1855.0	7879.0	32868.0
2	72362.0	22186.0	37226.0	1761.0	2830.0	11630.0	45876.0
3	61417.0	11092.0	48754.0	2825.0	3769.0	17093.0	44991.0
4	39331.0	11036.0	35149.0	3379.0	4940.0	11477.0	29054.0

	uninsured_19_64	pct_nhw	pct_black	pct_hisp	pct_asian	pct_noncitizen \
0	9038.0	0.007003	0.088648	0.900144	0.002187	0.289936
1	8405.0	0.004199	0.194950	0.784640	0.006063	0.254542
2	11995.0	0.005389	0.221828	0.770542	0.003533	0.273505
3	11157.0	0.175689	0.038856	0.511139	0.251437	0.298006
4	9541.0	0.076496	0.061374	0.492338	0.350794	0.388865

	pct_immigrants	pct_lep_hh	pct_povt	pct_overcrowded	pct_no_veh_hh	\
0	0.407630	0.208179515	0.287523803	0.1376764387	0.1171914586	
1	0.350737	0.1733826539	0.3286025168	0.06548087518	0.1459940186	
2	0.375442	0.1651777895	0.3065973854	0.1028201086	0.1652361768	
3	0.495437	0.2571571617	0.1806014621	0.1285785809	0.1715443084	
4	0.591352	0.413514171	0.2805929165	0.2055102786	0.3004500669	

	pct_broadband	pct_no_broadband	pct_uninsured_19_64	aggveh	pct_no_hins	\
0	0.7008324285	0.2991675715		0.243428	24689.0	0.168425
1	0.6201007398	0.3798992602		0.255720	21749.0	0.177007
2	0.6790447831	0.3209552169		0.261466	28622.0	0.192422
3	0.7779800646	0.2220199354		0.247983	29013.0	0.195940
4	0.6980294368	0.3019705632		0.328389	16170.0	0.256415

	veh_person	Total Population in Occupied Housing Units: Renter Occupied	\
0	0.418635		38172.0
1	0.409501		33035.0
2	0.393478		50340.0
3	0.471097		49509.0
4	0.409585		36035.0

	Owner Occupied Pop	% Renter Pop	% Owner Pop	Households
0	20803.0	0.647257	0.352743	14174.0
1	20076.0	0.621999	0.378001	13546.0
2	22401.0	0.692044	0.307956	18523.0
3	12077.0	0.803900	0.196100	25192.0
4	3444.0	0.912764	0.087236	18291.0

Okay, so looking at this data, there are some columns of interest: I want to keep zip codes, as well as the scaled debt values from less than 100 to over 1,000, the total number of delinquent accounts, the population, median household income, percent black, percent hispanic, percent asian, percent poverty, percent renter, percent owner, and households.

This is much more expansive than my actual analysis will be, but I want some flexibility in what I analyze in case some relationships I've hypothesized do not actually exist.

### 0.1.3 Cleaning the Data

Now that I have a sense of the columns I want to keep, I'm going to remove the rest, just so the data is a bit more manageable.

```
[8]: list(acs)
```

```
[8]: ['Zip Codes',
      'Count of Zip Code',
      'Sum of Less than $100',
      'Sum of $100-$200',
      'Sum of $200-$300',
```

'Sum of \$300-\$400',  
 'Sum of \$400-\$500',  
 'Sum of \$500-\$600',  
 'Sum of \$600-\$700',  
 'Sum of \$700-\$800',  
 'Sum of \$800-\$900',  
 'Sum of \$900-\$1000',  
 'Sum of More than \$1000',  
 'Sum of Total number of delinquent residential accounts',  
 'pop',  
 'nhw',  
 'black',  
 'hisp',  
 'asian',  
 'noncitizen',  
 'immigrants',  
 'ohu',  
 'lep\_hh',  
 'dpov',  
 'npov',  
 'mhhhi',  
 'overcrowded',  
 'no\_veh\_hh',  
 'w\_broadband',  
 'pop\_19\_64',  
 'uninsured\_19\_64',  
 'pct\_nhw',  
 'pct\_black',  
 'pct\_hisp',  
 'pct\_asian',  
 'pct\_noncitizen',  
 'pct\_immigrants',  
 'pct\_lep\_hh',  
 'pct\_povt',  
 'pct\_overcrowded',  
 'pct\_no\_veh\_hh',  
 'pct\_broadband',  
 'pct\_no\_broadband',  
 'pct\_uninsured\_19\_64',  
 'aggveh',  
 'pct\_no\_hins',  
 'veh\_person',  
 'Total Population in Occupied Housing Units: Renter Occupied',  
 'Owner Occupied Pop',  
 '% Renter Pop',  
 '% Owner Pop',  
 'Households']



```
[9]: refined_columns = ['Zip Codes',
    'Sum of Less than $100',
    'Sum of $100-$200',
    'Sum of $200-$300',
    'Sum of $300-$400',
    'Sum of $400-$500',
    'Sum of $500-$600',
    'Sum of $600-$700',
    'Sum of $700-$800',
    'Sum of $800-$900',
    'Sum of $900-$1000',
    'Sum of More than $1000',
    'Sum of Total number of delinquent residential accounts',
    'pop',
    'mhhi',
    'pct_black',
    'pct_hisp',
    'pct_asian',
    'pct_povt',
    '% Renter Pop',
    '% Owner Pop',
    'Households']
```

And, let me check my work....

```
[10]: acs = acs[refined_columns]
```

```
[11]: acs.head()
```

```
[11]: Zip Codes  Sum of Less than $100  Sum of $100-$200  Sum of $200-$300  \
0      90001          5726.0          4937.0          1582.0
1      90002          3130.0          2152.0          1052.0
2      90003          1829.0          1880.0          1562.0
3      90004          2199.0          1659.0          1119.0
4      90005          1712.0          1107.0           598.0

      Sum of $300-$400  Sum of $400-$500  Sum of $500-$600  Sum of $600-$700  \
0              684.0          395.0          283.0          220.0
1              708.0          493.0          385.0          343.0
2             1169.0          941.0          782.0          673.0
3              735.0          502.0          387.0          279.0
4              401.0          281.0          175.0          146.0

      Sum of $700-$800  Sum of $800-$900  Sum of $900-$1000  \
0              137.0          132.0           97.0
1              281.0          231.0          201.0
2              513.0          482.0          401.0
```

3	223.0	206.0	164.0
4	95.0	95.0	58.0

	Sum of More than \$1000 \
0	709.0
1	1779.0
2	3437.0
3	1116.0
4	426.0

	Sum of Total number of delinquent residential accounts	pop	mhhi \
0	14902.0	58975.0	38521.0
1	10755.0	53111.0	35410.0
2	13669.0	72741.0	37226.0
3	8589.0	61586.0	48754.0
4	5094.0	39479.0	35149.0

	pct_black	pct_hisp	pct_asian	pct_povt	% Renter Pop	% Owner Pop \
0	0.088648	0.900144	0.002187	0.287523803	0.647257	0.352743
1	0.194950	0.784640	0.006063	0.3286025168	0.621999	0.378001
2	0.221828	0.770542	0.003533	0.3065973854	0.692044	0.307956
3	0.038856	0.511139	0.251437	0.1806014621	0.803900	0.196100
4	0.061374	0.492338	0.350794	0.2805929165	0.912764	0.087236

	Households
0	14174.0
1	13546.0
2	18523.0
3	25192.0
4	18291.0

#### 0.1.4 Normalize the Data

So, I'm thinking I might want to make a map that shows the intensity of water bill debt by zip code. This means I need the percentage of the delinquent population each debt "bucket" represents.

I also want to know what percent of the population in each zip code has water bill debt, broadly. To do this, I need to create new columns for each of these columns as a percentage of the delinquent population and total population, respectively.

```
[12]: list(acs)
```

```
[12]: ['Zip Codes',
      'Sum of Less than $100',
      'Sum of $100-$200',
      'Sum of $200-$300',
      'Sum of $300-$400',
      'Sum of $400-$500',
```

```

'Sum of $500-$600',
'Sum of $600-$700',
'Sum of $700-$800',
'Sum of $800-$900',
'Sum of $900-$1000',
'Sum of More than $1000',
'Sum of Total number of delinquent residential accounts',
'pop',
'mhhi',
'pct_black',
'pct_hisp',
'pct_asian',
'pct_povt',
'% Renter Pop',
'% Owner Pop',
'Households']

```

```

[13]: acs['Percent Delinquent'] = acs['Sum of Total number of delinquent residential_
      ↪accounts']/acs['pop']
      acs['Percent Less than $100'] = acs['Sum of Less than $100']/acs['Sum of Total_
      ↪number of delinquent residential accounts']
      acs['Percent $100 - $200'] = acs['Sum of $100-$200']/acs['Sum of Total number_
      ↪of delinquent residential accounts']
      acs['Percent $200 - $300'] = acs['Sum of $200-$300']/acs['Sum of Total number_
      ↪of delinquent residential accounts']
      acs['Percent $300 - $400'] = acs['Sum of $300-$400']/acs['Sum of Total number_
      ↪of delinquent residential accounts']
      acs['Percent $400 - $500'] = acs['Sum of $400-$500']/acs['Sum of Total number_
      ↪of delinquent residential accounts']
      acs['Percent $500 - $600'] = acs['Sum of $500-$600']/acs['Sum of Total number_
      ↪of delinquent residential accounts']
      acs['Percent $600 - $700'] = acs['Sum of $600-$700']/acs['Sum of Total number_
      ↪of delinquent residential accounts']
      acs['Percent $700 - $800'] = acs['Sum of $700-$800']/acs['Sum of Total number_
      ↪of delinquent residential accounts']
      acs['Percent $800 - $900'] = acs['Sum of $800-$900']/acs['Sum of Total number_
      ↪of delinquent residential accounts']
      acs['Percent $900 - $1000'] = acs['Sum of $900-$1000']/acs['Sum of Total number_
      ↪of delinquent residential accounts']
      acs['Percent More than $1000'] = acs['Sum of More than $1000']/acs['Sum of_
      ↪Total number of delinquent residential accounts']

```

And, let me check my work....

```

[14]: acs.head()

```

```

[14]: Zip Codes Sum of Less than $100 Sum of $100-$200 Sum of $200-$300 \
0 90001 5726.0 4937.0 1582.0
1 90002 3130.0 2152.0 1052.0
2 90003 1829.0 1880.0 1562.0
3 90004 2199.0 1659.0 1119.0
4 90005 1712.0 1107.0 598.0

Sum of $300-$400 Sum of $400-$500 Sum of $500-$600 Sum of $600-$700 \
0 684.0 395.0 283.0 220.0
1 708.0 493.0 385.0 343.0
2 1169.0 941.0 782.0 673.0
3 735.0 502.0 387.0 279.0
4 401.0 281.0 175.0 146.0

Sum of $700-$800 Sum of $800-$900 Sum of $900-$1000 \
0 137.0 132.0 97.0
1 281.0 231.0 201.0
2 513.0 482.0 401.0
3 223.0 206.0 164.0
4 95.0 95.0 58.0

Sum of More than $1000 \
0 709.0
1 1779.0
2 3437.0
3 1116.0
4 426.0

Sum of Total number of delinquent residential accounts pop mhhi \
0 14902.0 58975.0 38521.0
1 10755.0 53111.0 35410.0
2 13669.0 72741.0 37226.0
3 8589.0 61586.0 48754.0
4 5094.0 39479.0 35149.0

pct_black pct_hisp pct_asian pct_povt % Renter Pop % Owner Pop \
0 0.088648 0.900144 0.002187 0.287523803 0.647257 0.352743
1 0.194950 0.784640 0.006063 0.3286025168 0.621999 0.378001
2 0.221828 0.770542 0.003533 0.3065973854 0.692044 0.307956
3 0.038856 0.511139 0.251437 0.1806014621 0.803900 0.196100
4 0.061374 0.492338 0.350794 0.2805929165 0.912764 0.087236

Households Percent Delinquent Percent Less than $100 Percent $100 - $200 \
0 14174.0 0.252683 0.384244 0.331298
1 13546.0 0.202500 0.291027 0.200093
2 18523.0 0.187913 0.133806 0.137537
3 25192.0 0.139464 0.256025 0.193154

```

4	18291.0	0.129031	0.336082	0.217314
---	---------	----------	----------	----------

	Percent \$200 - \$300	Percent \$300 - \$400	Percent \$400 - \$500	\
0	0.106160	0.045900	0.026507	
1	0.097815	0.065830	0.045839	
2	0.114273	0.085522	0.068842	
3	0.130283	0.085575	0.058447	
4	0.117393	0.078720	0.055163	

	Percent \$500 - \$600	Percent \$600 - \$700	Percent \$700 - \$800	\
0	0.018991	0.014763	0.009193	
1	0.035797	0.031892	0.026127	
2	0.057210	0.049235	0.037530	
3	0.045058	0.032483	0.025963	
4	0.034354	0.028661	0.018649	

	Percent \$800 - \$900	Percent \$900 - \$1000	Percent More than \$1000
0	0.008858	0.006509	0.047578
1	0.021478	0.018689	0.165411
2	0.035262	0.029336	0.251445
3	0.023984	0.019094	0.129934
4	0.018649	0.011386	0.083628

Looks like it worked!

### 0.1.5 Summary Statistics

Now that my data is cleaned and normalized, I want to get a better sense of how it looks. I'll calculate some summary stats.

```
[15]: acs['Percent Delinquent'].describe()
```

```
[15]: count    1020.000000
      mean      0.052453
      std      0.381212
      min      0.000000
      25%      0.005332
      50%      0.017357
      75%      0.045156
      max      11.881386
      Name: Percent Delinquent, dtype: float64
```

It seems that on average, about 5% of people per zip code have some level of water bill debt, with the median hovering around 1.7%. This suggests there are some outliers on the larger end that are pulling the average up higher. There are probably some zip codes that have a very high percentage of residential accounts with debt, and these are impacting the overall average. Hopefully my later analysis can shed some light on where these higher-level debt zip codes are.

I'm going to go ahead and look at summary stats for each of the bucketed debt levels; however,

since there are so many buckets, I am going to combine them into 1 table for key stats (mean and median).

```
[16]: acs[['Percent Delinquent',  
        'Percent Less than $100',  
        'Percent $100 - $200',  
        'Percent $200 - $300',  
        'Percent $300 - $400',  
        'Percent $400 - $500',  
        'Percent $500 - $600',  
        'Percent $600 - $700',  
        'Percent $700 - $800',  
        'Percent $800 - $900',  
        'Percent $900 - $1000',  
        'Percent More than $1000']].mean()
```

```
[16]: Percent Delinquent          0.052453  
      Percent Less than $100    0.269824  
      Percent $100 - $200      0.241373  
      Percent $200 - $300      0.142088  
      Percent $300 - $400      0.098082  
      Percent $400 - $500      0.230112  
      Percent $500 - $600      0.045637  
      Percent $600 - $700      0.036646  
      Percent $700 - $800      0.027409  
      Percent $800 - $900      0.022396  
      Percent $900 - $1000     0.018324  
      Percent More than $1000  0.113338  
      dtype: float64
```

```
[17]: acs[['Percent Delinquent',  
        'Percent Less than $100',  
        'Percent $100 - $200',  
        'Percent $200 - $300',  
        'Percent $300 - $400',  
        'Percent $400 - $500',  
        'Percent $500 - $600',  
        'Percent $600 - $700',  
        'Percent $700 - $800',  
        'Percent $800 - $900',  
        'Percent $900 - $1000',  
        'Percent More than $1000']].median()
```

```
[17]: Percent Delinquent          0.017357  
      Percent Less than $100    0.233190  
      Percent $100 - $200      0.214128  
      Percent $200 - $300      0.121451
```

```

Percent $300 - $400      0.078720
Percent $400 - $500      0.053389
Percent $500 - $600      0.038719
Percent $600 - $700      0.029613
Percent $700 - $800      0.020958
Percent $800 - $900      0.016476
Percent $900 - $1000     0.011659
Percent More than $1000  0.054545
dtype: float64

```

So, this is a lot of information, but I have noticed some interesting trends...

So, the bucket with the greatest percentage of delinquent accounts, on average per zip code, was less than 100 (at 23.3%). The next highest bucket was between 100 and 200, at 21.4%. This means that, on average, across zip codes, most debt-holders had debt on the lower end (less than \$200).

However, we can also tell from the mean vs median values that there is a slight skew in the percentages themselves. It is also worth mentioning that there are especially high outliers in the 400-500 bucket (with the mean at 23 percent and the median at 5.3 percent). Overall, this suggests that there are certain zip codes wherein these values make up a higher percentage of the total debt.

More broadly, in the context of the 'Percent Delinquent' bucket, this skew means that, there are certain zip codes within each bucket that might act as outliers that make up especially large percentages of water bill debt. This reflects unequal distribution of debt across zip codes. This may be due to other factors such as median household income, race and ethnicity, or owner v renter status. We may get a sense of where these zip codes lie as we further map the data.

How many total households have some level of water bill debt, though?

```

[18]: acs['Sum of Total number of delinquent residential accounts'].sum(axis = 0,
↳ skipna = True)

```

```

[18]: 2897402.00000002

```

In total, 2,897,402 households in the state (as reported in the survey) have some level of water debt, and based on my summary stats, it seems that the values are fairly evenly distributed across the state.

### 0.1.6 Data Visualization

I think it will be a little bit easier to visualize what is happening with the debt data if I put it in a histogram. First, I am going to do that with the total percentage of delinquent accounts.

```

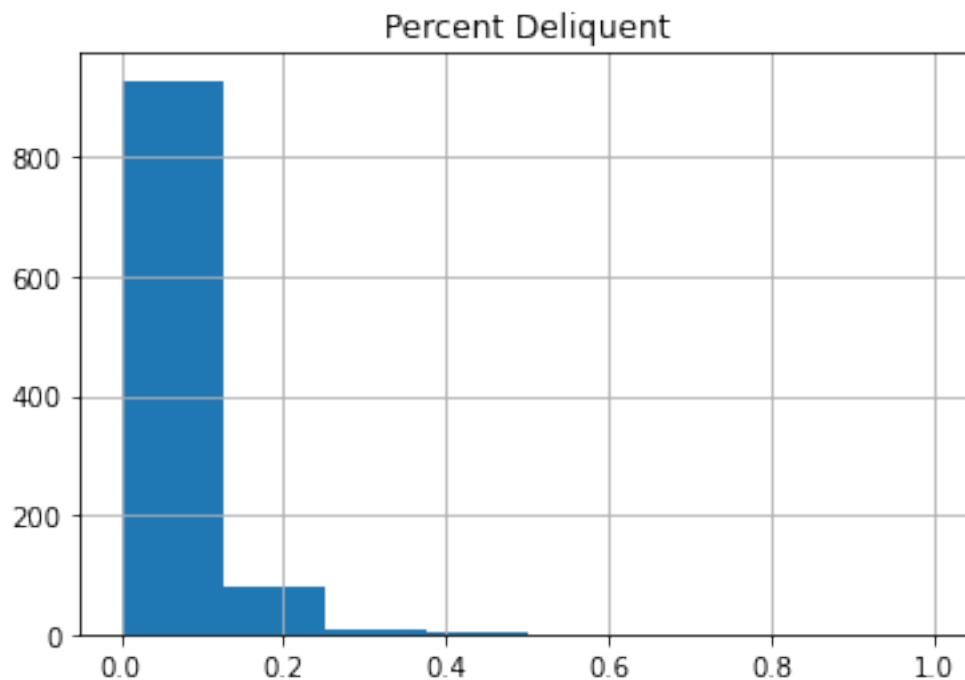
[19]: acs.hist(column = 'Percent Delinquent', bins = 8, range=(0, 1))

```

```

[19]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd229101ca0>]],
dtype=object)

```

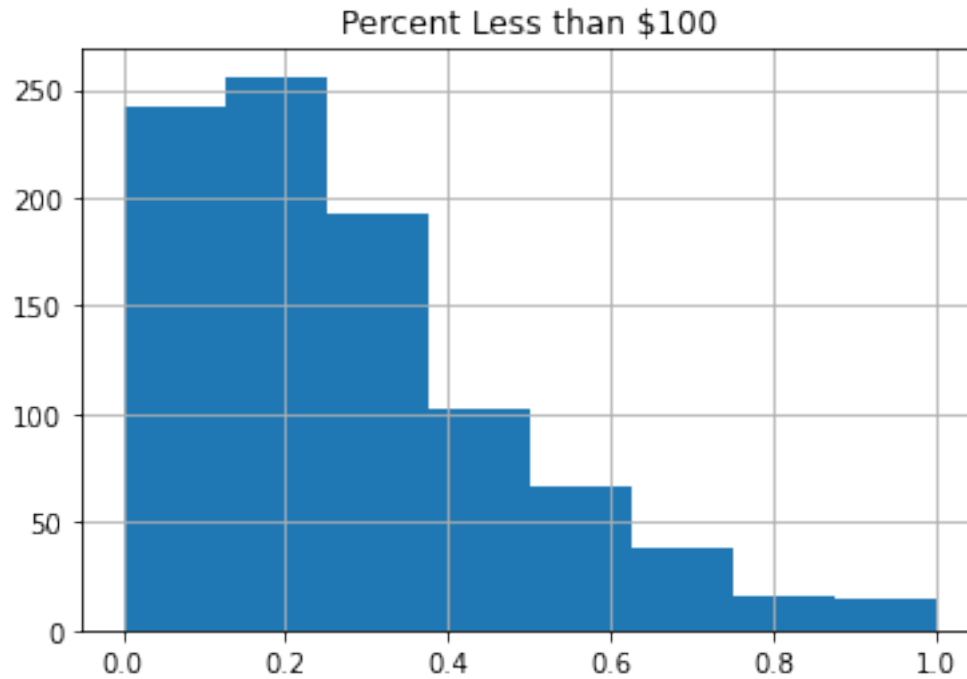


As expected from my summary stats, we can see the vast majority of %Delinquent accounts are centered at 1% of the population. However, there are a few outliers closer to 5%, and it will be interesting to see if there is any relationship with other demographic factors once we get into mapping.

```
[20]: acs.hist(column = 'Percent Less than $100', bins = 8, range=(0, 1))
```

```
[20]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd2291f5580>]],  
        dtype=object)
```



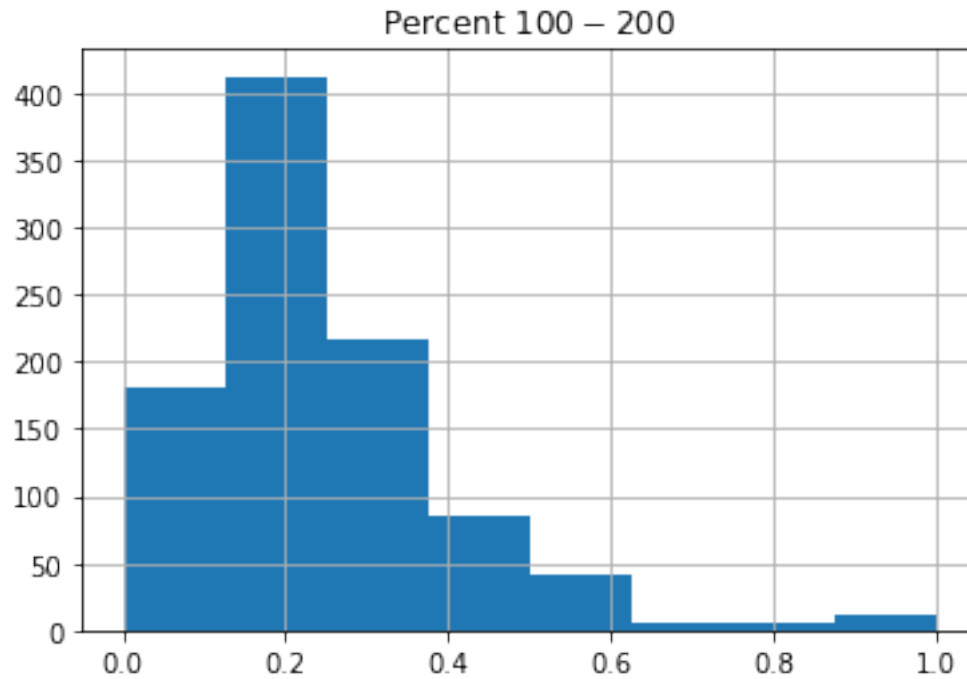


The distribution of Percent Less than \$100 is much more pronounced. From this histogram, you can see that the percentage of delinquent accounts (whose debt is less than 100) varies across zip codes: it can make up anywhere from 0 to 100 percent of the delinquent accounts across zip codes.

Let's look at the distribution across the other buckets, just to get a sense of their spread.

```
[21]: acs.hist(column = 'Percent $100 - $200', bins = 8, range=(0, 1))
```

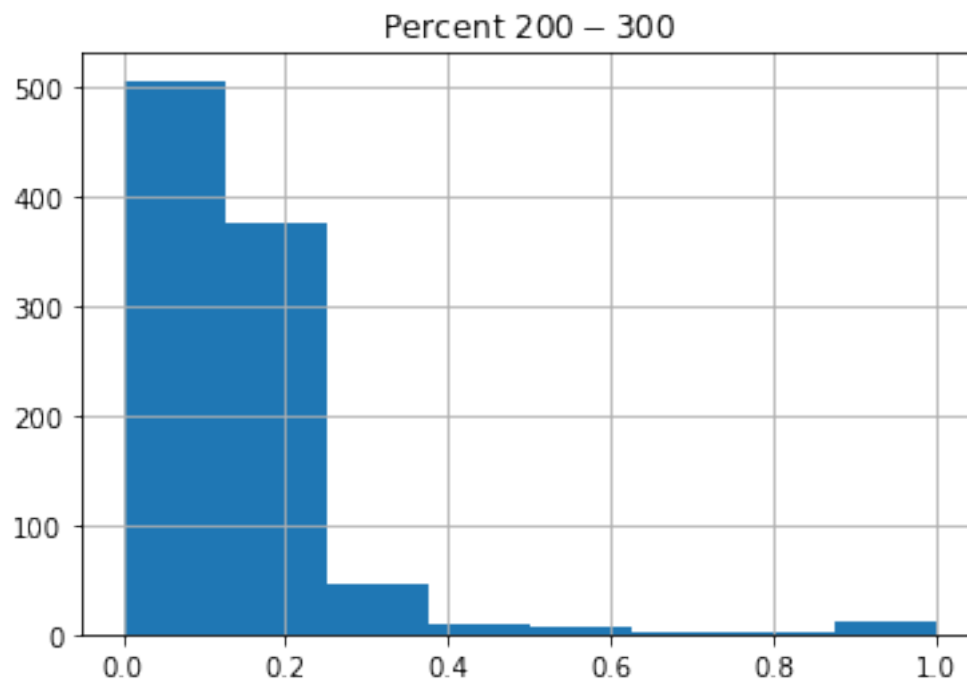
```
[21]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd229078eb0>]],  
      dtype=object)
```



There is a large peak with this bucket around 20%, which is reflected in the median and the mean.

```
[22]: acs.hist(column = 'Percent $200 - $300', bins = 8, range=(0, 1))
```

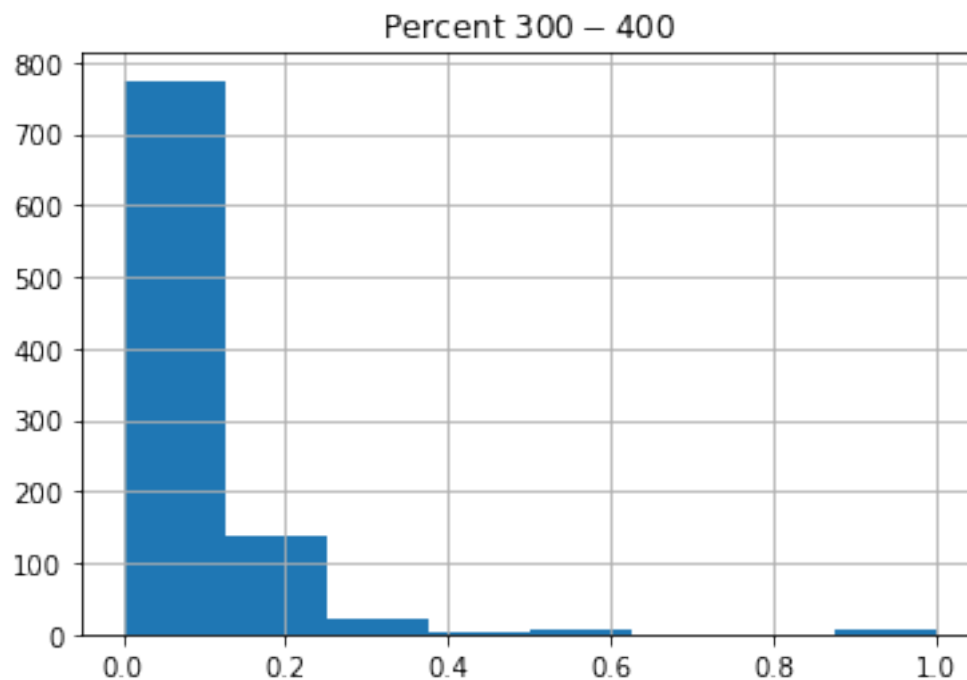
```
[22]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd22877fb80>]],  
      dtype=object)
```



There is a skew in this bucket between 0 - 10%

```
[23]: acs.hist(column = 'Percent $300 - $400', bins = 8, range=(0, 1))
```

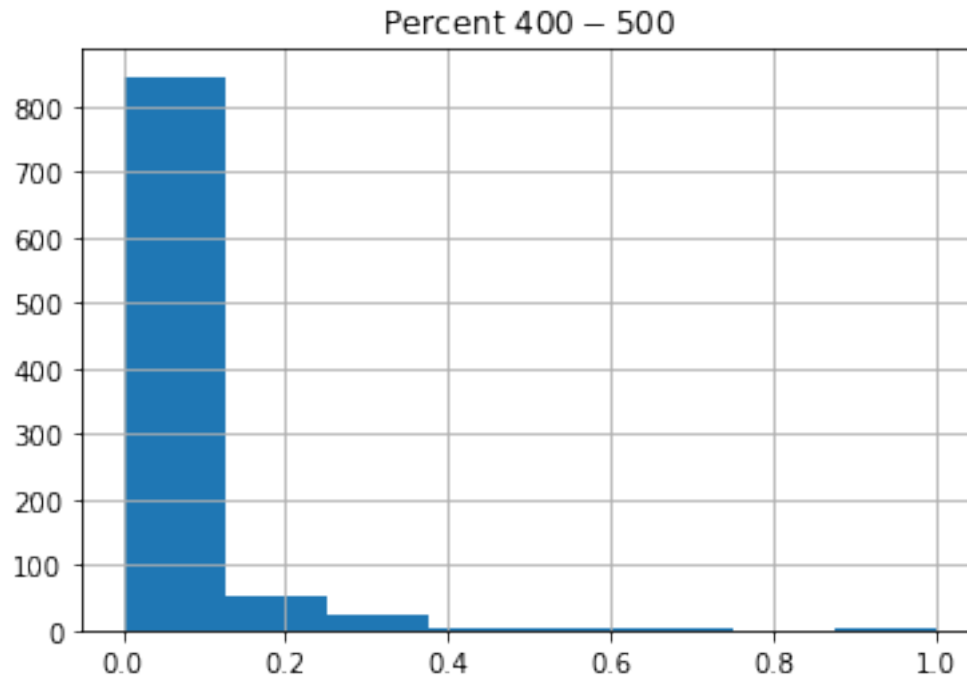
```
[23]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd2291ac460>]],  
        dtype=object)
```



Again, a skew here between 0-10%

```
[24]: acs.hist(column = 'Percent $400 - $500', bins = 8, range=(0, 1))
```

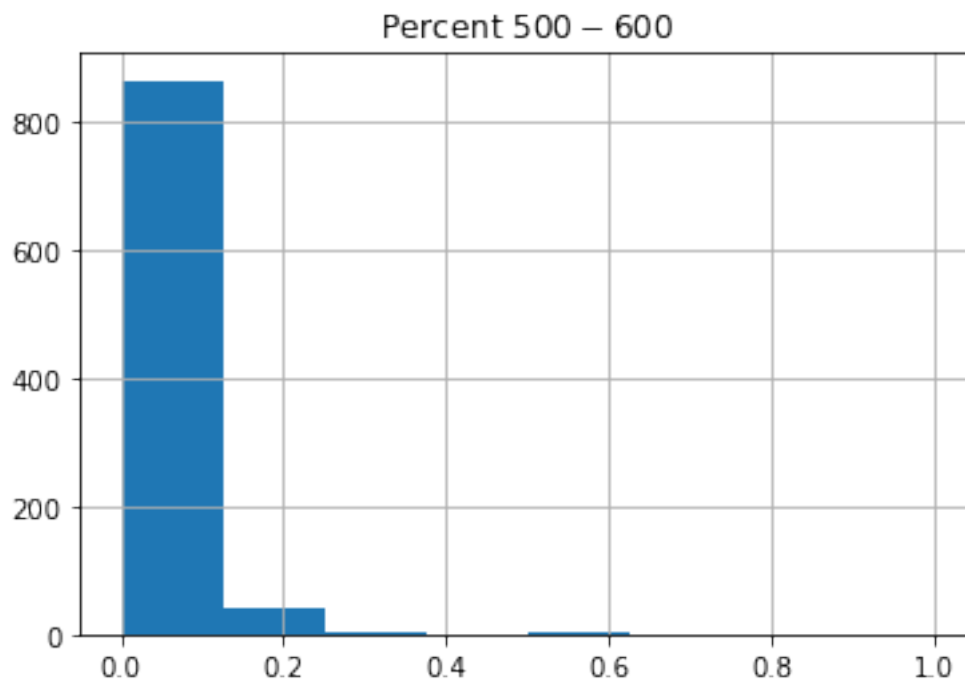
```
[24]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd2286baaf0>]],  
      dtype=object)
```



Similar skew among this data... with a few outliers around 60-70% and 90-100%

```
[25]: acs.hist(column = 'Percent $500 - $600', bins = 8, range=(0, 1))
```

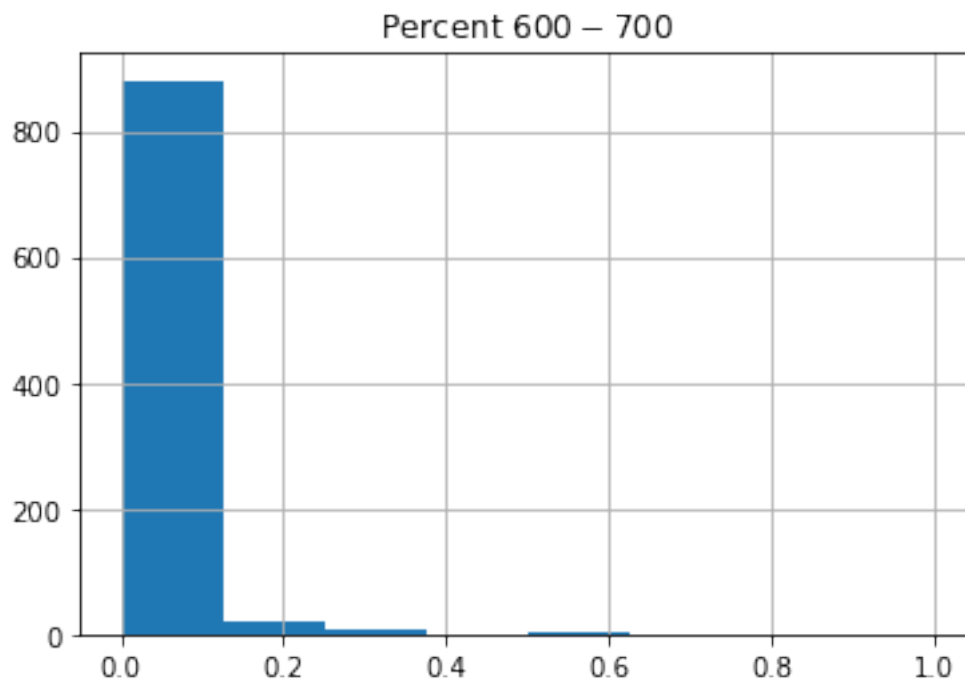
```
[25]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd228681160>]],  
      dtype=object)
```



Similar skew here as well, which is to be expected given the median/mode values.

```
[26]: acs.hist(column = 'Percent $600 - $700', bins = 8, range=(0, 1))
```

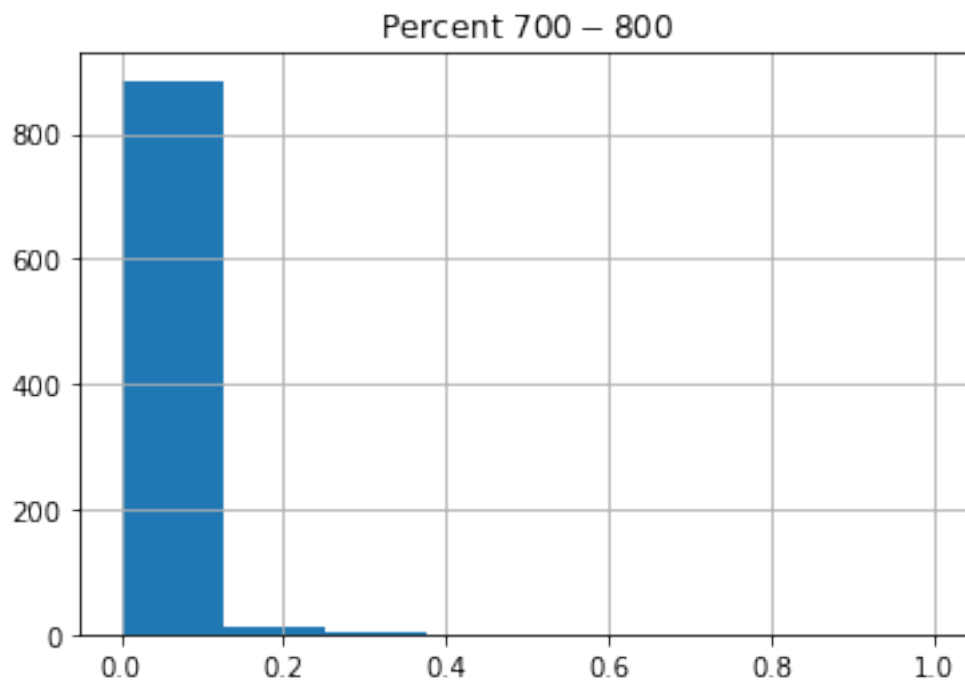
```
[26]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd228643b20>]],  
        dtype=object)
```



And again, similar skew here as well, which is to be expected given the median/mode values.

```
[27]: acs.hist(column = 'Percent $700 - $800', bins = 8, range=(0, 1))
```

```
[27]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd2285f38e0>]],  
        dtype=object)
```

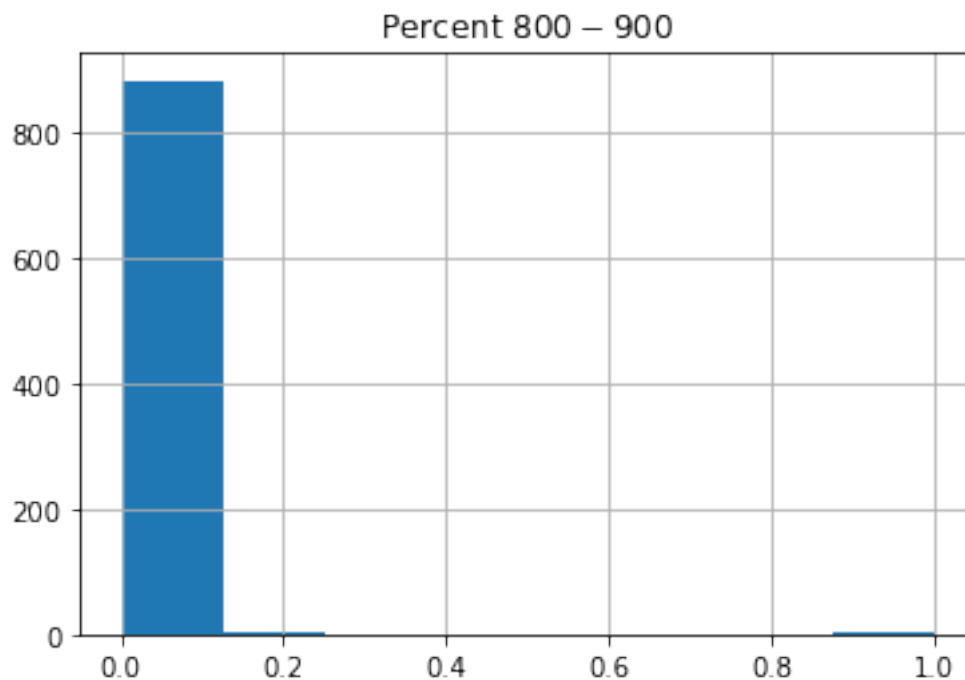


Again, a similar skew here.

```
[28]: acs.hist(column = 'Percent $800 - $900', bins = 8, range=(0, 1))
```

```
[28]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd2286657f0>]],  
        dtype=object)
```

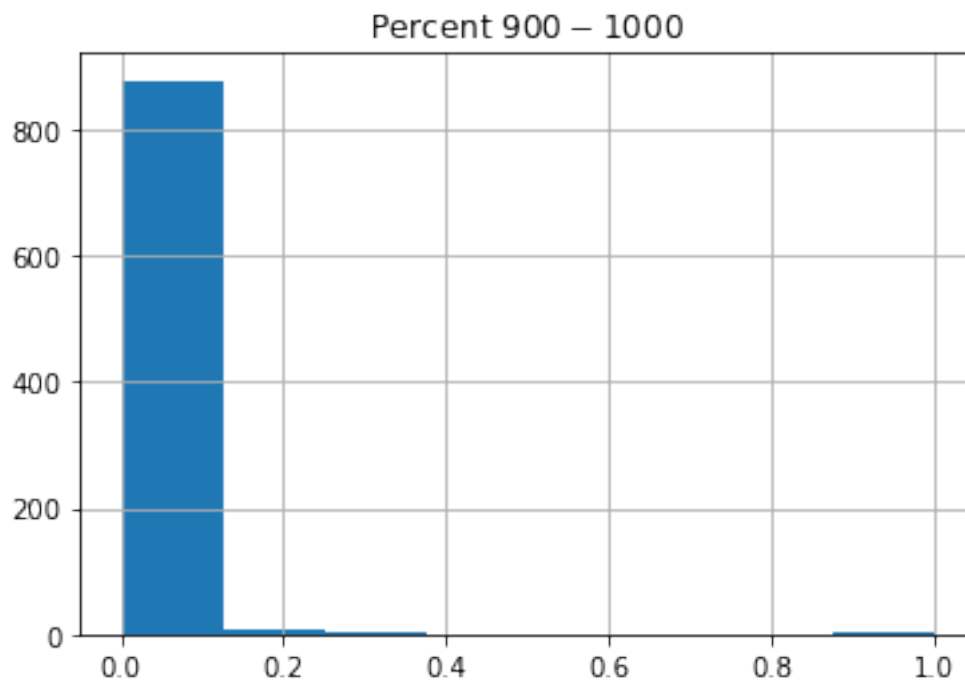




..and here. Though there is an interesting outlier near 90-100%

```
[29]: acs.hist(column = 'Percent $900 - $1000', bins = 8, range=(0, 1))
```

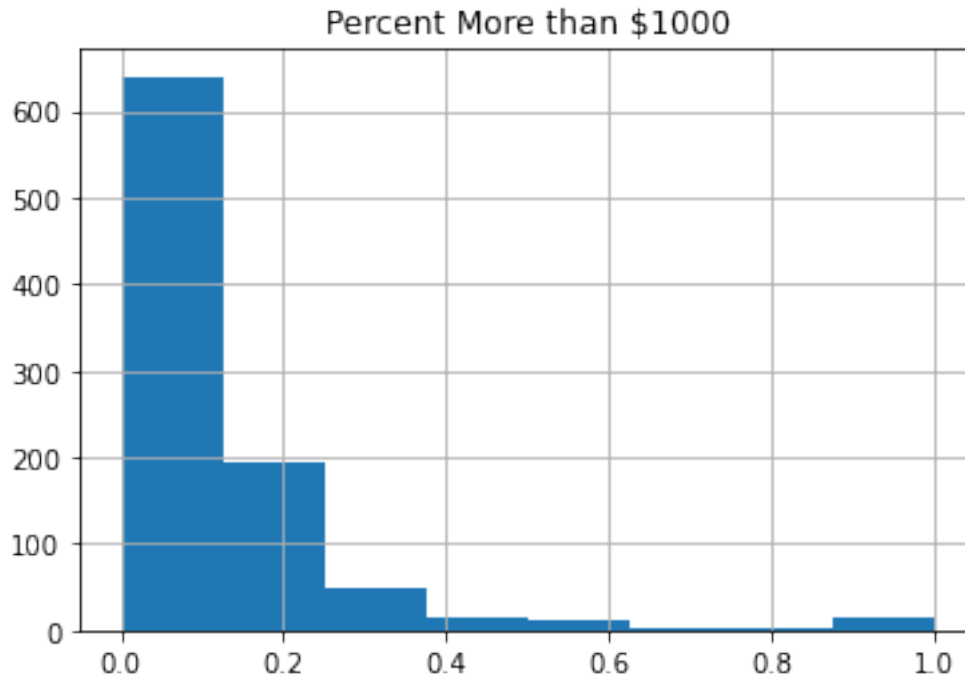
```
[29]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd228585550>]],  
      dtype=object)
```



And here... with a notable outlier near 90-100%. This might be interesting if it comes up during the mapping.

```
[30]: acs.hist(column = 'Percent More than $1000', bins = 8, range=(0, 1))
```

```
[30]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd22855ff70>]],  
        dtype=object)
```



There is a much larger spread in this data, and again, with some outliers near 90-100% which is quite concerning.

Essentially, we can see from all of these histograms that, for the most part, the debt buckets are centered near 0-10 percent and some peaks around 20%. There are, however, some outliers around 8-100 percent. This suggests that in some zip codes all of the household debt is contained within that debt bucket. For some of the higher buckets (i.e. greater than 1000, that is especially concerning).

### 0.1.7 Incorporation of Demographics

Now that we have a sense of how the debt is distributed, it is time to incorporate some demographic information in the form of scatterplots. The purpose here is to detect some trends in the data between the total percentage of delinquent households in a given zip code and our demographic factors of interest (median household income, %hispanic, %black, or percent poverty)

```
[31]: px.scatter(acs,
          x = 'mhhhi',
          y = 'Percent Delinquent')
```

We can see that many of the delinquent households have a median household income between 20,000 and 100,000. As income increases, we can tell that the percentage of delinquent households becomes much more sparse and in all of these higher income cases, the percentage of delinquent households is extremely small (centered around 0).

Notably, there are some outliers in this dataset. In particular, the dot at 12 on the y axis and at 2. I think these datapoints should be removed, as they are clearly mistakes since they equal over

100%.

```
[32]: acs.drop(acs[acs['Percent Deliquent'] > 1].index, inplace = True)
```

```
[33]: px.scatter(acs,  
               x = 'mghi',  
               y = 'Percent Deliquent')
```

Now we can get a much better sense of the trend in the data - there is a very high concentration of zip codes with some degree of water bill debt between 30,000 and 100,000. This is a clear trend in the data, wherein lower income zip codes tend to have more water bill debt than higher income zip codes.

Now let's see if there are trends in the other demographic factors.

```
[34]: px.scatter(acs,  
               x = 'pct_povt',  
               y = 'Percent Deliquent')
```

Looking at the percentage of poverty, there is not necessarily a clear trend. There are definitely a high concentration of delinquent households in zip codes with poverty levels between 10-20%. However, when we dive deeper into this data, there might be a clearer trend.

The poverty rate in the US is around 12%, so if we consider anything higher than 12% as a high poverty area, there is some indication that there are a high concentration of zip codes with some degree of delinquency on their bills in moderately-high poverty rate zip codes. Of course, there needs to be a bit more examination of the data before we make a clearer judgement call.

```
[35]: px.scatter(acs,  
               x = 'pct_black',  
               y = 'Percent Deliquent')
```

There is a very slight trend here, but not a very obvious one. There are some zip codes with a higher percentage of Black populations that have higher bill debt (as you can see on the right hand side of the scatterplot). However, more analysis is needed.

```
[36]: px.scatter(acs,  
               x = 'pct_hisp',  
               y = 'Percent Deliquent')
```

It definitely seems like there are many more datapoints with a larger hispanic population than black population. It also does appear that populations with a greater percentage of hispanic populations also have fairly high levels of water bill debt delinquency (when compared to black populations) I think more analysis should be done, but there seems to be a clearer trend in this data than with the % black population.

Since I am running out of memory, I will continue in a new notebook with my maps.