Project 1 - California Water Usage

Welcome to the first project in CS260! We will be exploring possible connections between water usage, geography, and income in California. The water data for this project was procured from the California State Water Resources Control Board and curated by the Pacific Institute. The map data includes US topography, California counties, and ZIP codes.

The dataset on income comes from the IRS (documentation). We have identified some interesting columns in the dataset, but a full description of all the columns (and a definition of the population in the dataset and some interesting anonymization procedures they used) is available in this description.

As usual, **run the cell below** to prepare the automatic tests. **Passing the automatic tests does not guarantee full credit on any question.** The tests are provided to help catch some common errors, but it is *your* responsibility to answer the questions correctly.

In [1]:

```
# Run this cell, but please don't change it.

import numpy as np
import math
from datascience import *

# These lines set up the plotting functionality and formatting.
import matplotlib
# matplotlib.use('Agg', warn=False)
%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')

# These lines load the tests.
from client.api.assignment import load_assignment
project1 = load_assignment('project1.ok')
```

```
=====================================================================
Assignment: Project 1
OK, version v1.18.1
=====================================================================
```

First, load the data. Loading may take some time.

In [2]:

```
# Run this cell, but please don't change it.

districts = Map.read_geojson('water_districts.geojson')
zips = Map.read_geojson('ca_zips.geojson.gz')
usage_raw = Table.read_table('water_usage.csv', dtype={'pwsid': str})
income_raw = Table.read_table('ca_income_by_zip.csv', dtype={'ZIP': str}).drop(['STATEF
wd_vs_zip = Table.read_table('wd_vs_zip.csv', dtype={'PWSID': str, 'ZIP': str}).set_for
```

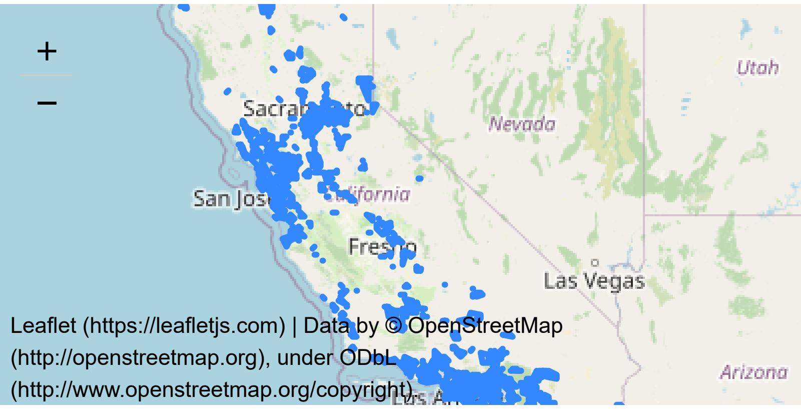
Part 0: Maps

The `districts` and `zips` data sets are `Map` objects. Documentation on mapping in the `datascience` package can be found at data8.org/datascience/maps.html. To view a map of California's water districts, run the cell below. Click on a district to see its description.

In [3]:

```
districts.format(width=400, height=200)
```

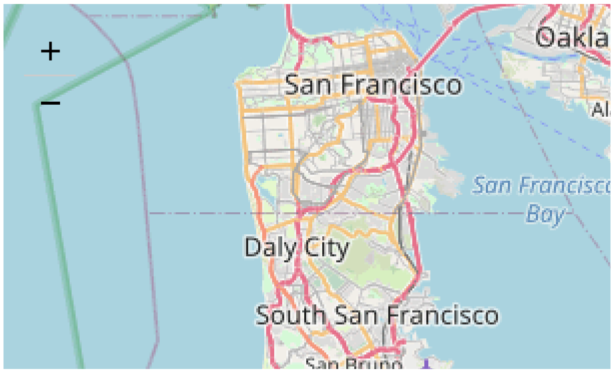
Out[3]:

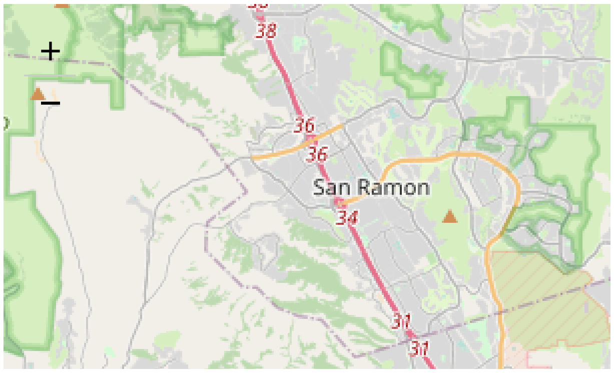
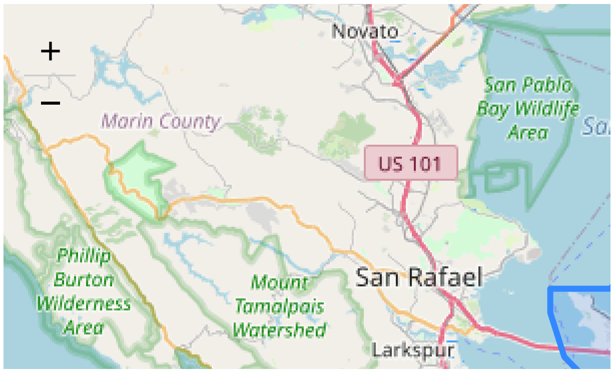

+
−
Sacramento
Nevada
Utah
San Jose
California
Fresno
Las Vegas
Arizona
Los An
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

A `Map` is a collection of regions and other features such as points and markers, each of which has a **string** `id` and various properties. You can view the features of the `districts` map as a table using `Table.from_records`.

In [4]:

```
district_table = Table.from_records(districts.features)
district_table.show(3)
```

PWSID	feature	id	popupContent
0110001	 + − Oakla San Francisco Al San Francisco Bay Daly City South San Francisco San Bruno	0	Alameda County Water District

PWSID	feature	id	popupContent
0110003		1	California Water Service Company Livermore
0110005		2	East Bay Municipal Utilities District

... (407 rows omitted)

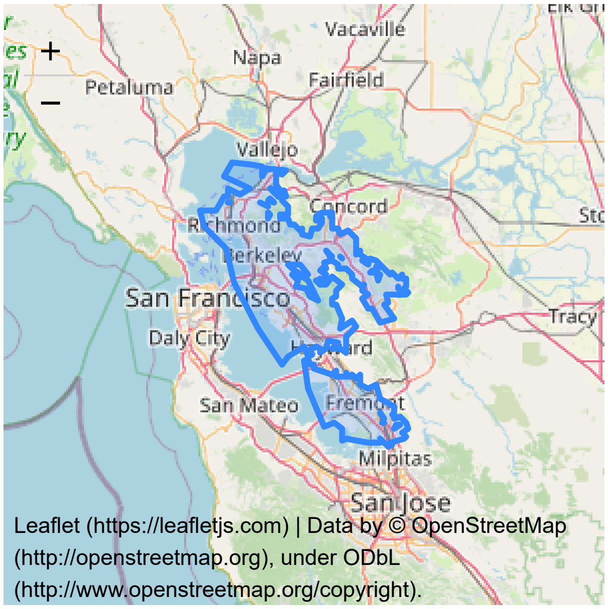
To display a `Map` containing only two features from the `district_table`, call `Map` on a list containing those two features from the `feature` column.

Question 0.0 Draw a map of the Alameda County Water District (row 0) and the East Bay Municipal Utilities District (row 2).

In [5]:

```
alameda_and_east_bay = district_table.take(0,2).column("feature")
Map(alameda_and_east_bay, height=300, width=300)
```

Out[5]:


Vacaville
Napa
Petaluma
Fairfield
Vallejo
Concord
Richmond
Berkeley
San Francisco
Daly City
Hayward
Tracy
San Mateo
Fremont
Milpitas
San Jose
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

Trust Notebook

In [6]:

```
_ = project1.grade('q00')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed
```

Hint: If scrolling becomes slow on your computer, you can clear maps for the cells above by running `Cell > All Output > Clear` from the `Cell` menu.

Part 1: California Income

Let's look at the `income_raw` table.

In [7]:

```
income_raw
```

Out[7]:

ZIP	N1	MARS1	MARS2	MARS4	PREP	N2	NUMDEP	A00100	N02650	A02650	N00200
90001	13100	6900	1890	4270	10740	29670	15200	181693	13100	184344	10220
90001	5900	1700	1970	2210	4960	17550	9690	203628	5900	204512	5610
90001	1480	330	760	390	1240	4710	2470	89065	1480	89344	1440
90001	330	50	210	70	290	1100	560	28395	330	28555	320
90001	160	30	100	40	130	510	250	24676	160	25017	150
90001	0	0	0	0	0	0	0	0	0	0	0
90002	12150	6330	1460	4330	9580	27240	14070	167261	12150	170095	9440
90002	5030	1510	1490	1980	4120	14410	7890	173280	5030	174335	4760
90002	1320	300	600	400	1060	4090	2180	78559	1320	78871	1270
90002	340	90	190	90	270	1060	530	28502	340	28558	320

... (8888 rows omitted)

Some observations:

1. The table contains several numerical columns and a column for the ZIP code.
2. For each ZIP code, there are 6 rows. Each row for a ZIP code has data from tax returns in one *income bracket* -- a group of people who make between some income and some other income.
3. According to the IRS documentation, all the numerical columns are *totals* -- either total numbers of returns that fall into various categories, or total amounts of money (in thousands of

dollars) from returns in those categories. For example, the column `'N02650'` is the number of returns that included a total income amount, and `'A02650'` is the total amount of total income (in thousands of dollars) from those returns.

Question 1.0. Since we don't care about income brackets, but we do care about totals per ZIP code, let's group together our income data by ZIP code. Assign the name `income_by_zipcode` to a table with just one row per ZIP code. When you group according to ZIP code, the remaining columns should be summed. In other words, for any other column such as `'N02650'`, the value of `'N02650'` in a row corresponding to ZIP code 90210 (for example) should be the sum of the values of `'N02650'` in the 6 rows of `income_raw` corresponding to ZIP code 90210.

In [8]:

```
income_by_zipcode = income_raw.group("ZIP",sum)
income_by_zipcode
```

Out[8]:

ZIP	N1	MARS1	MARS2	MARS4	PREP	N2	NUMDEP	A00100	N02650	A02650	I
	sum	sum	sum	sum	sum	sum	sum	sum	sum	sum	
90001	20970	9010	4930	6980	17360	53540	28170	527457	20970	531772	
90002	18960	8230	3830	6800	15120	47200	24850	462823	18960	467128	
90003	26180	11310	5130	9640	20570	64470	33760	612733	26180	618848	
90004	27360	15330	7000	4670	20260	51180	17800	1.61777e+06	27360	1.64943e+06	
90005	15430	8550	3870	2830	11210	29910	11130	707020	15430	717290	
90006	22630	11470	5400	5630	17840	47590	20210	563530	22630	571157	
90007	11710	6350	2270	3020	8310	23380	9950	311779	11710	315581	
90008	14710	8060	2310	4110	9990	27000	10310	662036	14710	668523	
90010	2210	1270	690	210	1760	3790	960	314333	2210	320471	
90011	36670	15540	8600	12390	30240	95640	51260	857731	36670	864961	

... (1473 rows omitted)

In [9]:

```
_ = project1.grade('q10')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 1.1. Relabel the columns in `income_by_zipcode` to match the labels in `income_raw`; you probably modified all the names slightly in the previous question.

Hint: Inspect `income_raw.labels` and `income_by_zipcode.labels` to find the differences you need to change.

Hint 2: Since there are many columns, it will be easier to relabel each of them by using a `for` statement. See Section 3.2 of the textbook for details.

Hint 3: You can use the `replace` method of a string to remove excess content. See lab02 for examples.

Hint 4: To create a new table from an existing table with one label replaced, use `relabeled`. To **change** a label in an existing table permanently, use `relabel`. Both methods take two arguments: the old label and the new label. You can solve this problem with either one, but `relabel` is simpler.

In [10]:

```
for label in income_by_zipcode:
    income_by_zipcode.relabel(label, label.strip(" sum"))
income_by_zipcode
```

Out[10]:

ZIP	N1	MARS1	MARS2	MARS4	PREP	N2	NUMDEP	A00100	N02650	A02650
90001	20970	9010	4930	6980	17360	53540	28170	527457	20970	531772
90002	18960	8230	3830	6800	15120	47200	24850	462823	18960	467128
90003	26180	11310	5130	9640	20570	64470	33760	612733	26180	618848
90004	27360	15330	7000	4670	20260	51180	17800	1.61777e+06	27360	1.64943e+06
90005	15430	8550	3870	2830	11210	29910	11130	707020	15430	717290
90006	22630	11470	5400	5630	17840	47590	20210	563530	22630	571157
90007	11710	6350	2270	3020	8310	23380	9950	311779	11710	315581
90008	14710	8060	2310	4110	9990	27000	10310	662036	14710	668523
90010	2210	1270	690	210	1760	3790	960	314333	2210	320471
90011	36670	15540	8600	12390	30240	95640	51260	857731	36670	864961

... (1473 rows omitted)

In [11]:

```
_ = project1.grade('q11')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 1.2. Create a table called `income` with one row per ZIP code and the following columns.

1. A `ZIP` column with the same contents as `'ZIP'` from `income_by_zipcode` .
2. A `returns` column containing the total number of tax returns that include a total income amount (column `'N02650'` from `income_by_zipcode`).
3. A `total` column containing the total income in all tax returns in thousands of dollars (column `'A02650'` from `income_by_zipcode`).
4. A `farmers` column containing the number of farmer returns (column `'SCHF'` from `income_by_zipcode`).

In [12]:

```
income = Table().with_columns([
        'ZIP', income_by_zipcode.column('ZIP'),
        'returns', income_by_zipcode.column('N02650'),
        'total', income_by_zipcode.column('A02650'),
        'farmers', income_by_zipcode.column('SCHF')
        ])
income.set_format('total', NumberFormatter(0)).show(5)
```

ZIP	returns	total	farmers
90001	20970	531,772	0
90002	18960	467,128	0
90003	26180	618,848	0
90004	27360	1,649,431	0
90005	15430	717,290	0

... (1478 rows omitted)

In [13]:

```
_ = project1.grade('q12')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Question > Suite 1 > Case 2

>>> print(income.take([0, 1, 2]))
ZIP   | returns | total   | farmers
90001 | 20970   | 531,772 | 0
90002 | 18960   | 467,128 | 0
90003 | 26180   | 618,848 | 0

# Error: expected
#     ZIP   | returns | total        | farmers
#     90001 | 20970   | 531,772      | 0
#     90002 | 18960   | 467,128      | 0
#     90003 | 26180   | 618,848      | 0
# but got
#     ZIP   | returns | total   | farmers
#     90001 | 20970   | 531,772 | 0
#     90002 | 18960   | 467,128 | 0
#     90003 | 26180   | 618,848 | 0

Run only this test case with "python3 ok -q q12 --suite 1 --case 2"
---------------------------------------------------------------------
```

```
Test summary
    Passed: 1
    Failed: 1
[ooook.....] 50.0% passed
```

Question 1.3. What is the average total income reported on all California tax returns that include a total income amount? **Express the answer in *dollars* as an `int` rounded to the nearest dollar.**

In [14]:

```
average_income = int(np.average(income.column("total")))
average_income
```

Out[14]:

```
828283
```

Question 1.4. All ZIP codes with less than 100 returns (or some other special conditions) are grouped together into one ZIP code with a special code. Remove the row for that ZIP code from the `income` table. *Hint:* This ZIP code value has far more returns than any of the other ZIP codes.

Hint: To **remove** a row in the `income` table using `where` , assign `income` to the smaller table using the following expression structure:

```
income = income.where(...)
```

Hint 2: Each ZIP code is represented as a string, not an `int` .

In [15]:

```
income = income.where("ZIP", are.not_equal_to('99999'))
```

In [16]:

```
_ = project1.grade('q14')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 1.5. Among the tax returns in California for ZIP codes represented in the `incomes` table, is there an association between income and living in a ZIP code with a higher-than-average proportion of farmers?

Answer the question by comparing the average incomes for two groups of *tax returns*: those in ZIP codes with a greater-than-average proportion of farmers and those in ZIP codes with a less-than-average (or average) proportion. Make sure both of these values are displayed (preferably in a table). *Then, describe your findings.*

In [17]:

```
# Build and display a table with two rows:
#   1) incomes of returns in ZIP codes with a greater-than-average proportion of farmer
#   2) incomes of returns in other ZIP codes
```

```
farm_proportion = income.column('farmers')/income.column('returns')
avg_farm_proportion = np.average(farm_proportion)
fincome = income.with_columns('farm_proportion', farm_proportion)
fincome1= fincome.where("farm_proportion", are.above(avg_farm_proportion))
fincome_avg1 = (np.average(fincome1.column('total')/fincome1.column("returns"))*1000)
fincome2= fincome.where('farm_proportion', are.below_or_equal_to(avg_farm_proportion))
fincome_avg2 = np.average((fincome2.column('total')/fincome2.column('returns'))*1000)
income_by_farm_proportion = Table().with_columns('Averages', make_array(fincome_avg1,
                                                                         fincome_avg2))
income_by_farm_proportion
```

Out[17]:

Averages
60225.7
79038.3

There is a 13.5% disparity between incomes of returns in ZIP codes with a greater-than-average proportion of farmers vs. incomes of returns in other ZIP codes. This can be contributed to other skill professionals that are included in the greater-than-average proportion of farmers.

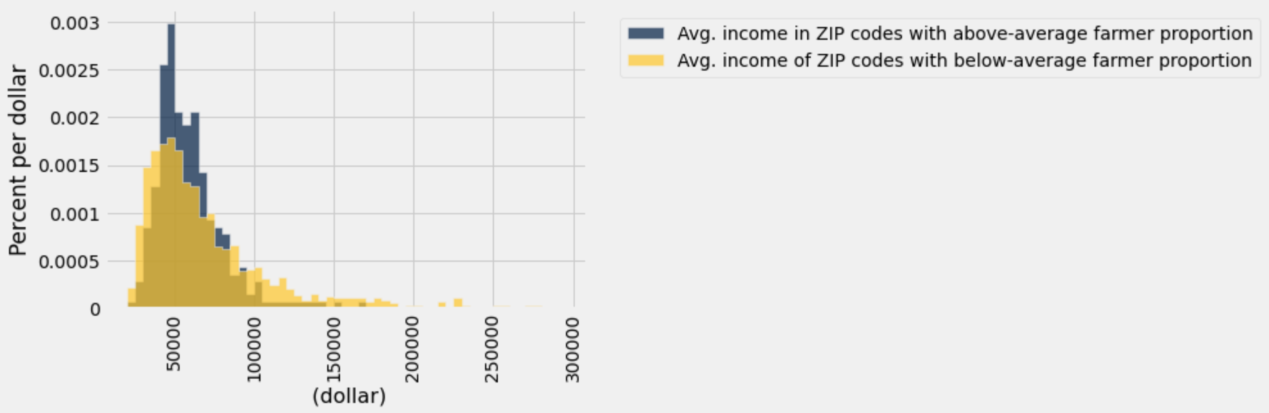
Question 1.6. Investigate the same question by comparing two histograms: the average incomes of ZIP codes that have above-average vs below-average proportions of farmers. Quantify and describe the difference in the standard deviations of average incomes for the two kinds of ZIP codes.

In [18]:

```
# You do not need to change this cell; just look at the chart it generates.

bins = np.arange(20000, 300000, 5000)
avg_income = 1000 * income.column('total')/income.column('returns')
binned_incomes=Table()\
    .with_column('income', avg_income)\
    .where(farm_proportion > np.mean(farm_proportion))\
    .bin(bins=bins).relabeled(1, 'Avg. income in ZIP codes with above-average farmer pr
    .join('bin', Table().with_column('income', avg_income)\
        .where(farm_proportion <= np.mean(farm_proportion))\
        .bin(bins=bins).relabeled(1, 'Avg. income of ZIP codes with below-average farme
binned_incomes.hist(counts='bin', bins=bins, unit='dollar')
```

```
C:\Anaconda\lib\site-packages\datascience\tables.py:4732: UserWarning: counts arg of his
t is deprecated; use bin_column
  warnings.warn("counts arg of hist is deprecated; use bin_column")
C:\Anaconda\lib\site-packages\datascience\tables.py:5206: UserWarning: FixedFormatter sh
ould only be used together with FixedLocator
  axis.set_xticklabels(ticks, rotation='vertical')
```



In [19]:

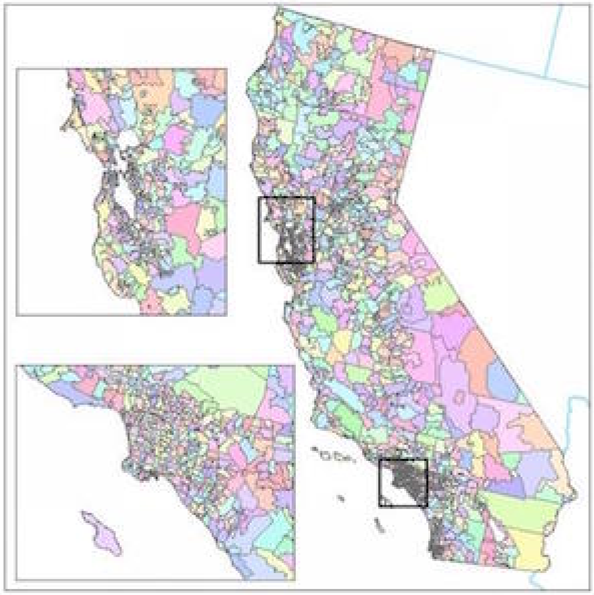
```
# Compute and compare the spread of both distributions
SD_1 = np.std(binned_incomes.column(
    "Avg. income in ZIP codes with above-average farmer proportion"))
SD_2 = np.std(binned_incomes.column(
    "Avg. income of ZIP codes with below-average farmer proportion"))
Standard_Deviation = Table().with_columns("Standard_Deviation", make_array(SD_1,SD_2))
Standard_Deviation
```

Out[19]:

Standard_Deviation
9.88413
30.3214

The standard deviation (SD) for above and below average farmer proportion follows a similar theme in the explanation for the disparity between incomes. The bell shape curve/distribution for above average farmer proportion is narrower (less of a spread) vs. the below average farmer proportion, where the spread is wider. We can see these points by looking at the horizontal axis above. In addition, the population also should be factored into the explanation. We can also see from the zip codes coverage that certain regions in California are heavily populated (e.g., LA and the Bay). There are higher income earners in these areas but with the increase population contributes to the spread for above average. However, the remaining regions in California are less populated with potentially more farmers with lower income.

ZIP codes cover all the land in California and do not overlap. Here's a map of all of them.



Question 1.7. Among the ZIP codes represented in the `incomes` table, is there an association between high average income and some aspect of the ZIP code's location? If so, describe one aspect of the location that is clearly associated with high income.

Answer the question by drawing a map of all ZIP codes that have an average income above 100,000 dollars. *Then, describe an association that you observe.*

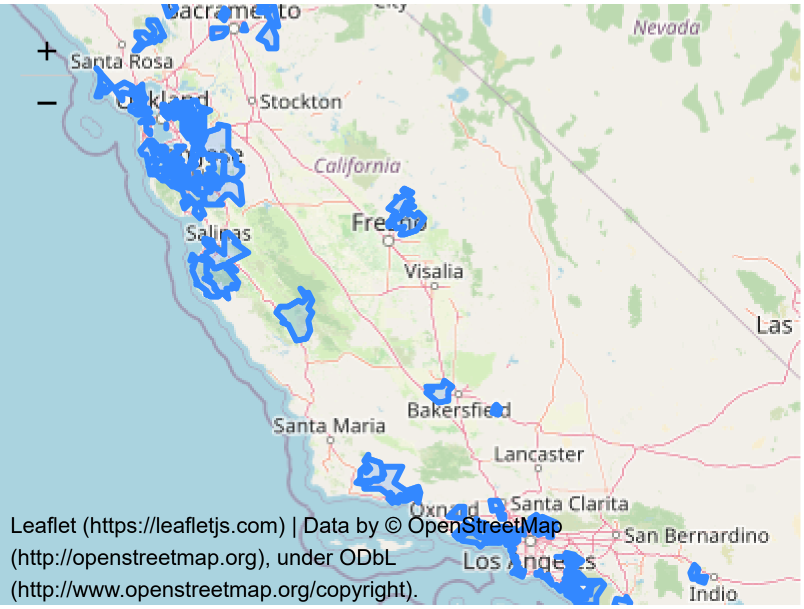
In order to create a map of certain ZIP codes, you need to

- Construct a table containing only the ZIP codes of interest, called `high_average_zips` ,
- Join `high_average_zips` with the `zip_features` table to find the region for each ZIP code of interest,
- Call `Map(...)` on the column of features (provided).

In [20]:

```
# Write code to draw a map of only the high-income ZIP codes
zip_features = Table.from_records(zips.features)
high_average_zips1 = income.with_columns('avg', ((income.column(
    "total")/income.column("returns"))*1000))
high_average_zips = high_average_zips1.where('avg', are.above(100000))
high_zips_with_region = high_average_zips.join('ZIP', zip_features,"ZIP")
Map(list(high_zips_with_region.column('feature')), width=400, height=300)
```

Out[20]:



Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

The higher average income locations are on the cost line of California and centralized with LA and the Bay area.

In [21]:

```
_ = project1.grade('q17')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed
```

Part 2: Water Usage

We will now investigate water usage in California. The `usage` table contains three columns:

- `PWSID` : The Public Water Supply Identifier of the district
- `Population` : Estimate of average population served in 2015
- `Water` : Average residential water use (gallons per person per day) in 2014-2015

In [22]:

```
# Run this cell to create the usage table

usage_raw.set_format(4, NumberFormatter)
max_pop = usage_raw.select([0, 'population']).group(0, max).relabeled(1, 'Population')
avg_water = usage_raw.select([0, 'res_gpcd']).group(0, np.mean).relabeled(1, 'Water')
usage = max_pop.join('pwsid', avg_water).relabeled(0, 'PWSID')
usage
```

Out[22]:

PWSID	Population	Water
0110001	340000	70.7
0110003	57450	90.2727
0110005	1390000	76
0110006	151037	57.1818
0110008	73067	96.6364
0110009	79547	68.6364
0110011	31994	85.8182
0310003	23347	82.8182
0410002	101447	142
0410005	11208	88.8182

... (401 rows omitted)

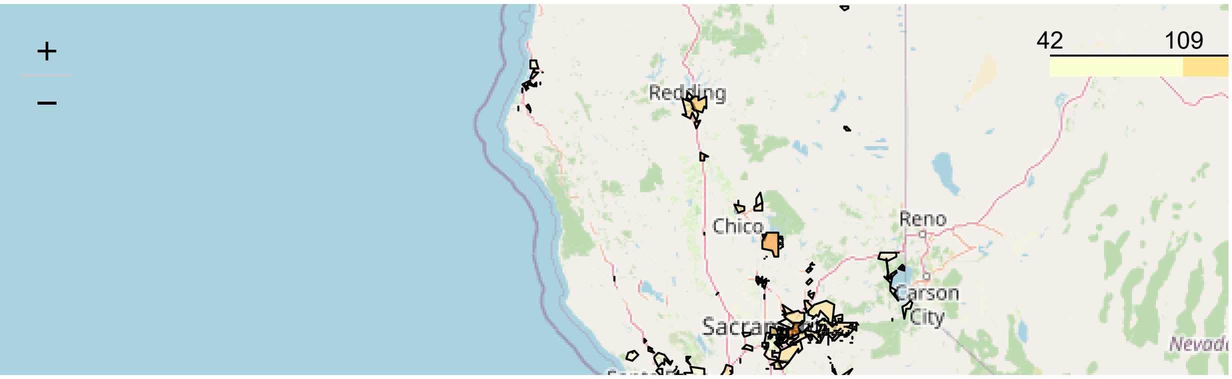
Question 2.1. Draw a map of the water districts, colored by the per capita water usage in each district.

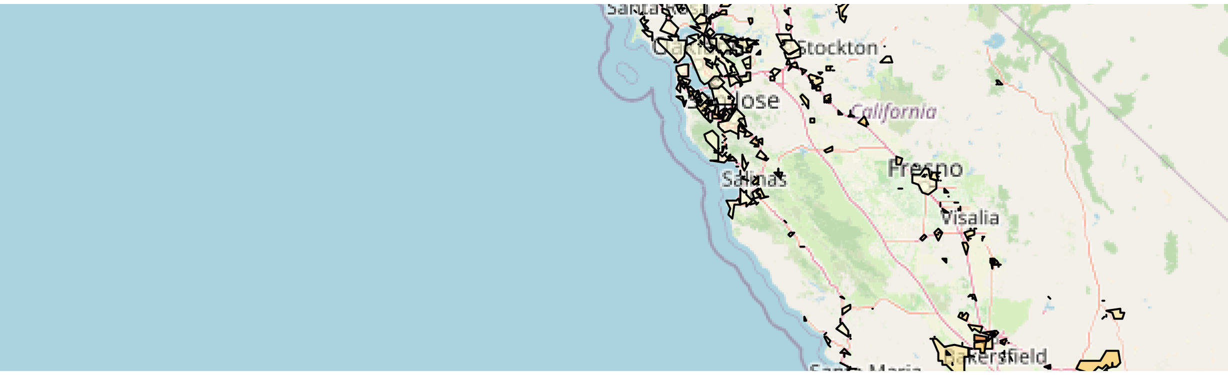
Use the `districts.color(...)` method to generate the map. It takes as its first argument a two-column table with one row per district that has the district `PWSID` as its first column. The label of the second column is used in the legend of the map, and the values are used to color each region.

In [23]:

```
per_capita_usage = Table().with_columns("PWSID", usage.column(0), "Per cap usage",
                                        usage.column(1*2))
districts.color(per_capita_usage, key_on='feature.properties.PWSID')
```

Out[23]:





In [24]:

```
_ = project1.grade('q21')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 2.2. Based on the map above, which part of California appears to use more water per person, the San Francisco area or the Los Angeles area?

Replace this line with a description of your findings.

Next, we will try to match each ZIP code with a water district. ZIP code boundaries do not always line up with water districts, and one water district often covers multiple ZIP codes, so this process is imprecise. It is even the case that some water districts overlap each other. Nonetheless, we can continue our analysis by matching each ZIP code to the water district with the largest geographic overlap.

The table `wd_vs_zip` describes the proportion of land in each ZIP code that is contained in each water district and vis versa. (The proportions are approximate because they do not correctly account for discontiguous districts, but they're mostly accurate.)

In [25]:

```
wd_vs_zip.show(10)
```

PWSID	ZIP	District in ZIP	ZIP in District
0110001	94536	9.41%	68.51%
0110001	94538	18.87%	67.31%
0110001	94539	13.13%	44.36%
0110005	94541	1.61%	68.11%
0110006	94541	18.68%	98.46%
0110005	94542	0.17%	6.79%
0110006	94542	18.24%	91.80%
0110001	94544	0.33%	3.02%

PWSID	ZIP	District in ZIP	ZIP in District
0110005	94544	0.00%	0.01%
0110006	94544	28.24%	97.61%

... (2836 rows omitted)

Question 2.3. Complete the `district_for_zip` function that takes a ZIP code. It returns the PWSID with the largest value of `ZIP in District` for that `zip_code` , if that value is at least 50%. Otherwise, it returns the string `'No District'` .

In [26]:

```
def district_for_zip(zip_code):
    zip_code = str(zip_code) # Ensure that the ZIP code is a string, not an integer
    districts = wd_vs_zip.where('ZIP', are.equal_to(zip_code))
    at_least_half =  np.any(districts.column("ZIP in District")>=.5)
    if at_least_half:
        return districts.sort("ZIP in District", descending=True).column(
            "PWSID").item(0)
    else:
        return 'No District'

district_for_zip(94709)
```

Out[26]:

```
'0110005'
```

In [27]:

```
_ = project1.grade('q23')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 4
    Failed: 0
[ooooooooook] 100.0% passed
```

This function can be used to associate each ZIP code in the `income` table with a `PWSID` and discard ZIP codes that do not lie (mostly) in a water district.

In [28]:

```
zip_pwsids = income.apply(district_for_zip, 'ZIP')
income_with_pwsid = income.with_column('PWSID', zip_pwsids).where(zip_pwsids != "No Dis
income_with_pwsid.set_format(2, NumberFormatter(0)).show(5)
```

ZIP	returns	total	farmers	PWSID
90001	20970	531,772	0	1910067
90022	26680	767,484	0	1910036
90024	14690	4,395,487	20	1910067
90025	25110	4,019,082	20	1910067
90034	29950	1,828,572	0	1910067

... (662 rows omitted)

Question 2.4. Create a table called `district_data` with one row per PWSID and the following columns:

- `PWSID` : The ID of the district
- `Population` : Population estimate
- `Water` : Total annual water usage (millions of gallons)
- `Income` : Average income in dollars of all tax returns in ZIP codes that are (mostly) contained in the district according to `income_with_pwsid` .

Hint: First create a `district_income` table that sums the incomes and returns for ZIP codes in each water district.

In [29]:

```
district_income = income_with_pwsid.group("PWSID", sum).join(
    'PWSID', usage, 'PWSID').drop ("ZIP sum", "farmers sum")
district_data = district_income.with_columns("Income",((district_income.column(
    "total sum") / district_income.column("returns sum"))*1000)).drop ("total sum", "re
district_data.set_format(['Population', 'Water', 'Income'], NumberFormatter(0))
```

Out[29]:

PWSID	Population	Water	Income
0110001	340,000	71	79,032
0110005	1,390,000	76	82,497
0110006	151,037	57	52,924
0110008	73,067	97	163,257
0110009	79,547	69	133,902
0410002	101,447	142	50,401
0410006	18,300	286	38,721
0410011	9,615	92	44,707
0710001	106,455	110	53,551
0710003	197,536	102	73,914

... (200 rows omitted)

In [30]:

```
_ = project1.grade('q24')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 2.5. The `bay_districts` table gives the names of all water districts in the San Francisco

Bay Area. Is there an association between water usage and income among Bay Area water districts? Use the tables you have created to compare water usage between the 10 Bay Area water districts with the highest average income and the rest, then describe the association. *Do not include any districts in your analysis for which you do not have income information.*

The names below are just suggestions; you may perform the analysis in any way you wish.

In [31]:

```
bay_districts = Table.read_table('bay_districts.csv')
bay_water_vs_income = bay_districts.join(
    "District", district_table, "popupContent").drop("feature").drop('id').join(
    "PWSID", district_data, "PWSID")
top_10 = bay_water_vs_income.sort("Income", descending=True).take(np.arange(10))
Bottom_19 = bay_water_vs_income.sort("Income").take(np.arange(19))
Gallons = np.average(top_10.column("Water"))-np.average(Bottom_19.column("Water"))
Gallons
```

Out[31]:

```
24.645454545454545
```

Complete this one-sentence conclusion: In the Bay Area, people in the top 10 highest-income water districts used an average of `24.6` more gallons of water per person per day than people in the rest of the districts.

In [32]:

```
bay_districts = Table.read_table('bay_districts.csv')
bay_districts.join("District", district_table, "popupContent").drop (
    "feature", "id").join('PWSID', district_data, 'PWSID').sort (
    "Income", descending=True)
```

Out[32]:

PWSID	District	Population	Water	Income
4110006	California Water Service Company Bear Gulch	58,895	170	1,160,494
4310001	California Water Service Company Los Altos/Suburban	68,163	125	349,183
4310009	Palo Alto, City of	64,403	89	334,057
4110017	Menlo Park, City of	16,066	75	278,733
2110002	Marin Municipal Water District	188,200	86	176,643
0110008	Pleasanton, City of	73,067	97	163,257
4310007	Mountain View, City of	76,781	66	138,570
4110021	Estero Municipal Improvement District	37,165	66	137,893
4110001	Mid-Peninsula Water District	26,730	84	137,537
4110008	California Water Service Company Mid Peninsula	135,918	70	135,967

... (19 rows omitted)

Question 2.6. In one paragraph, summarize what you have discovered through the analyses in this project and suggest what analysis should be conducted next to better understand California water usage, income, and geography. What additional data would be helpful in performing this next analysis?

TThe summary/analysis for this project would be a combination from the previous explanations. The major item to call out would be regions in California that are heavily populated clearly sway faveolate to income but not so favorable to water usage. Since there is an increase of ~24.6 gallons per person per day vs. the rest of the districts.

I think it would be helpful to know within populated zip codes (e.g. LA and the Bay) working class vs. government aid individuals. This is to potentially exclude individuals that receives government aid to know if that would change water usage per person which I suspect is a diluted number. I think this would exclude a good portion of the number of people who file a return in these zip codes. Which in return would increase the water usage.

Congratulations - you've finished Project 1 of CS260!

To submit:

1. Select `Run All` from the `Cell` menu to ensure that you have executed all cells, including the test cells. Make sure that the visualizations you create are actually displayed.
2. Select `Download as PDF via LaTeX (.pdf)` from the `File` menu. (Sometimes that seems to fail. If it does, you can download as HTML, open the .html file in your browser, and print it to a PDF.)
3. Read that file! If any of your lines are too long and get cut off, we won't be able to see them, so break them up into multiple lines and download again. If maps do not appear in the output, that's ok.
4. Submit that downloaded file (called `project1.pdf`) to Gradescope.

In [33]:

```python
# For your convenience, you can run this cell to run all the tests at once!
import os
_ = [project1.grade(q[:-3]) for q in os.listdir("tests") if q.startswith('q')]
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Question > Suite 1 > Case 2

>>> print(income.take([0, 1, 2]))
ZIP   | returns | total   | farmers
90001 | 20970   | 531,772 | 0
90002 | 18960   | 467,128 | 0
90003 | 26180   | 618,848 | 0

# Error: expected
#     ZIP   | returns | total      | farmers
#     90001 | 20970   | 531,772    | 0
#     90002 | 18960   | 467,128    | 0
#     90003 | 26180   | 618,848    | 0
# but got
#     ZIP   | returns | total   | farmers
#     90001 | 20970   | 531,772 | 0
#     90002 | 18960   | 467,128 | 0
#     90003 | 26180   | 618,848 | 0

Run only this test case with "python3 ok -q q12 --suite 1 --case 2"
---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 1
[oooook.....] 50.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests
```

```
----------------------------------------------------------------------
Test summary
    Passed: 4
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed
```

If you want, draw some more maps below.

In [68]:

```
# Your extensions here (completely optional)
```