Lab 8: More Table Joins

What You'll Learn: This lesson covers additional work with tables, including importing text files, combining rows, and navigating tricky joins. The work is organized as small projects, the first with step-by-step instructions, and the next two less so. You should have read, and be ready to refer to Chapter 8 in the GIS Fundamentals text. This project requires synthesis of what you've learned up until now.

Data are in the L8\ directory, separated by projects.

Project 1: A Common Table Join Problem

This project introduces something quite common, joining ASCII tabular data with a shapefile, with two kinds of common pre-processing. Here, we will combine a text file on corn production, in bushels per acre, for US counties with a polygon layer, but we must first create a join item, and summarize our data on a county basis, as we often must do for many other social, political, and environmental data distributed as non-spatial files.

First, Subset the Table Variables

Start QGIS, and

- add the lower48cnty shape file from the L8\Project1\ subdirectory. This is a general county boundary file, downloaded from the USGS Geodata Portal. It contains county boundaries, county names, state and county Federal Information Processing Standard (FIPS) codes for state and county, among other variables.

- add the text file cnty26.csv to this data view, using the add vector data tool. This file contains 1996 through 1999 corn production, in bushels, for counties in the United States where production is above a minimum threshold. These data were downloaded from the National Agricultural Statistical Service website, www.nass.usda.gov/, and we're most interested in the columns:

Stfips: the state FIPS code,
CoFips: county FIPS code,
Harvested: the acres harvested for a given year/yield category in a county,
Yield: Bushels per acre harvested for the year/yield category,
Production: Total bushels produced for the year (yield times harvested) for the given yield level.

We want to delete most columns in the cnty26 file. Keep **Year**, **Stfips, CoFips, Harvested, Yield,** and **Production** from the cnty.csv table, delete the rest of the fields.

Remember that to easily delete multiple fields, first change to table view, toggle editing, and then use the delete columns tool you learned earlier to remove all but the desired columns from the cnty26 table.

Save and exit the changes to the table after deleting columns.

We're interested in the following columns in the lower48cnty layer:
Area: polygon area
Perimeter: polygon perimeter
State: the state FIPS code
County: the county FIPS code
Name: The county name

Keep these fields, delete the rest from the lower48cnty layer, again using the tools learned. Remember to save changes before exiting the table.

Second, Create Valid Keys in Each Table So We Can Join Them

We want a data layer that lists average corn production by county. We can get this if we combine the data in these tables, the cnty26.csv file, and the attribute table for the lower48cnty.shp. Unfortunately, there are two problems. First, we don't have a ready-made key for the join. There is no column that maps cleanly from the *raw_corn_dat.dbf* records to the *lwr48* shapefile records, matching counties properly. You can't use county FIPS codes alone, as described in the textbook in the section on concatenated keys. Second, there are four years of data for each county in the cnty26.csv file, and we want average production for each county.

Open the Attribute Table for the *lower48cnty* shapefile.

Notice that both tables have county and state FIPS codes, in the COUNTY and STATE columns, in the polygon file, and in the Stfips and Cofips in the .csv file. Each state has a unique FIPS code, and each county within a state has a unique code. If we combine the state and county FIPS codes, we can create a unique ID for each county/state combination in both files, we can then use this composite variable as a key for the join. This is described in the textbook, in Chapter 8, under concatenated keys.

Toggle the editing button for lower48cnty table (to start editing),

Calculate a new column using the Field calculator. Name this column sta_count and assign the Type to be a Whole number (integer), with at width of 8 or larger (precision).

Use the ***Field calculator*** to assign a unique FIPS code to the sta_count variable.

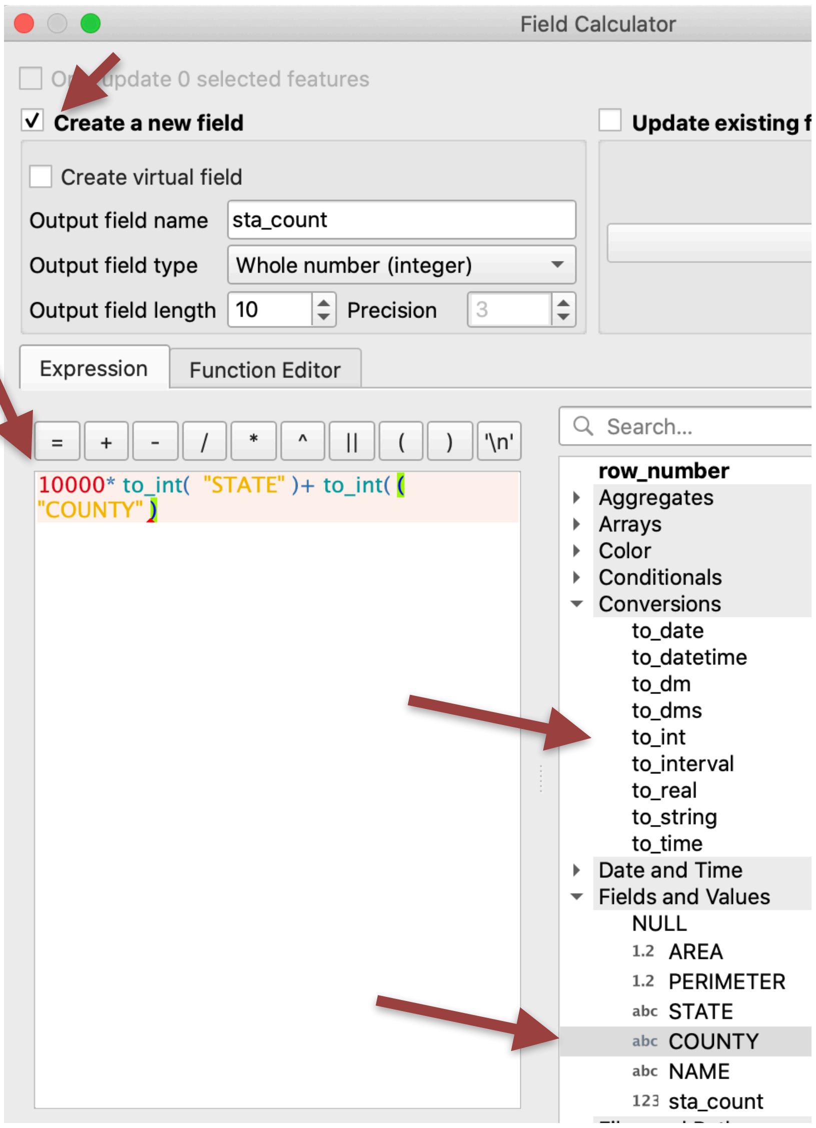
State variables are two digit, while county variables can have up to three digits

Specify "Create new field" (see figure at right).

Create a unique ID by multiplying the STATE by 10000 and adding to COUNTY (remember, the fields are accessed through the fields and values caret).

Note that in the formula we must convert the STATE and COUNTY variables to integers, using the to_int operation, because they are of type STRING in the input dataset.

The formula creates a unique 5-digit code for each polygon (see the figure). Verify this worked by looking at the STATE, COUNTY, and sta_count fields within the same row.



Output field name: sta_count
Output field type: Whole number (integer)
Output field length: 10
Precision: 3

```
10000* to_int( "STATE" )+ to_int( (
"COUNTY" )
```

Think about why this works, given that state codes are two digits, and county codes are three digits. Why multiply by 10000, instead of 1000 or 100,000? Would the reverse order work, county *10000+state? Why or why not?

Save your changes to lower48cnty.shp and stop editing, by clicking again on the toggle edit button.

We need to create a similar key for the cnty26.cvs file, but the conversion doesn't always work on a .csv file. At the time of this writing, the Field Calculator sometimes stops and returns an obscure error message to the effect that the field can't be converted.

We have fewer issues if we export the file to a .dbf format.

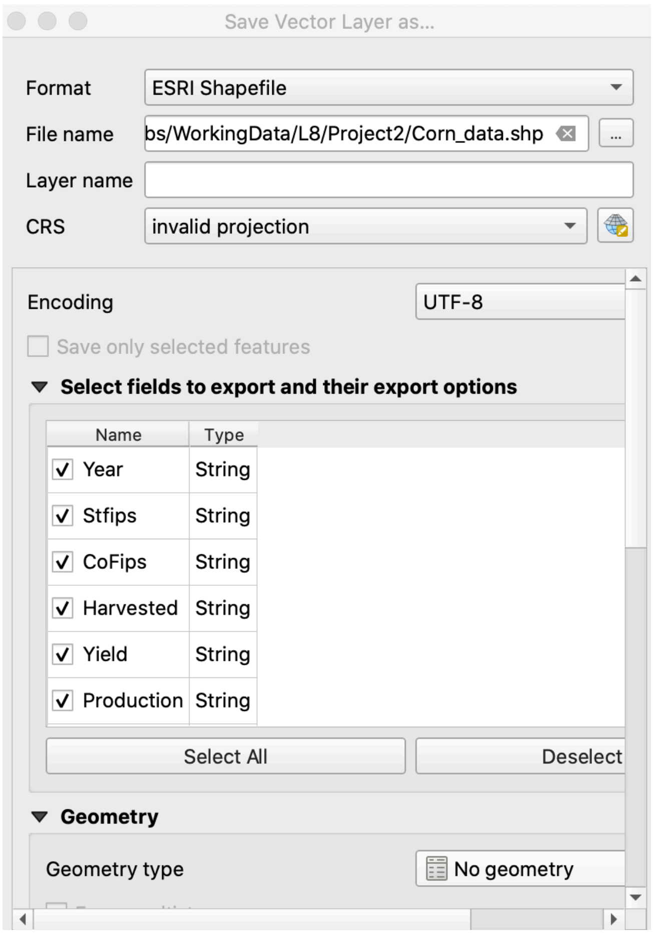
This format isn't explicitly supported by QGIS, but we can generate one via a shape file export, just ignoring the geometry.

Click on the table in the Layers/TOC, and Export.

Choose an ESRI shapefile format.

Ignore the invalid projection warning in the CRS.

Set the geometry to "No geometry"



Format: ESRI Shapefile
File name: bs/WorkingData/L8/Project2/Corn_data.shp
Layer name:
CRS: invalid projection

Encoding: UTF-8
Save only selected features

Select fields to export and their export options

Name	Type
✓ Year	String
✓ Stfips	String
✓ CoFips	String
✓ Harvested	String
✓ Yield	String
✓ Production	String

Geometry
Geometry type: No geometry

I named the file Corn_data.shp here, pick something descriptive and different from the original Cnty26 csv file name.

Run the export. You should get a conversion message saying it was successful, then a warning saying the coordinate system is invalid.

Load just the output .dbf file using the add vector data tool, in my case I named my export Corn_data.dbf. It will appear as a table in the Layer/TOC.

Open the Corn_data table, and toggle editing on, and use the Field Calculator to add an integer sta_count column similar to the one above.

Note the type of the Production field. See that it is a text field. Create a new variable, called something like R_Production, that is a real number, again with the Field

Calculator, applying the operation to_real to the text production variable to create a real number we can use in subsequent steps.

If you wish, use the delete column tool to remove the columns you no longer need, e.g., the Stfips, CoFips, and Production (text version) columns.

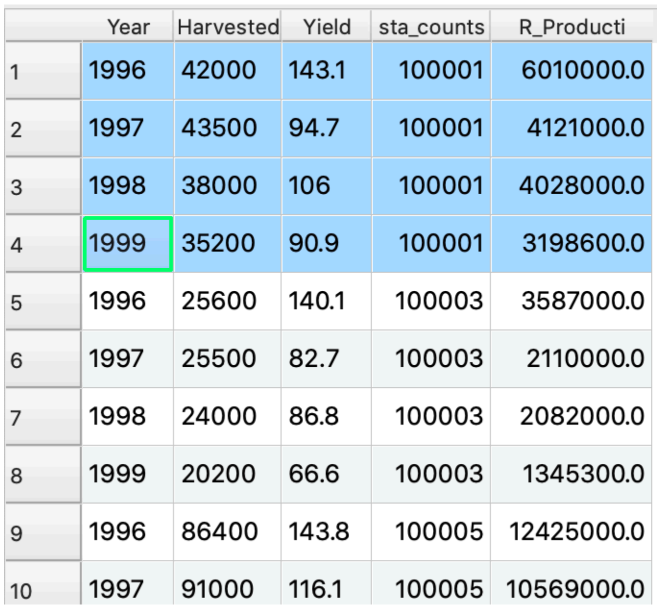
After you calculate the new columns, verify the contents by examining a few rows, then save and toggle editing off.

Third, Summarize Production

Our overall goal is to calculate the average Production of Corn, in bushels for each county. We need to summarize the Production variable by unique state/ county combination. However, as is often the case, this isn't straight-forward.

Open the Corn_data table and sort it by the sta_count column.

Note that there are typically four entries with the same sta_count values, corresponding to different years (blue rows for one instance, at right). We need to average these, and save the results into a new table, so that we have one entry per county.



	Year	Harvested	Yield	sta_counts	R_Producti
1	1996	42000	143.1	100001	6010000.0
2	1997	43500	94.7	100001	4121000.0
3	1998	38000	106	100001	4028000.0
4	1999	35200	90.9	100001	3198600.0
5	1996	25600	140.1	100003	3587000.0
6	1997	25500	82.7	100003	2110000.0
7	1998	24000	86.8	100003	2082000.0
8	1999	20200	66.6	100003	1345300.0
9	1996	86400	143.8	100005	12425000.0
10	1997	91000	116.1	100005	10569000.0

There is an external tool named Group Stats that helps with these calculations. You may need to load the plug-in, as demonstrated earlier with different external tools.

Open the Plugins -> Manage and Install Plugins, and search for GroupStats. If it is not shown as installed under all plugins, download and check to install.

Find the Group Stats toolbar, typically loaded in one of the lower rows of existing toolbars that cross the top of the main QGIS window.

Click on Group Stats and

Select Corn_dat in the Layers dropdown and

Drag sta_count to the Rows box,

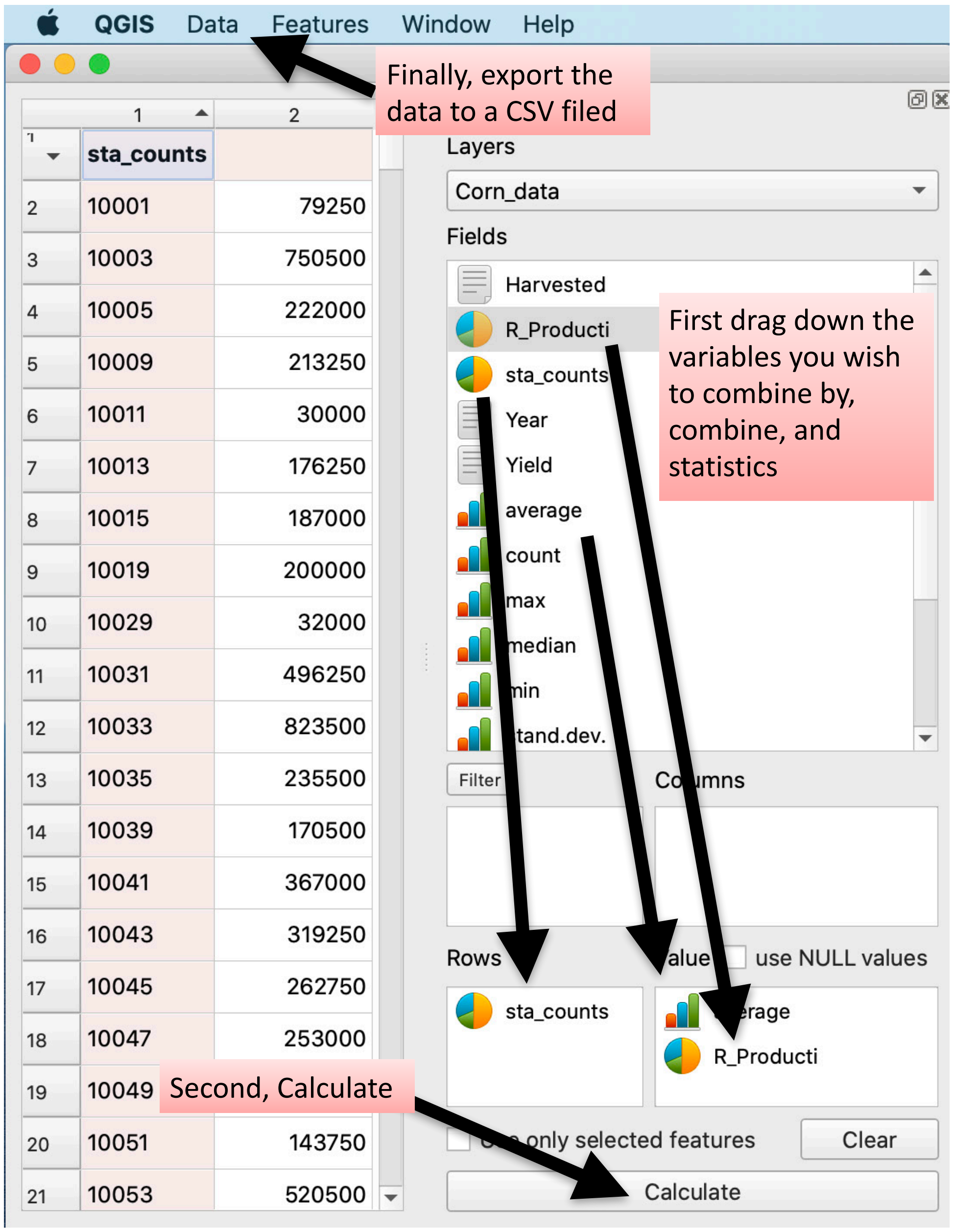
Drag average and R_Production to the Value box. (see left)

Click on Calculate.

Your output should appear as shown to the right.

Next click on Data and Save All to CSV file.

Name the file something like AvgProd.csv



Finally, export the data to a CSV filed

First drag down the variables you wish to combine by, combine, and statistics

Second, Calculate

Layers: Corn_data

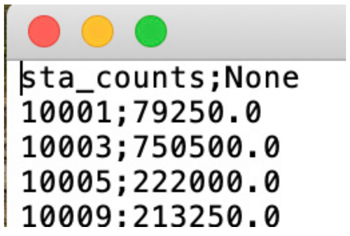
Fields: Harvested, R_Producti, sta_counts, Year, Yield, average, count, max, median, min, stand.dev.

Rows: sta_counts

Value: average, R_Producti

	1	2
1	sta_counts	
2	10001	79250
3	10003	750500
4	10005	222000
5	10009	213250
6	10011	30000
7	10013	176250
8	10015	187000
9	10019	200000
10	10029	32000
11	10031	496250
12	10033	823500
13	10035	235500
14	10039	170500
15	10041	367000
16	10043	319250
17	10045	262750
18	10047	253000
19	10049	
20	10051	143750
21	10053	520500

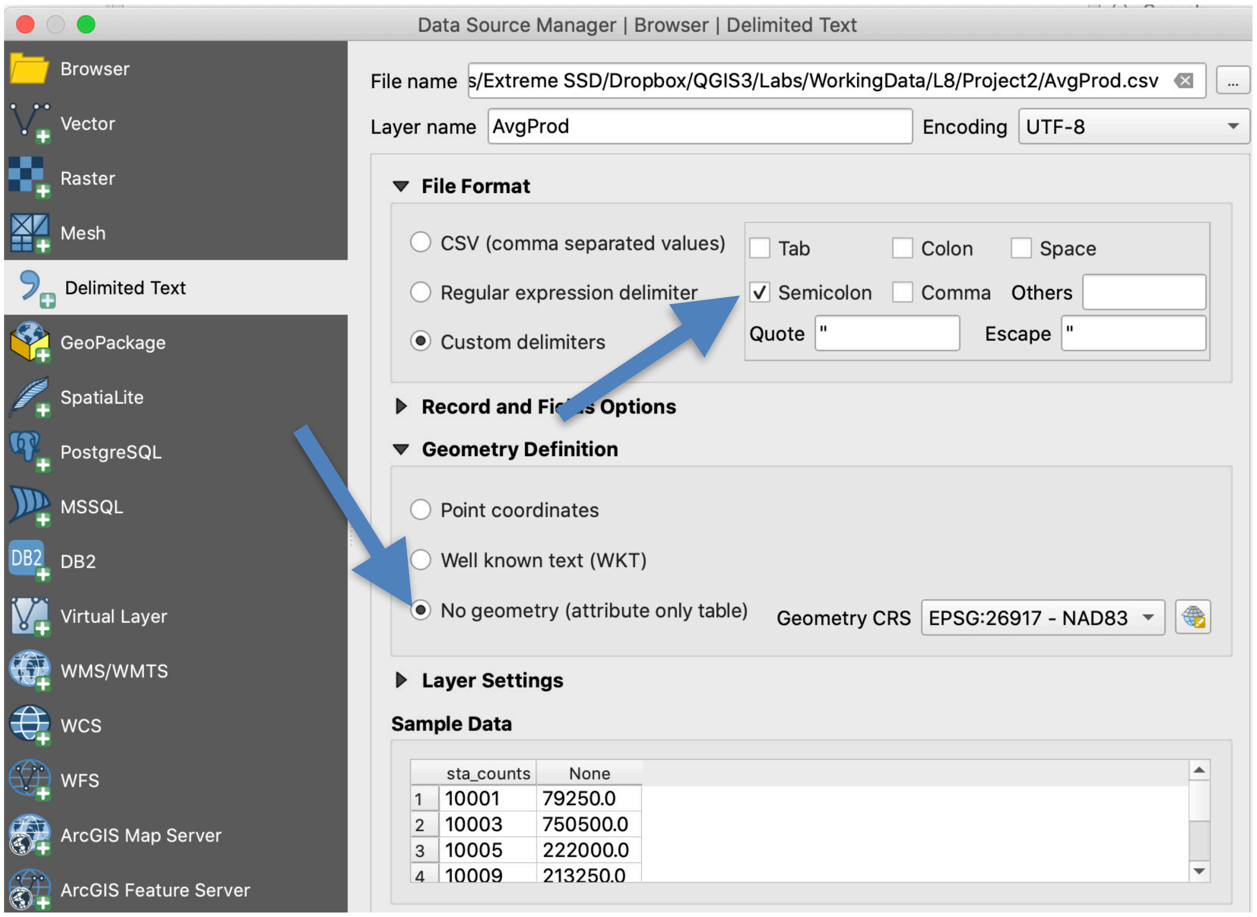
When I open the file in a text reader, see something like the figure below:



```
sta_counts;None
10001;79250.0
10003;750500.0
10005;222000.0
10009:213250.0
```

I see that the delimiter between columns is a semi-colon, and that there are not very informative column headings. I'll use this information when I import these data.

When we add a csv or similar file to Layers panel as before, using the add vector tool, we noticed it defaults to text variables, and we had to convert some of them. If we instead use the Data Source Manager, Delimited Text option, we can specify that the delimiter is a semi-colon, and that we want no geometry for the imported file (it is a table only).



File name s/Extreme SSD/Dropbox/QGIS3/Labs/WorkingData/L8/Project2/AvgProd.csv
Layer name AvgProd
Encoding UTF-8

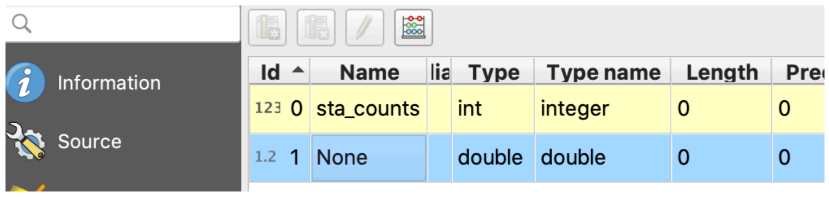
File Format: [ ] CSV (comma separated values) [ ] Regular expression delimiter [✓ Custom delimiters]
[ ] Tab [ ] Colon [ ] Space [✓ Semicolon] [ ] Comma Others
Quote " Escape "

Geometry Definition: [ ] Point coordinates [ ] Well known text (WKT) [✓ No geometry (attribute only table)]
Geometry CRS EPSG:26917 - NAD83

Sample Data

	sta_counts	None
1	10001	79250.0
2	10003	750500.0
3	10005	222000.0
4	10009	213250.0

Inspect the variables for the file in the Layers/TOC after import. Note that, when imported, the variables default to integer and real, as we need for mapping.



Id	Name	lia	Type	Type name	Length	Pre
123 0	sta_counts		int	integer	0	0
1.2 1	None		double	double	0	0

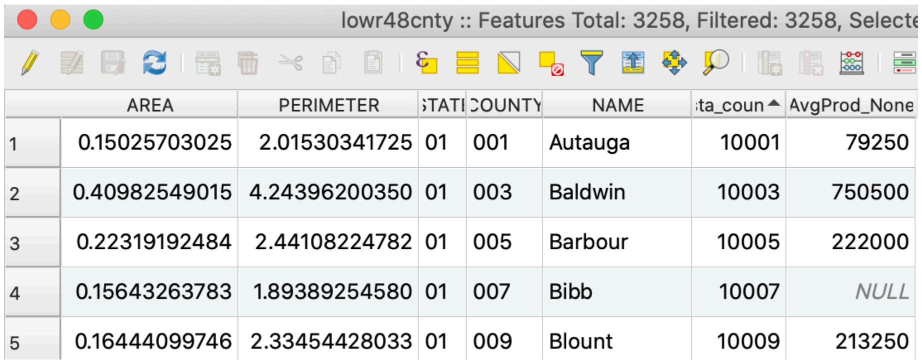
Also note that your summary field will be named "None". We can rename it later, for now we'll just note its name.

Join the AvgProd table you just created to the *lower48county.shp* file

Examine the .shp file to verify that the new columns joined correctly.

Note there are some counties that do not grow corn. They will not have any joined data from the AvgProd file.

Your lwr48.shp file should look something like this:


lowr48cnty :: Features Total: 3258, Filtered: 3258, Selecte

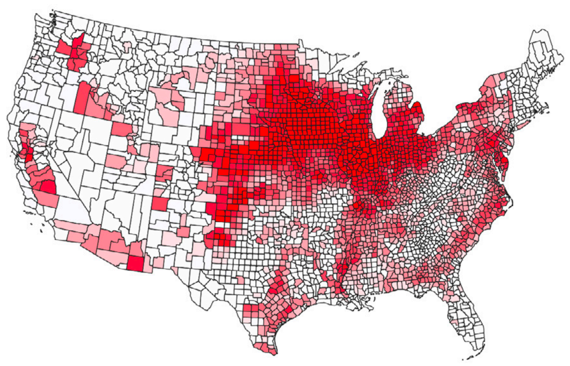
	AREA	PERIMETER	STATE	COUNTY	NAME	ta_coun	AvgProd_None
1	0.15025703025	2.01530341725	01	001	Autauga	10001	79250
2	0.40982549015	4.24396200350	01	003	Baldwin	10003	750500
3	0.22319192484	2.44108224782	01	005	Barbour	10005	222000
4	0.15643263783	1.89389254580	01	007	Bibb	10007	NULL
5	0.16444099746	2.33454428033	01	009	Blount	10009	213250

Symbolize your joined layer with a color gradient, a light/white color at the low end of Average Production, and a darker color at the higher end.

Note that there are large "holes," missing areas where county boundaries are missing, e.g., most of the state of Maine, or central Florida, or the Intermountain West? These are counties with zero production, and so not linked in the table. Their values are Null for the AveProd variable, and so they aren't mapped.

We can fix this in at least two ways. The first is to edit the table, selecting all of those records with a Null production value, and converting the Null AveProd values to zero values, using the Field Calculator.

The second is simply to load a copy of the Lower48cnty data to the Layers, symbolize it with the same color as the zero value for the joined Lower48cnty and AveProd data, and position it under those data to outline the missing counties.



Use either approach to display a map similar to that at the right, with the usual map elements, and export as a pdf.

Project 2

Your next task is to produce a map showing average income by county in California. These instructions will be less detailed than previous lessons, as we omit most steps we've already covered. The goal is for you to synthesize these previously-taught tools on your own.

Start a new blank project. Do not add any layer yet.
Maps must be produced in a UTM Zone 11, NAD83 projection (EPSG: 26911).

Data for this project are in L8\project2\ subdirectory:

- California county boundaries are in *Cal.shp,* you should reproject the *Cal.shp* file to the UTM Zone 11, NAD83 projection (EPSG: 26911).
- *Income.dbf* is a database file that lists average per capita income by county; add this to file to the Income data frame.

The income and county files have a common attribute – cnty_name. This common field allows you to join these files together.

Income Map

You need to produce a map showing only those counties with an average per capita income greater than $16,000. Create this map in a new project.

You will need to be careful on the join of the income table, Income.dbf, to the California county layer, Cal.shp.

Inspect the fields, and types. Which of the fields is most likely to work as keys in your join?

Be careful to inspect the output from your join. The first attempt will often fail, having to do with field types.

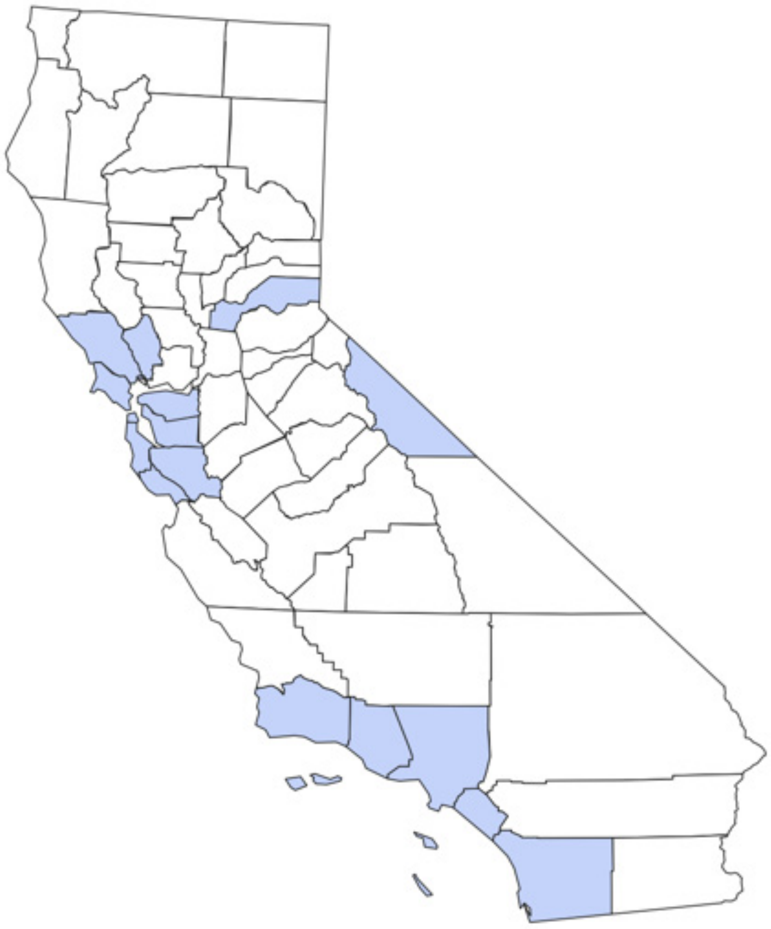
There are two common ways students can identify over vs. below $16,000 PCI counties after joining the tables. The first uses Symbology to differentiate between counties making more than $16,000 from those making less. By setting the boundaries for your categories in a graduated map from 0 to 16,000, and 16,000 to larger than the largest income value, you can then click on the patches and assign distinct colors. We've done each of these steps in different previous labs, so we leave the synthesis/combination for you to figure out.

A common second approach uses a binary indicator attribute.

In this workflow, you can join the tables and save to a new file. Then add a new integer variable to the new file, and assign a value of zero to all records. Select values from the table that have a per capita income above 16,000, and assign them a value of 1.

Save these changes, then create a categorical map based on this binary (0/1) variable.

The output should look something like this.



Project 3

Park/Forest Map
You need to create a map of California that identifies counties containing a park, a forest, or both. The database file named *rec.dbf* lists many recreation types, including parks and forest. We must combine this with the state county outline layer, Cal.shp, through a join, but this join is problematic, because the source *rec.dbf* table does not have a proper key for this join. We have to create a proper key before joining, removing **many-to-one** relationship for the counties in *rec.dbf* with counties in the *Cal.shp* file. There may be multiple entries including for each county, one for each park, forest, reservoir, or other features found in a county. You need to develop a list of counties with parks or forests from this *rec.dbf.* ***Video: Lab 8 (Park_Forest)*****.**

One way is to open the *rec.dbf* table and select all those with parks, and save your selected records to a Parks table. For clarity you might create an indicator variable, with a column value of 1 or all the counties with parks. You might have repeat entries for a county, e.g., two parks in the same county, but this will not create problems with our workflow. Save and exit this new Parks table.

Join this table to your Cal.shp table. Export the joined data as a new layer, let's call it CalParks.

Load the rec.dbf, again, this time selecting and saving only the records with forests to create a new Forests table. You don't need to create an indicator variable, but you might if you find it clearer. Save and exit this new forests table.

Join the forests tables to *CalParks.shp*, and then select those rows with a park or a forest. There is a map showing the correct set of counties below.

A note of caution; this is a tricky exercise, and many students do not produce this final map correctly. The main problem comes from multiple entries (parks or forests) for each county. You need to be very careful in the table joins, and look at the maps you produce. Make sure your final product makes sense. One helpful guide may be the flowchart or the maps of the respective component maps; in this case a map of those counties with forests, and a separate map of those counties with parks. The "OR" condition should include data from both joined files, so your final project 2 map should have both colored in.

Apply your analysis until it matches the counties shown below, then create a map with the two sets of colored polygons, a legend, title, scalebar, north arrow, and your name.

Counties with either a park or a forest.