**Integrating Google Fusion Tables in the Google Maps API for CMS.633**

**Daniel Sheehan**

**Senior GIS Specialist, MIT Libraries**

[**dsheehan@mit.edu**](mailto:dsheehan@mit.edu)**,** [**gishelp@mit.edu**](mailto:gishelp@mit.edu)

**libguides.mit.edu/gis**

**Introduction**

Google Fusion tables have been around for several years yet remain in beta. Fusion tables are fundamentally flat files, or spreadsheets, but have several unique properties, including one that we will explore here – they are spatially enabled, by default. As you saw in the example in the presentation, you can have a two column location, such as latitude and longitude. You can also use a single field address, like ’77 Massachusetts Avenue, Cambridge, MA 20139’ or ‘Harvard Yard’ or even ‘TD North Garden’ (the latter is where the Boston Celtics and Bruins play). Anything that returns a single address in Google Maps will work here. For instance, ‘Harvard Yard’ works because there is only one well known one but ‘Schools in Cambridge’ would not, since many would be returned. The location type that you might use depends on your application. If you interested in urban areas and have addresses, use them – they will work well, depending on the country. If you have sensors placed in unpopulated remote areas then you likely have used a GPS to get the latitudes and longitudes for their locations with a GPS. You do need to be consistent so don’t mix latitude and longitude with addresses.

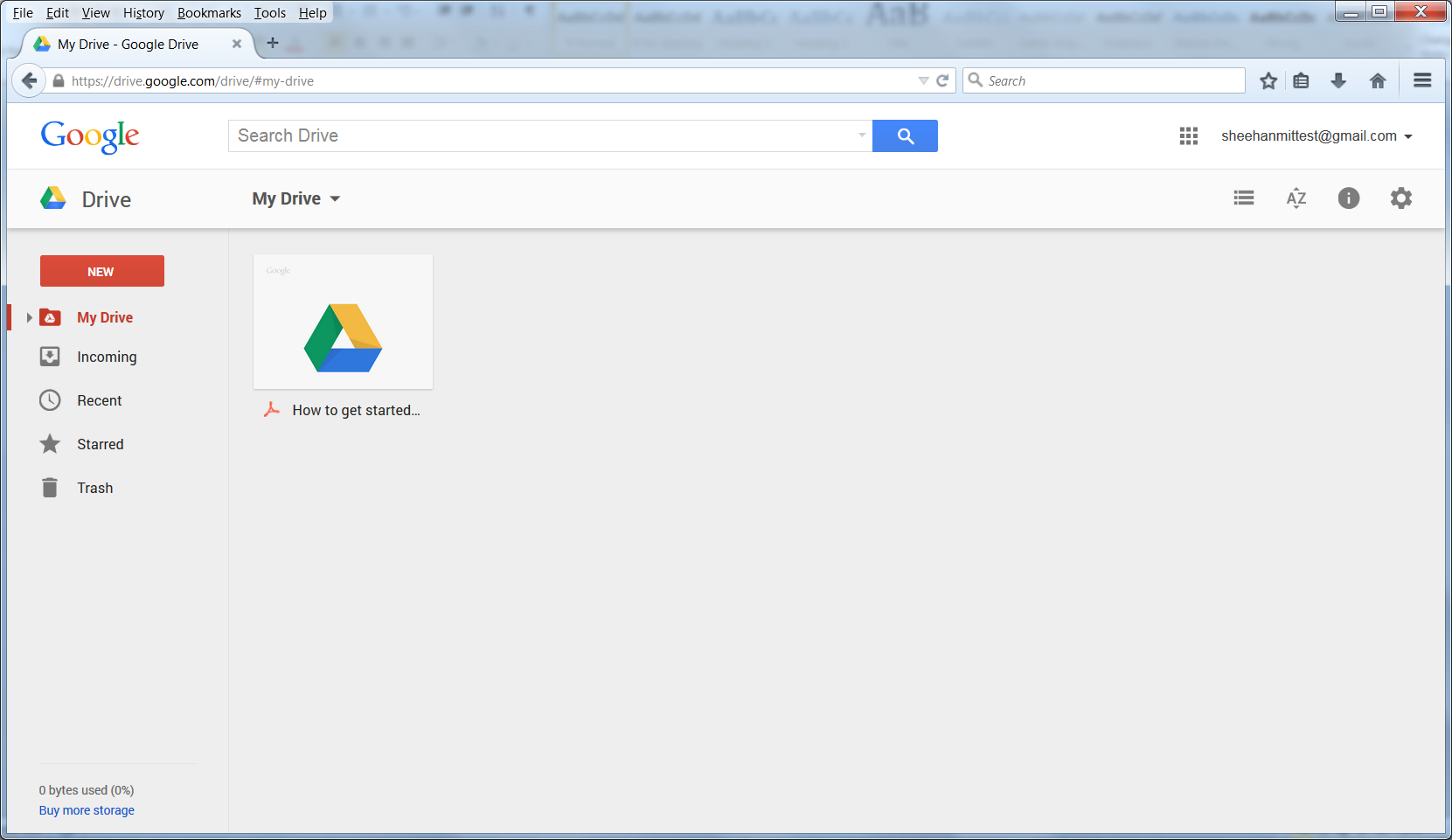
Once you have a location field, you will have a map tab. If you have latitude and longitude, as the location column, then your map draws immediately when you open the map tab. If you have addresses, then the addresses are geocoded. Unfortunately, while the addresses draw in the correct locations, as determined by available street networks, they can’t be exported with the latitude and longitude. In other words, you can’t use this as a geocoding tool.

You need to have a Google account (including a Gmail account) in order to use drive.google.com and Fusion tables. Please sign on or create an account now using mail.google.com.

Google Maps API is a well-developed interface to the mass market Google Map (maps.google.com). Using the API requires some facility with HTML and with Javascript. The more you know of both of these, the more sophisticated maps and web pages you can create. Once we have a table with spatial data in it, we will work with HTML and Javascript to show your data in a browser, outside of Fusion tables.

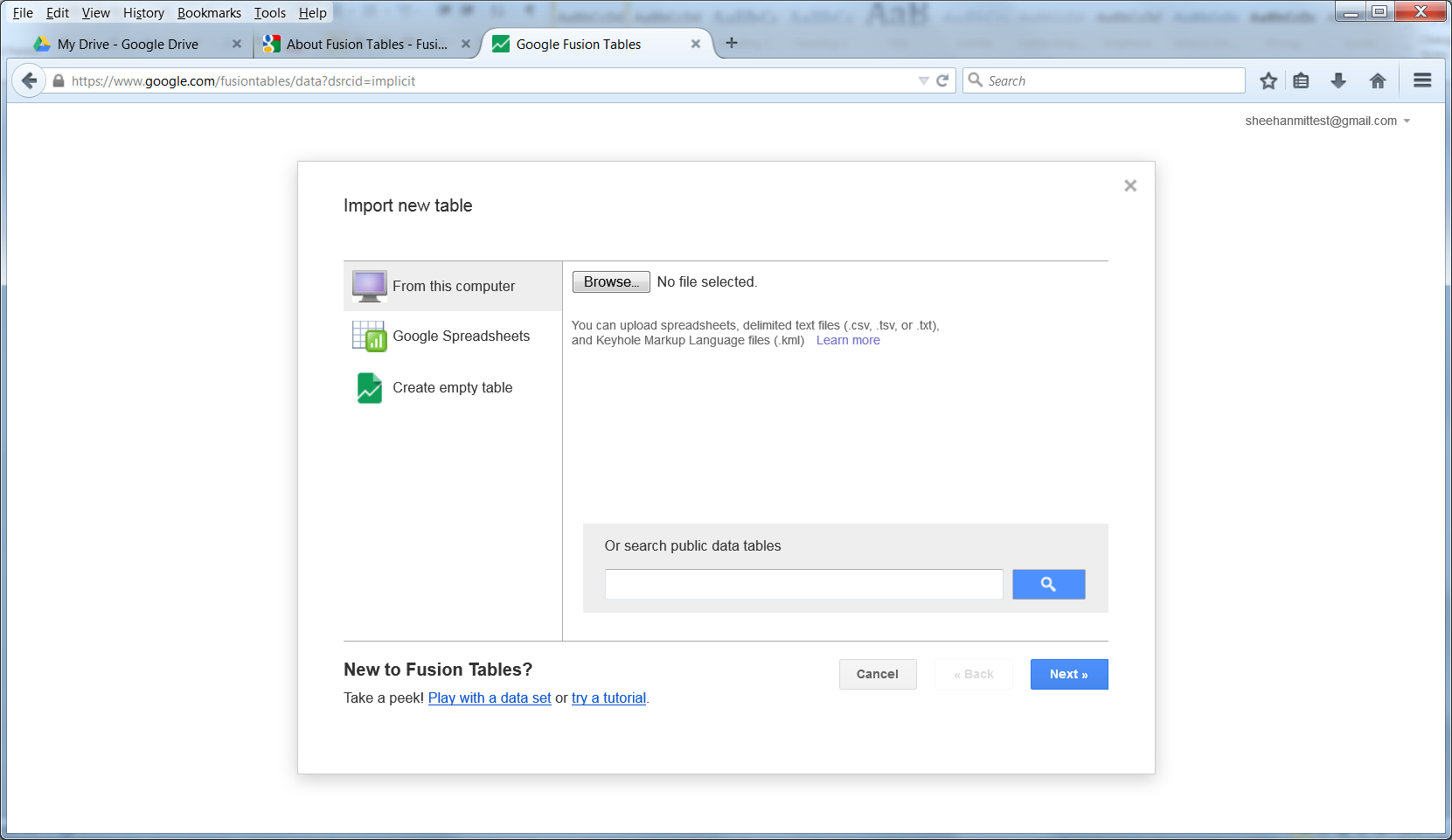
**Making a Fusion Table**

First, make a Fusion table. Google Fusion tables appear in your Google Drive (drive.google.com). You can create them from the following interface, once you connect Fusion tables to your Google Drive. Try to create one from the red NEW (or CREATE) button in drive.google.com:



If you have made Fusion tables before, you should see Fusion tables as an option available in the NEW (or CREATE) list, under More (options). If not, try this link: <https://support.google.com/fusiontables/answer/2571232?hl=en&ref_topic=1652595>

There will be a CREATE FUSION TABLE button on this page. Click it and you will be prompted for your password, even if you are already logged on. You should see this window:



Use the Moby Dick data as a data source for this exercise. In the future, if you are comfortable with Excel and have data in hand, and you have Excel on your machine, then you can create spread sheets with the following fields or something that better matches your data:

Latitude

Longitude

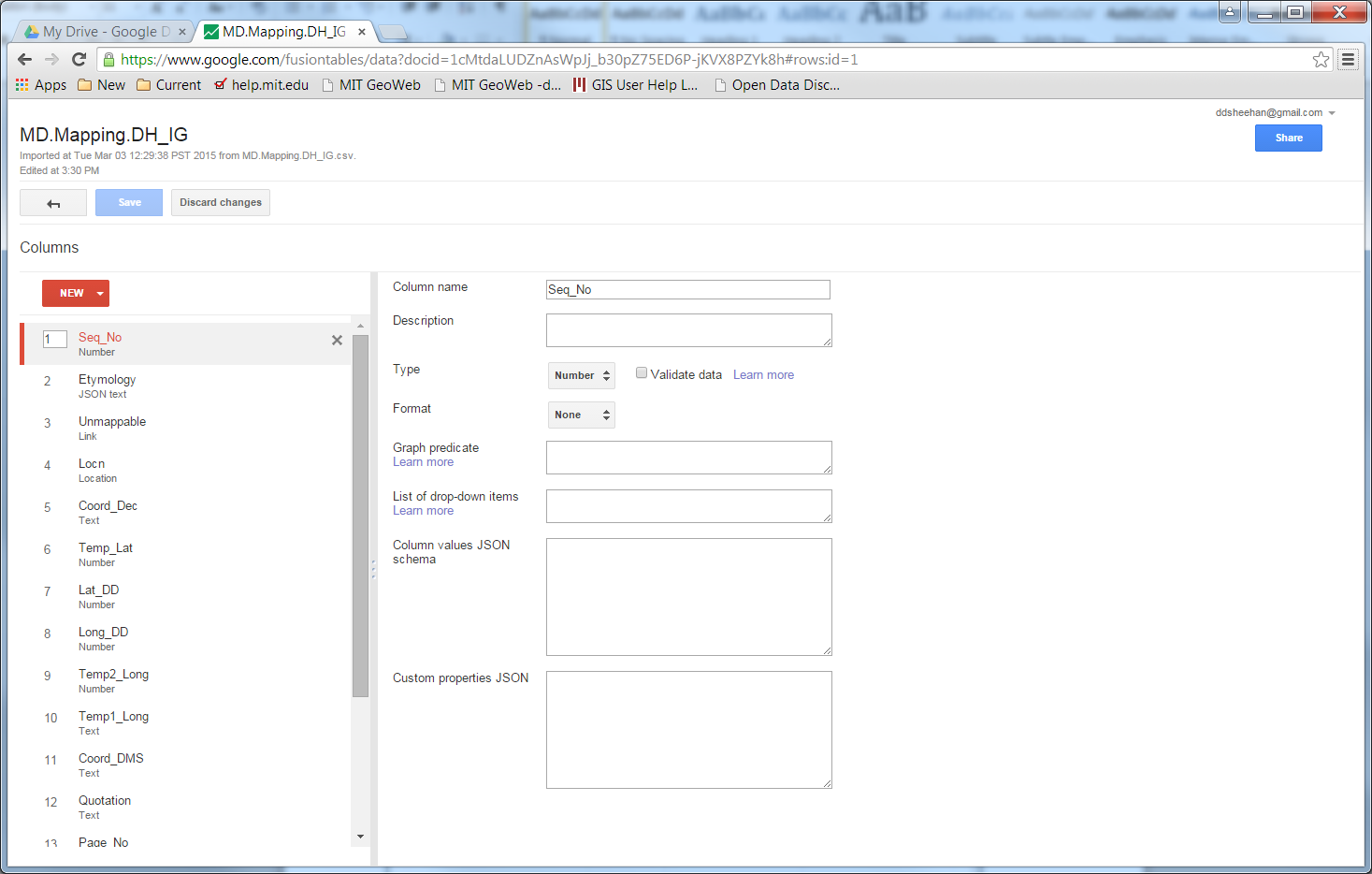
Notes

Link

(Latitude and Longitude and can be substituted with address, linking should be a URL that opens when users click on it.)

Import the Moby Dick CSV file from the above interface. If you are already familiar with Google Spreadsheets, feel free to use one and import it using the above interface. There will be several steps on the way. Follow the directions that appear. There is very little input from you required in the import process. By default, the table name is the name of your file.

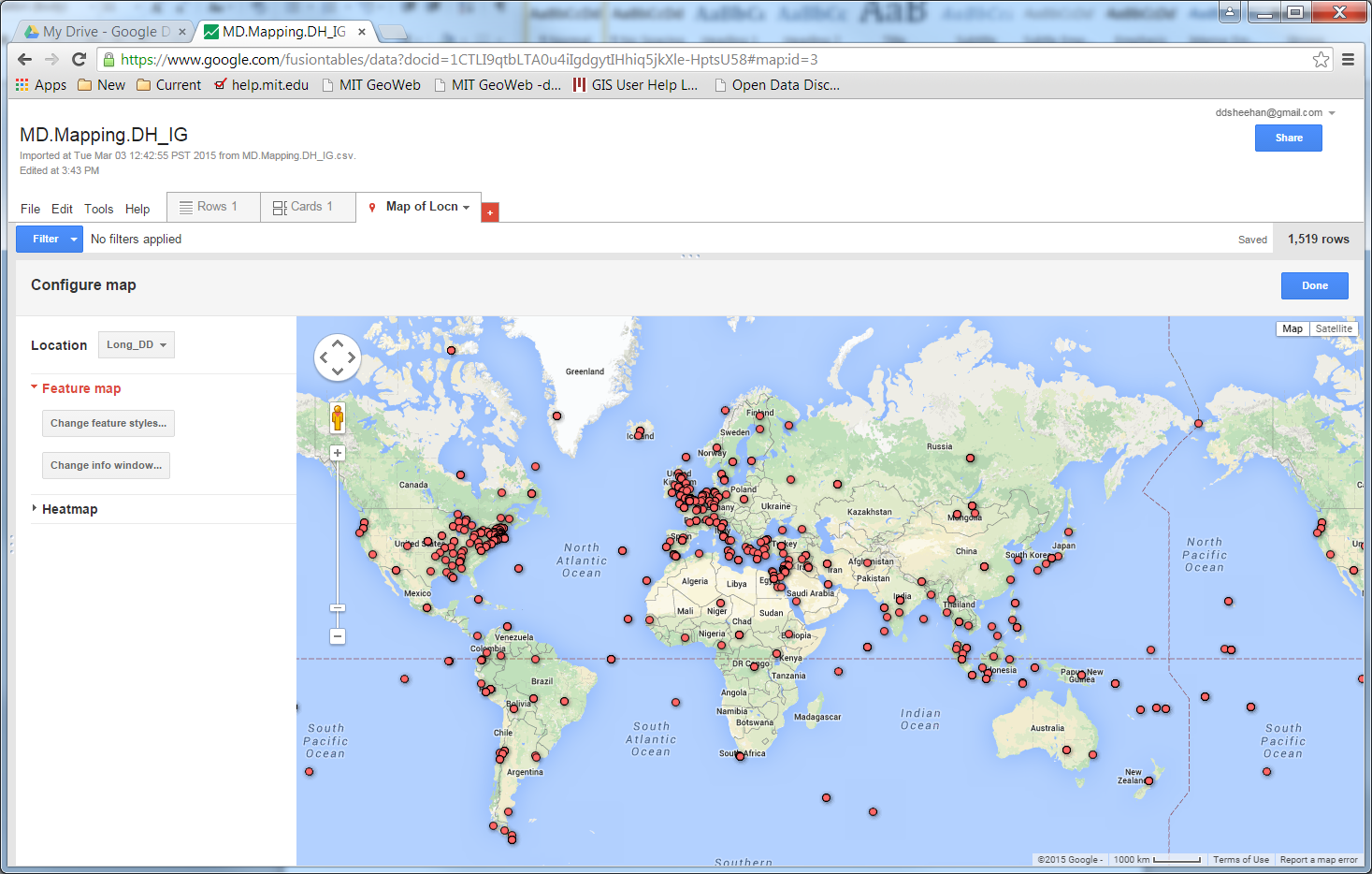
Once you have the file imported, you should set the location field. By default, the country name was used in my case and it ignored the latitude and longitude columns. Use the Change Columns tool in the Edit menu. You will see this interface:



Change the Locn field to type Text. Then change Long\_DD to a Location type and make it a two column location. Long\_DD is the longitude in decimal degrees. Use Lat\_DD (latitude in decimal degrees as the latitude field. Order is important - be sure to change the Locn field before you make Save your changes.

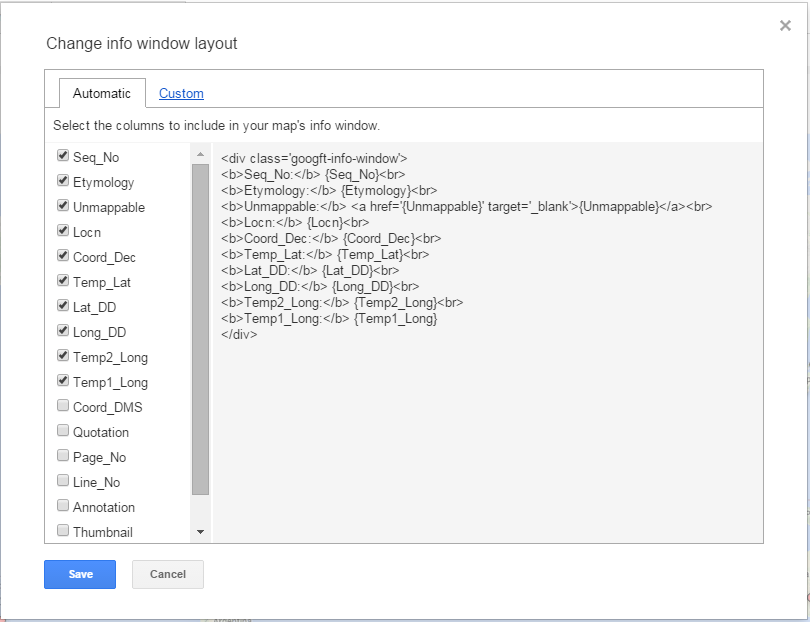
**Opening the map**

Find the map tab and click on it. It should look like this:



If you had used addresses, you would see a message about the geocoding process, which is slow, about one second per address and, once the geocoding process is finished, you should see the points on a map.

Click on random points and check if the links are clickable. You will notice that all fields aren’t visible in the popup window that opens when you click on a point. With the map open, click on the Change Info Window button. You should see this window:



You can drop some of the columns and add others that makes sense. The Seq\_No may be important for reference and the Quotation, Page\_No, and Line\_No could be good candidates for inclusion in the popup window. Once you have checked the columns you want and unchecked the ones you don’t want, feel change to the custom tab (see above) and add “<p>.<p>” before the </div>. This corrects an issue with the display window that cuts off the last column you are displaying.

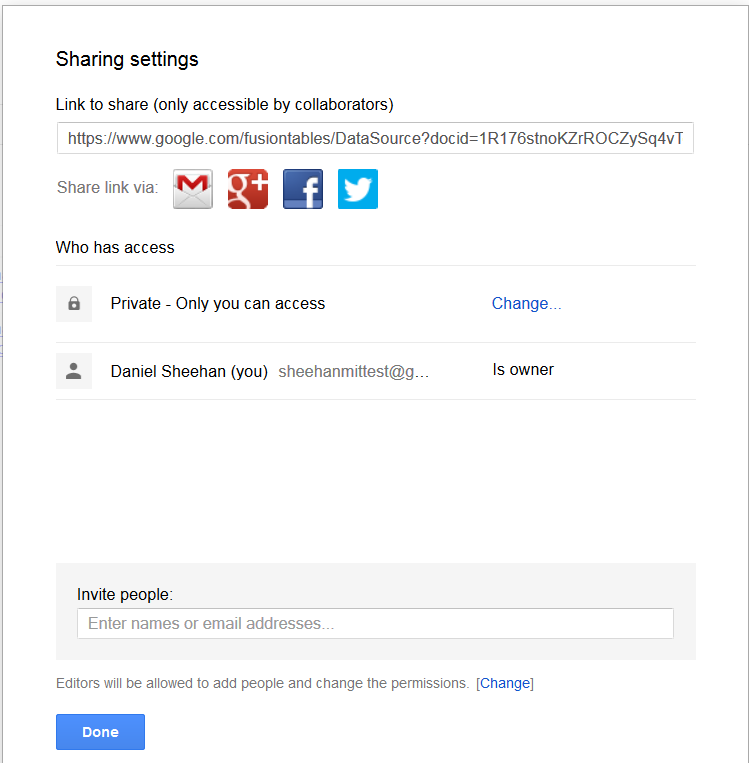
Unfortunately the info window layout does not currently apply when you render the map with the Maps API. You will need to change the columns that are in the table for purposes of web mapping. You need to drop columns in the table, or in a copy of the table that you will use with the Maps API. To do this, go back to the table view (Rows tab) and click on Edit then Change Columns. Hover over the columns you need to delete. You will see an X for delete. Click on the X to remove columns. Remove the Coord\_Dec, Temp\_Lat, Temp2\_Long, Temp1\_Long, and Coord\_DMS columns. When you save this you will be prompted to be confirm that you want to do this. Again, this is for purposes of mapping and having a readable info window. Don’t do this to a master copy of a table that you will want the columns for other purposes.

**Heatmaps**

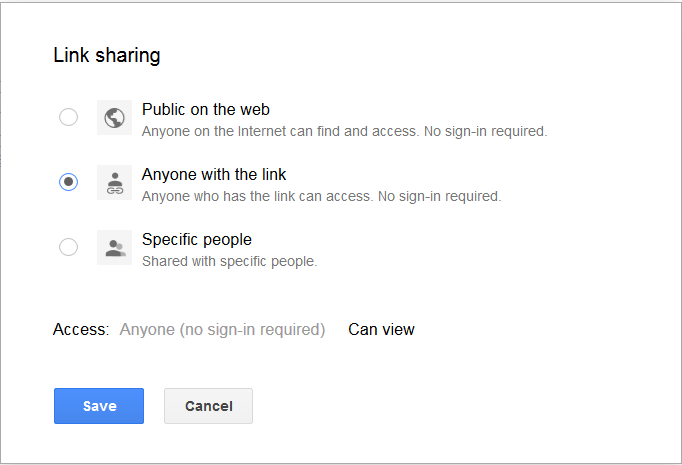
Heatmaps are useful for seeing the density of point data. While in the map view of a table, change the map type from Feature map to Heatmap. Try changing the radius and opacity. The larger the radius, the more points that potentially make an area more visible. Radius is related to pixels, therefore when you zoom in the hot spots are diminished. The opacity changes the color density, blocking the background map. Change the opacity to see how it affects the map.

**Make the table (and map) public so you can use it in the Maps API**

Once you have a map, you can add it to a web map with the Maps API. But first you need to make it public. There are four options – private (the default option), available to certain people, public but unlisted and unsearchable, and public and searchable. You table must be public for it to be visible in a map. Click on the SHARE button to get this interface and change the permissions:



Click on the Change link to see this interface:



For our purpose, change the Link sharing to “Anyone with the link”. It will work with Public on the web but that is not required.

**The free standing web map**

Download the showMD.html file from the Stellar site. If you click on it, it should open in a browser. This sparse version has a Maury map from the 1850s as background. Notice that you can click it on and off.

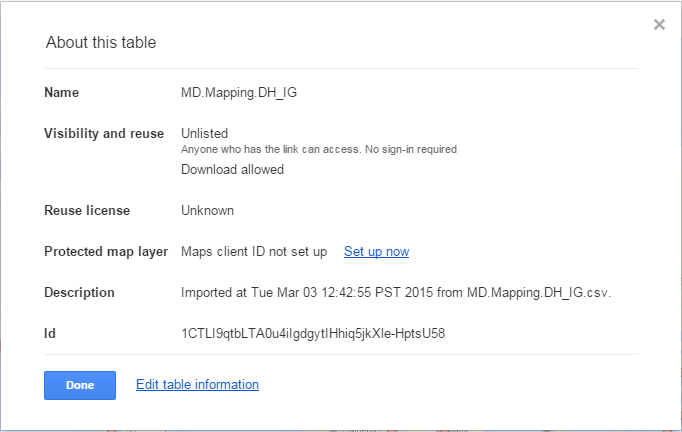
Notice where the map is centered and the zoom level, two items that can be changed to change the appearance of your map.

**Extra credit**

If you have sufficient knowledge of HTML, use the code in the MapOpacity.txt file (on the Stellar site) to change the opacity of the Maury map (one line of the code replaces a line in the current file unchanged and a second line replaces another, this time changed – it should be obvious were it goes if you keep the order of the code in the file). It will be useful in some cases to see the historical map and the current map. Try to create a widget that allows the user to change the opacity of the Maury map at will. This requires some HTML *and* some Javascript. Please disregard this step if you don’t have these skills – it isn’t required for this exercise.

**Add the Fusion table**

Now try adding the Fusion Table you created. Go back to the Fusion Table and click on File then on About This Table. You should see this form:



Write the ID on paper or text file (it won’t copy but you can right click on this in Google Chrome and then search for it, this leaves you with the text that you can copy). Then use this code, substituting your ID for my ID shown below:

FusionLayer = new google.maps.FusionTablesLayer(“1CTLI9qtbLTA0u4ilgdgytIhiq5jkXle-HptsU58”, {suppressInfoWindows: false});

FusionLayer.setMap(map);

This code is in the AddingFusionTable.txt file on Stellar.

**Finished!**

You are finished with the mapping exercise. Feel free to explore CartoDB (account required) or expand on the ideas presented in this exercise.

If you have maps that you want to use as a background for future maps, come to the GIS Lab, Rotch Library. See the website and contact information on page 1 of this document.

**Some Fusion Tables documentation from Google**

This page has information on selection and heat maps for Fusion Tables rendered with the Google Maps API:

<https://developers.google.com/maps/documentation/javascript/fusiontableslayer>

Help for customizing the info window:

<https://support.google.com/fusiontables/answer/171216?hl=en>