Exercise 07
RasterLite, SQL, and Print Composer
Mar. 2, 2018

While QGIS doesn't contain built-in commands that let you store raster data in a spatialite database directly, the “gdal” library which comes with QGIS can be used indirectly to do that. The commands, using the OSGeo4W command line window or via Python, are complex enough that I don't want to explain them here. However once the data (and an appropriate CRS) is saved in a spatialite database using the rasterlite extensions, QGIS CAN read that raster data. For this exercise I've provided my version of the Laramie land additions shapes in one spatialite databases (`exercise_07.sqlite`) and the Laramie map in another (`laramie_map_nad27.sqlite`).

In this exercise we'll load the vector and raster data, we'll use a couple simple “SQL” (Structured Query Language) statements to inquire about contents of the databases, and finally we'll use the QGIS “Print Composer” to generate a nicely formatted version of the map. The print composer lets you add legends, scales, grids, and other components you would want on a final version of your data. You can have a single data project but multiply types of output maps (using different Print Composers) or you can have multiple data projects but use a common Print Composer template for similar maps.

Download the `exercise_07_data.zip` file to your local storage and extract the following three files:

`exercise_07.sqlite` (containing all the vector information)
`laramie_map_nad27.sqlite` (containing all the vector information)
`laramie_land_additions_further_information.xls`

Loading and examining a RasterLite database map

First we'll load the Laramie map -- but this time stored within a SpatiaLite database instead of in the original tiff file. It COULD be stored in the same database as the other shapefiles -- but because it is so much larger, I've chosen to store it in a separate database all of its own. I've also added the CRS information to it so that you do NOT need to manually specify that CRS when you load it.

While spatiaLite <u>vector</u> layers are loaded via the `Layers/Add Layers/Add SpatiaLite Layer` command, it turns out that spatiaLite <u>raster</u> layers are loaded via the `Layers/Add Raster Layer` command we've been using for other raster files. Use `Layers/Add Raster Layer` and select the `laramie_map_nad27.sqlite` file. The map should load without it asking for a CRS. To see what CRS was actually used, in the `Layers` panel right click on the map and select `Properties`. Under `General` you should see a CRS entry. If the right end is cut off, click on the globe button next to the entry, which will bring up a window where you can see the full text. This is the specification for a polyconic projection, with units of meters, using the North American Datum of 1927 (NAD27), and with offsets measured east from 105.5625W longitude, north from the equator. After examining the text, click **Cancel** to avoid changing the definition. Click **Cancel** again to back out of the **Properties** dialog window.

We'll next examine the **laramie_map_nad27.sqlite** database. We need to"connected" to that database first -- and just loading the raster from it unfortunately isn't enough. Open the **Browser** panel (which usually is above or below the **Layers** panel. If it isn't from the main menu click on **View/Panels** or **Settings/Panels** (depending on your system) and add a check in the box before **Browser** to make it visible. Within the **Browser** panel right click on **SpatiaLite** then select **New Connection**. In the dialog which opens select the **laramie_map_nad27.sqlite** file.

Now from the main menu select **Database / DB Manager / DB Manager** . In the left panel click on the > or **+** before **SpatiaLite** then on the **>** or **+** before the **laramie_map_nad27.sqlite** which should appear. You should see a tree showing two "user" tables: **source_metadata and source_rasters** To store the actual data the RasterLite extension to SpatiaLite has broken the large raster array up into a large number of "tiles" (actually 208) each of height and width no more than 256x256 pixels. The **source_raster** table contains the binary "blobs" which hold the actual data. The **source_metadata** table contains information about where each tile is located in space. You can look at general information about the table by clicking on the **Info** tab in the right panel, and you can see the table itself by clicking on the **Table** tab. The information won't be that interesting for **source_raster** or **source_metadata.** However we'll see more detail when we examine one of the vector layer tables later.

Early versions of QGIS also showed many "system" tables, including one we want to examine called the **spatial_ref_sys** but the current version of QGIS hides it since only rarely does the ordinary user needs to access it. However I did in fact need to modify it to add the special CRS for the Laramie map. We'll use an SQL command to display it.

With SpatiaLite databases you can if needed use standard SQL (Structured Query Language) commands to interrogate or modify the database. For example if you wanted, you could create an SQL command which would search for features within a certain distance of a given location, or select features with attribute entries (for example geochron ages) within a certain range. With QGIS there are usually built-in tools or plugins which let you avoid having to create SQL commands yourself

To see just one simple SQL command, within the **DB Manager** window make sure the Laramie map is selected then from this **DB Manager** window click on the menu item **Database/SQL Window**. A new window will open. Into the upper panel type the following SQL command:

```
SELECT * from spatial_ref_sys;
```

(the final semicolon is important, and it is a semicolon, not a colon) then click the **`Execute (F5)`** button.

You are telling it to search the hidden **`spatial_ref_sys`** table and select all rows. The default spatialite database contains thousands of possible coordinate systems but in this table I've deleted all of them except three standard ones and the special Laramie Map one I inserted.

The first column gives a Spatial Reference ID number. The **`auth_name`** column tells who defined this coordinate system. Most are defined by the European Petroleum Survey Group (ESPG). The next column gives that groups ID number for this coordinate system, and the next gives a name. Finally the **`proj4text`** and **`srtext`** columns give two alternate ways of specifying the parameters for this projection. The proj4 text is what we entered long ago when defining the special CRS for the Laramie map. (If you can't see all of that you can drag the vertical … separators between column headings to change the column widths.)

Rows -1 and 0 are generic cartesian (x,y) and geographic (longitude,latitude) coordinate systems. I've kept EPSG 4326 which is just (longitude, latitude) in the WGS84 datum. The last row is the one I added. High number srid's are reserved for individual user's definitions.

Next go back to the upper SQL command window and modify that SQL text to read:

```
SELECT * from spatial_ref_sys WHERE ref_sys_name LIKE "Laramie Map";
```

and hit **`Execute (F5)`**

This command is telling it to select from that table only the row where the ref_sys_name column contains the string “Laramie Map”. Only that row should be displayed now.

Adding SpatiaLite Vector Layers

Next add all three layers which are contained in the **exercise_07.sqlite** database. First connect to that database by repeating for it the steps you did above for the map, beginning with right-clicking on **SpatiaLite** in the **Browser** panel. Once you are connected to it from the main menu use **Layer / Add Layer / Add SpatiaLite Layer**. In the window which opens click on the list of layers near the top and select **exercise_07.sqlite** then click the **Connect** button. In the tree of layers which appears, select all three (holding **CTRL** to select additional ones) then click **Add**.

Back in the **Layers** panel right click on the layers and adjust their order and their style properties (such as transparency) to produce a reasonable display, with the land additions color coded by year_added. While in **Properties** look under the **General** tab to see what **CRS** these layers use. It should be **EPSG:4326 - WGS 84**. Zoom in to be sure that the GPS track really does start and end at the Geology Building front door -- i.e. confirm that the track layer and the map are using their proper CRS's.

If the above shapefiles do NOT seem to be anywhere close to the location of the Laramie map, this may be due to a bug in QGIS which is sometimes confused about how to project these to a common canvas. Usually, right clicking on the Laramie map and selecting **Set Project CRS from Layer** fixes the problem. You might also need to click the **Refresh screen** icon  .

Finally, examine the **exercise_07.sqlite** database using the DB Manager, as we did above for the map. Open **DB Manager** and select that file. You should see tables for **gps_track**, **gps_waypoint**, and **rrh_land_additions**. Select the latter. Under the **Info** tab you should see a list of fields (columns) including the manually created **id** and **year_added**, plus other built-in columns such the **pk** primary index column, and a **geom** column saying whether the feature contains a point, line, or polygon. Under the **Table** tab you can see the attribute table itself.

Finally, once again open the SQL window with **Database/SQL Window**. Enter and execute the following command to select just those rows (i.e. features) which were added in 1963.

```
SELECT * FROM rrh_land_additions WHERE year_added LIKE 1963;
```

Enter the following to select just the id columns for the above rows:

```
SELECT id FROM rrh_land_additions WHERE year_added LIKE 1963;
```

Adding Excel data to the attribute table.

You often create additional data in other programs (such as Excel) which you then need to combine with your GIS attribute table. For example you may have age or composition information to add to the attribute table for your samples. To combine them you need to have one column in the attribute table and one column in your spreadsheet which acts as an index or key. In the above case it would be a common "sample number" column.

From the main QGIS menu use `Layer / Add Vector Layer` to select and add the `laramie_land_additions_further_data.xls` spreadsheet. If you don't see that listed in the file dialog, change the `filter` at the very bottom to `All Files` . Once added the spreadsheet should appear in the `Layers` panel preceded by a spreadsheet icon. Open its attribute table to see the spreadsheet. In this case I have an index column named `id` in both the spreadsheet and the `laramie_land_additions` attribute table. Close the attribute table. Right click on `laramie_land_additions` in the `Layers` panel. Select `Properties` then the `Joins` tab. Click the "+" button and in the dialog which appears, set `Join Layer` to `Laramie_Land_Additions_Further_Information Sheet1`, set the `Join Field` to `id`, and `Target field` also to `id`. Leave checked "`Cache join layer in virtual memory`" and click `OK`. Back in the main **Joins** window again click `OK`. Open the `laramie_land_additions` attribute table to see the combined result. In a real application we might want to save this combined data -- but for now we'll leave it combined only in "virtual memory".

Map Composer

Although we can save an image version of our displayed map using the `Project / Save as Image ...` command usually we need to create a more sophisticated presentation. As a preview of that, from the main menu select `Project / New Print Composer` and give this new composer the name `exercise_07`. (When you eventually save your project, the print composer parameters will be saved in the project file -- unless you take other steps to save it separately.)

In the composer window which opens, first, under the `Composition` tab in the panel on the right, adjust the paper size parameters so that it expects ANSI A (Letter 8.5x11 inch) paper and `Landscape` mode. Enlarge the window and select `View / Zoom Full` to make the canvas match the window. From the main composer menu select `Layout / Add Map` to activate the `Add Map` tool, and drag the cursor across the canvas to create a map of your desired size. You can resize it graphically. If you select the `Item Properties` tab in the panel on the right, you can change the map properties -- for example the scale, extent, etc.

From the main composer menu select `Layout / Add Scalebar` and again drag across the canvas to add that scalebar.

Experiment with adding other items. To change the properties of a previously added item, select it, then adjust values in the `Item Properties` panel tab on the right.

In order to line up items you want to place on the Map you can use Guides and Grids. First, to make sure the grid is visible, under the `View` menu make sure `Show Grid` is checked. If you don't see an obvious grid, go to the main menu's `Settings / Composer Options`, and in the `Grid appearance` section set the `Grid style` to `Solid` and the `Grid Color` to something which stands out. (For real work you will probably want a more subtle grid appearance.) While *default* grid properties are specified here, you adjust the actual Grid properties in the `Composition` tab on the right, in the `Guides and Grids` section. In that `Composition` panel, adjust the `Grid spacing` and `Snap tolerance`, turn on `Snap to Grid` under the view menu, and see how that affects placement of features like labels which you try to place on the map. For example try placing an arrow on the map, using either the icons on the left or the `Layout / Add Arrow` menu. With snap enabled, as when the mouse gets within the specified tolerance of a grid point, it snaps precisely to that location. Also experiment with `Layout/Add Legend` and other features.

From the `Composer` menu item you can print or export to various file formats such as pdf. Save a pdf version of your map for submission next Friday.

After exiting Composer, save your QGIS Project in case you need to return to make any changes later.