QStudio/PRQL Quick Start #2 - Playing with QStudio

Lesson #2 tells how to install QStudio, download and open the *Property_in_Lyme* SQLite database, and then start making your own queries.

1. **Install QStudio**

   On a Mac:

   - Download the Mac version of QStudio from: https://randomneuronsfiring.com/wp-content/uploads/QStudio.zip
   - Unzip the result to get the **QStudio.app** application
   - Double-click the application. You will see a "Can't open - move to Trash" or "unverified developer" (or similar) warning. Use **System Preferences -> Privacy and Security** to indicate that you trust the application. Click "Open anyway" if necessary. QStudio opens.

   On a Windows machine:

   - Download the **Installer for Windows** from the QStudio Downloads page: https://www.timestored.com/QStudio/download
   - Run the installer in the usual way
   - Using the command line, enter `winget install prqlc` to install the PRQL compiler.
   - Open QStudio

2. **Register for a free QStudio license** When QStudio starts up, it requires you to register for a free account. Provide your email address and you will be taken to a page that displays a license code (a long string). Copy that string and paste it into the QStudio window.

3. **Download the "Property in Lyme" SQLite database** The latest copy of the database is always available from: https://raw.githubusercontent.com/TaxFairness/TaxFairness/refs/heads/main/Property_In_Lyme.sqlite

4. **Open the "Property in Lyme" SQLite database in QStudio.** To do this, drag the database's icon into the QStudio window (top left corner, but the bottom half of that pane).

5. (You might also download the Chinook database that is frequently used for examples. You can also drag the Chinook database to QStudio - it will work properly when both are opened.)

Playing with QStudio

QStudio is now running. Here are some hints for trying its features.

View the tables

Click any of the names of tables at the top-left of the window. The columns and rows appear at the bottom of the window. Interesting tables are:

- **CleanScrapedData** - a cleaned-up list of every parcel in Lyme, gleaned from information from the Vision database along with easement and frontage information.
- **GraftonCtyRoD** - all records from Grafton County Register of Deeds back to April 2019.
- **LymeOldToNew** - data from the (original) 2024 MS-1. I will update when we are permitted to view the info.

Create and edit queries

The basic process is:

- Create a new file in QStudio (**File -> New File**)
- Save as `QStudioTest.prql` (".prql" suffix is important) using **File -> Save As...** *I recommend you save it to a new folder on the Desktop so it's easy to find later.*
- Enter a query. Start simple with `from LymeOldToNew` - this statement just displays all the rows and columns from the `LymeOldToNew` table.
- Save the file with **Cmd-S** on Mac; **Ctl-S** on Windows
- Execute the query (**Cmd-E** / **Ctl-E**)
- *You may be prompted with a request to install the "SQLite driver". Allow it to happen.*
- The result from the query appears in the bottom "Result" pane. If the Result pane is not visible, choose **Windows -> Result**
- Edit the query by typing in the window. Save and Execute it as above.

Enhancing your query

Here are examples taken from the first lesson. You can experiment with QStudio by modifying the queries as shown below:

> Note - these examples use the PRQL language (pronounced "Prequel"), not SQL. The PRQL Reference Book gives full details on creating queries.

Select two columns from the `LymeOldToNew` table, ignoring the rest

Do this by adding a `select` statement at the bottom. Note that the `select` statement requires a list enclosed within `{ ... }` that has column names ("LO_Location" and "LO_Owner") separated by ",".

```
from LymeOldToNew
select {
   LO_Location,
   LO_Owner,
}
```

Here's the way to think about the query above. The statement `from LymeOldToNew` starts a "pipeline" with the contents of that table. It passes it to the next statement (`select ...`) that modifies the table, by retaining only the two named columns (with all the rows) that could be passed along to any subsequent statement.

Note that PRQL doesn't care about line breaks. A statement can be on one or many lines: the following is equivalent to the above: `select { LO_Location, LO_Owner }`

Select a couple more columns

Add more column names between the `{ ... }`, separated by commas. Trailing commas are allowed, even if no other column name appears afterwards.

```
from LymeOldToNew
select {
   LO_Location,
   LO_Owner,
   LO_OldValue,
   LO_NewValue,
   LO_Year,
}
```

The QStudio result now shows five columns, the first two plus the three new ones. The Result pane shows them in the order they are listed in the query.

Create a new column named "difference" of the new to old values

To do this, add a new line with the column name ("difference") followed by `=`, and then the calculation.

```
from LymeOldToNew
select {
   LO_Location,
   LO_Owner,
   LO_OldValue,
   LO_NewValue,
   LO_Year,
   difference = LO_NewValue - LO_OldValue,
}
```

Note that the new "difference" column in the result is just as "real" as any of the original "LO_xxx" columns.

Note too, that the comma at the end of the line is optional, but recommended because you can comment out any line (see below) without hassle.

OK. That's all very interesting, but which property had the biggest change? Let's sort by the "difference" field. Add this line to the end, then re-execute the query:

```
sort {difference}
```

This sorts the rows by the value in the "difference" column. To reverse the sort, use `sort { -difference }`.

Oh, but that shows many different years. Let's just look at 2024. Add this line to the end:

```
filter (LO_Year == 2024)
```

This includes any rows where the LO_Year is equal (`==` *) to 2024. (The double-equal is required.) To exclude the year 2024, you can use "not-equal"* `!=`*. And the* `>`*,* `>=`*,* `<`*, and* `<=` *operators work as expected.*

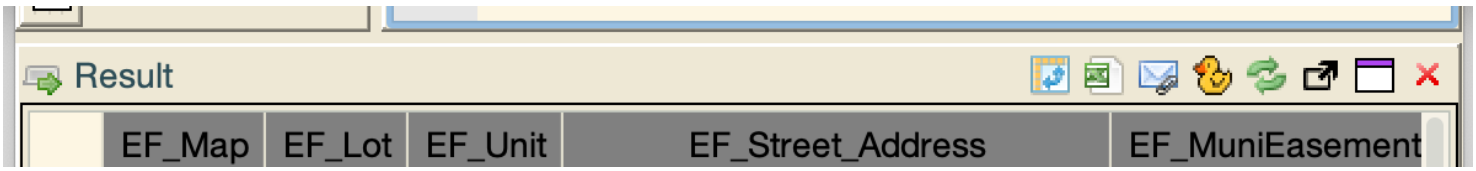
Commenting out lines

Sometimes you want to temporarily (or permanently) remove a line or group of lines from a query. PRQL uses `#` as the comment character. When it appears, the remainder of that line will be ignored. There is a keyboard shortcut - **Cmd-/** (Mac) or **Ctl-/** (Windows) - that comments out/uncomments the lines that are selected.

Export to Excel

One cool feature of QStudio is that you can export the results directly to Excel. The Result pane opens directly in a new workbook. To do this, click the small Excel icon in the top-right of the window. It's the second icon on

the right side of the image below.



Further observations

1. PRQL statements "transform" the table they receive by modifing it in some way and passing it along to the next statement. This is the power feature of PRQL: the statements can be "stacked" to form a pipeline that continually modifies the data as it passes through to produce a final result.

2. The `filter` statement could have been placed much earlier in the query. In fact, it could have immediately followed the initial `from ...` That would have removed all the other years prior to passing them to the first `select` - and in the end, the result would have been identical.

3. Similarly, the `sort ...` statement could have been placed anywhere in this pipeline. From that point in the pipeline, all the rows would have remained in the same order.

4. Finally, a bit of "PRQL lingo":

    - Since PRQL statements "transform" a table, they're also called *transforms*.
    - Tables are also called *relations*.
    - A *tuple* is the PRQL term for column names that are enclosed in `{ ... }`. For example, it's proper to say the `select` transform requires a "tuple" (instead of requiring a "list").

Looking ahead

Lesson #3 talks about some tricky bits you may encounter when using QStudio and PRQL.

There are a few other PRQL transforms (`join` , `aggregate` , `take` , `group`) that will be covered in Lesson #4.