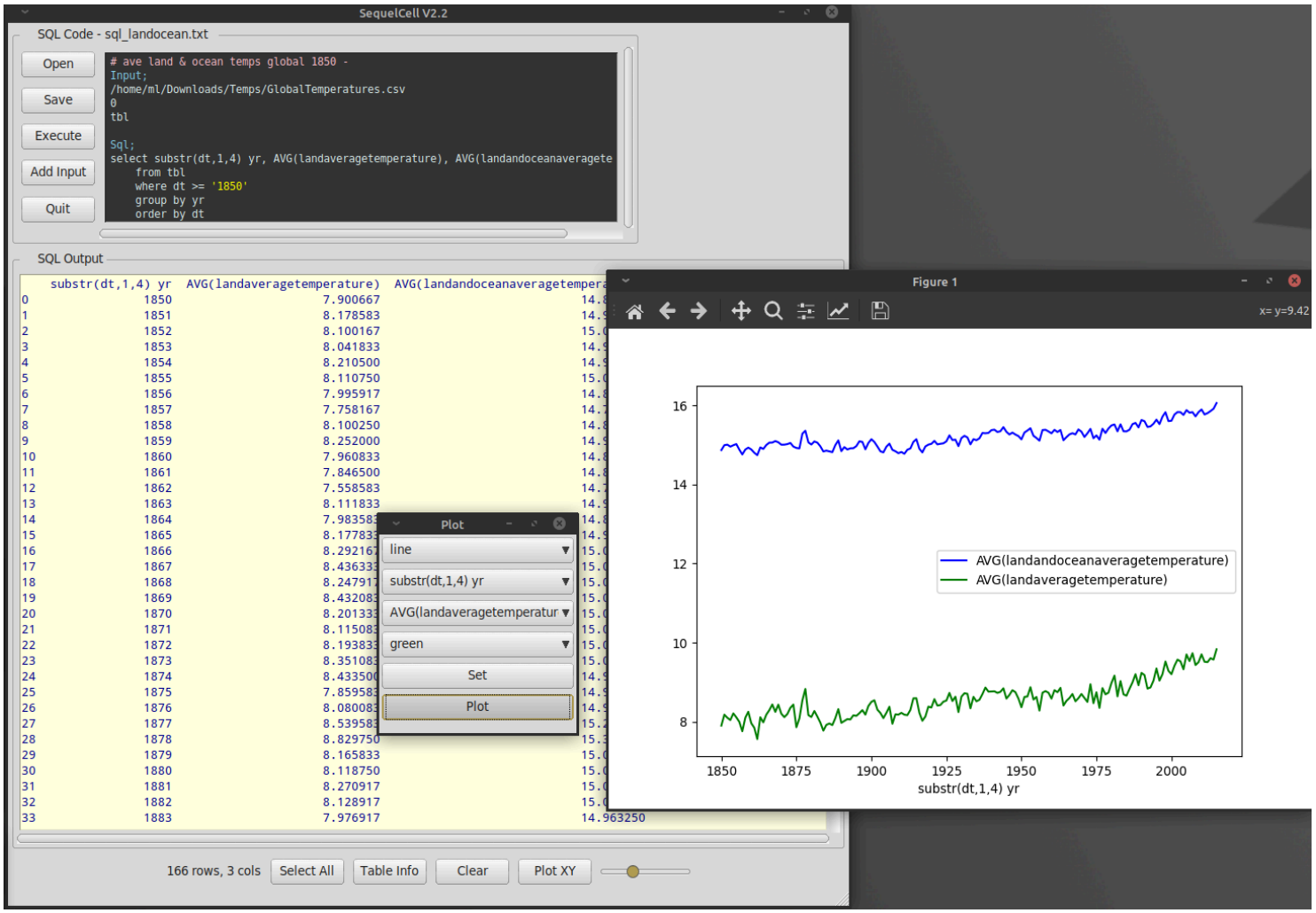
SQLcel (SequelCell)

SequelCell is a desktop application that allows users to combine data from various input types, including Excel spreadsheets, CSV files, and SQLite databases. It can also output the query results to a file in any of the three formats. SequelCell can be run as a GUI app or from the command line. It does not support running SQL update or insert commands.

SequelCell can produce simple charts for data visualization.

SequelCell **does not** run SQL "update" and "insert" commands.



Installation

Download the installation from https://www.python.org/downloads/ or https://github.com/MLeidel/sqlcel

Extract the folder from the zip file. Then in the sqlcel_main folder run:
`pip install -r requirements.txt`

Using SequelCell

Required Python 3.x modules:

ttkthemes
sqlalchemy
matplotlib
pandas
xlrd
openpyxl
lxml

Run as GUI app

> python sqlcel.py

Run in CONSOLE

> python sqlcel.py *sqlCodeFile.txt*

The SQL code file

Each SQL query is coded into a text file where the *inputs*, optional *output*, and *sql* statement are defined. The three tags: "**Input;**", "**Output;**", and "**SQL;**" are case insensitive, but must appear on a single line. The ";" is optional. You can have more than one "**Input**" statement, but only one Output. "**Input**" and "**Output**" statements must appear before the "**SQL**" statement.

Input;

defines the Excel, CSV, or Sqlite input file path, with sheet name or number, or database table name, and the SQL table name to use in the

query. You can have multiple "**Input;**" statements.

Example:

```
Input
      insurance/sales.xlsx
      0
      sal
```

To insert a new **"Input;"** section use the **Add Input** button.
Code is inserted at the cursor location in the code frame.
The **"Input;"** tag must come before the "**SQL;**" tag.
The button inserts a 0 for sheet number/name and “tbl” for the SQL table name.
So, you will usually have to code the actual "sheet" (name or number) and SQL table name that you want to use in the query.

The "0" seen on the line after the input file path is the sheet number.
Sheet numbers (zero relative) only work with spreadsheets. Instead of a number you can type in the actual sheet name if you want to.
Each "Input;" section will load only 1 sheet (or table.)
The workbook may contain more than one sheet, and you can access another sheet from the workbook by defining another **"Input;"** statement.

For CSV files the "0", sheet name or number, is simply ignored.

For Sqlite databases change the "0" to the name of the database table you wish to access.

The third line inserts "**tbl**" when using the "**Add Input**" button. This is the table name that will be used in the SQL statement for this input. Change this to whatever you want to use for a table name in the SQL select statement.

The **Add Input** button always inserts a *full path* to the selected file.
You may adjust this path to a relative path from the sqlcel folder or use the full path.

Output; *optional*

defines the *optional* Excel, CSV, or Sqlite output file path.

You can have only a single "**Output;**" statement, and it is optional.

Example:

```
Output
insurance/profits.db
```

The optional "**Output;**" tag may be included anywhere above the "**SQL;**" tag.

Under the "**Output;**" tag type in the fullpath for your output Excel, CSV, or Sqlite database ***file***. When using Excel as the output target file, the output sheet name will always be "**sheet1**". When using Sqlite for output file the table name will always be "**table1**". Use ".db" for the Sqlite output file's extention.

Datacols; *optional*

Reformates a date column into a standard date time format:

YYY-MM-DD HH:MM:SS

Example:

```
Datecols
datefield1,date2,active_date
```

SQL;

indicates the SQL select statement is to follow.

Example:

```
Sql
select sal.Item, sum(Amount), sum(Cost), count(*),
sum((Amount - Cost)) Profit
      from sal, cst
      group by sal.Item
      order by Profit
```

Everything beyond the **"Sql;"** tag is part of the sql select statement.

# indicates a comment line (column 1 only)
Blank lines and comments are discarded when
the code file is parsed. Blank lines and comment lines
Are permitted anywhere throughout the sqlcel file.

Example of a simple code file:

```
# sql_sample.txt
# Find sales amount by month
```

```
Input
sales/orders.xlsx
0
ord

SQL
select sum(purch_amt) from ord
    group by ord_date
```

In this example orders.xlsx resides in the 'sales' directory.
The 'sales' directory shares a directory with sqlcel.py.
The **Input;** and **Output;** statements can also use fully
qualified paths. The input target table will be the first sheet
“0” in the orders.xlsx file. The SQL code will reference this as table “ord”.

Preview columns & row data

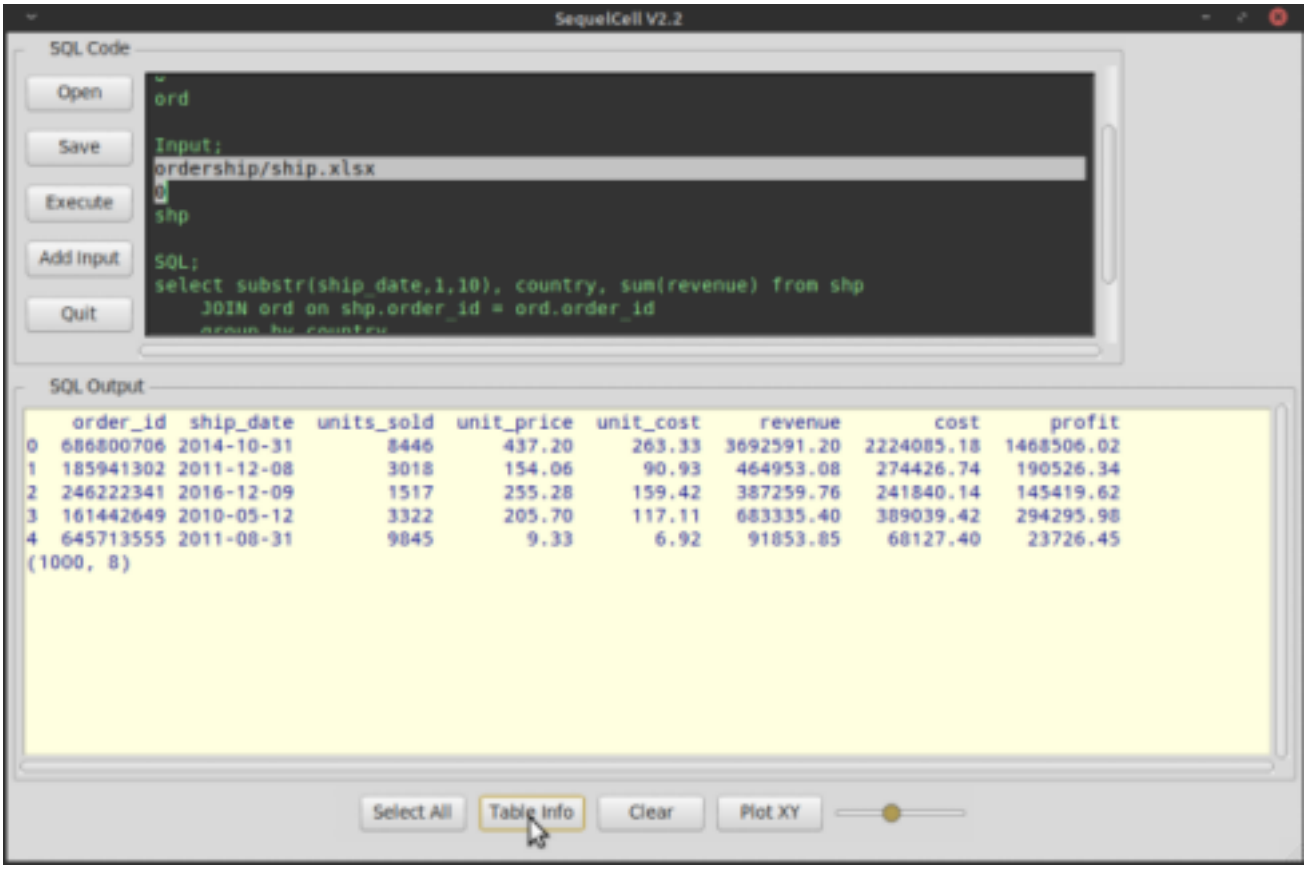
Select the (line) with the Excel or CSV file path in the code frame.
For Excel select both the path line and the sheet line.
Then click the **Table Info** button to view the columns
and 5 rows of data in the output frame.

With the file path line and sheet line selected,
click the **Execute** button to *preview the entire sheet/table*.

Previewing is not implemented for Sqlite database tables.

With nothing selected in the code frame clicking the **Execute** button
runs the SQL statement showing the results in the output frame.

Selecting the input file path and sheet number to return table info.



Run the SQL

With nothing selected in the code frame click the **Execute** button to run the SQL statement. The results will appear in the "SQL Output" frame. The "Select All" and "Clear" buttons at the bottom of the app pertains to the "SQL Output" frame.

Use the **Open** and **Save** buttons to load and save the SQL code. Use **Ctrl-S** to quickly save the loaded code file. **Ctrl-q** and **Esc** will prompt to exit the app.

The "**Plot**" button lets you select one or more X/Y columns to graph. After setting each *type, axis, and color,* click *Set*. Click *Plot*

Style Options

For some style variations backgrounds, foregrounds, and fonts can be tweaked in the sqlcel.ini file.

```
# sqlcel.ini

# after editing this file - restart sqlcel.py

# ----------------------   -----------------


# SQL CODE frame Text

Font = Roboto Mono

Size = 9

Backg = #333

Foreg = #FFF

Tab = 4

Cursor = white

Remark = lightgreen

Section = deepskyblue

Literal = yellow

Number = skyblue


# SQL RESULT frame Text

Ofont = DejaVu Sans Mono

Obg = lightyellow

Ofg = darkblue

# THEME

# Windows: xpnative

# clam, scidblue, radiance, scidgrey, alt, default

WinTheme = scidblue
```

Shorcuts:

Control-s	Save
Alt-s	Save As
Control-q	Exit
Control-a	Select All
Escape	Exit
Control-e	Execute
Control-o	Open Sql File
Control-i	Insert Data Source