

# SequelCell

SequelCell is a desktop app lets you easily combine data from several different input types and assess them using standard SQL language statements.

Input types:

- Excel spreadsheets
- CSV files
- SQLite database tables

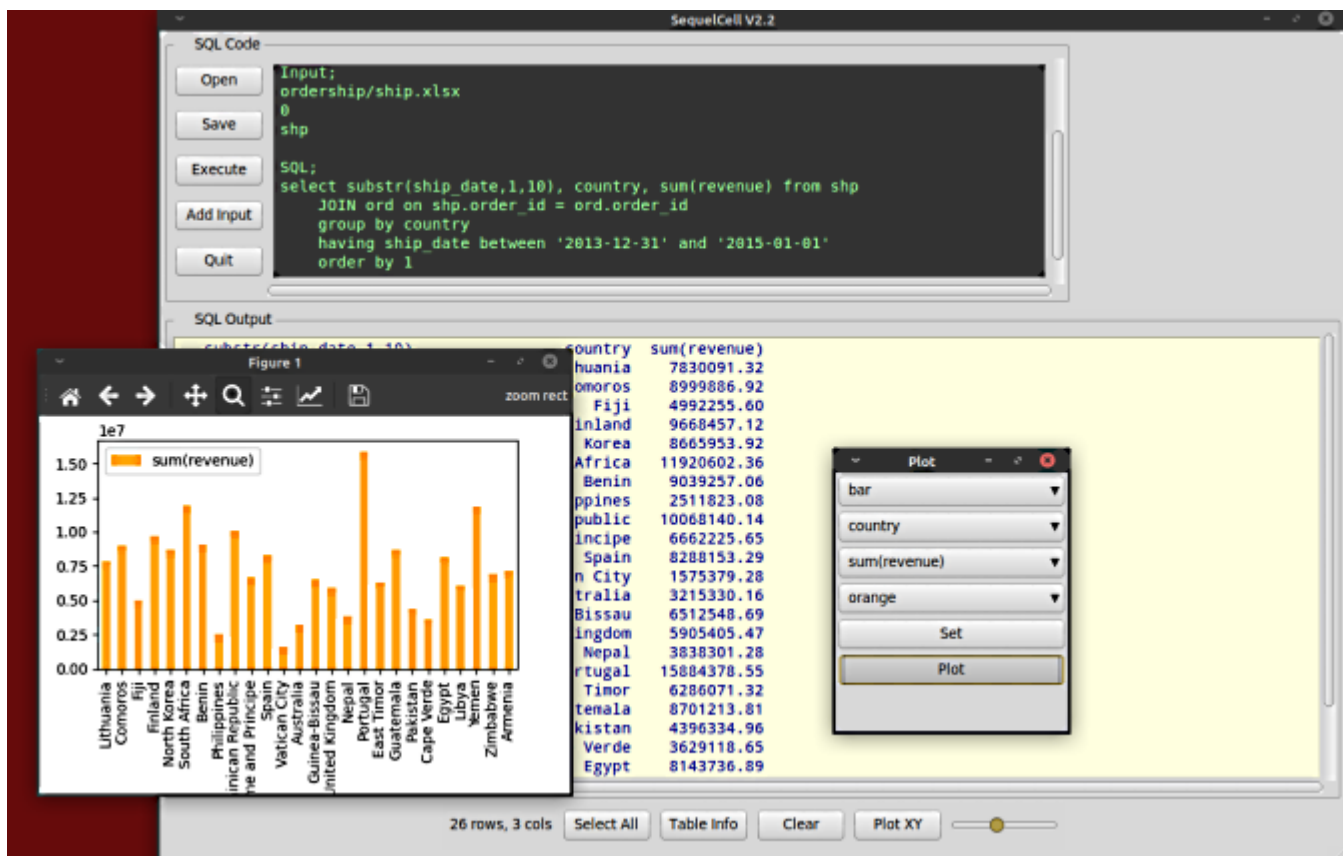
Also you can output the query result to a file in any of the three formats.

Furthermore, multiple sheets, from an Excel workbook, and multiple tables from an SQLite database can be input and accessed together with other inputs by the SQL.

SequelCell can produce simple charts for data visualization.

SequelCell (sqlcel.py) may be run as a GUI app or from the command line.

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



## Using SequelCell

SequelCell has a few Python system requirements:

- Python 3.x
- modules:
  - tkinter
  - ttkthemes
  - pandas
  - matplotlib
  - sqlalchemy

## 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 and end with the ";" semi-colon. You can have more than one "**Input;**" statement. "**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. 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.

Note, you may have to code the actual "sheet" (name or number) and "table" name that you want to use in the sql query.

The "0" seen on the line after the input file path is the sheet number.

Sheet numbers (zero relative) only work with Excel 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/table.

The workbook may contain more than one sheet, and you may 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 inserted defaults to "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.

### **Output;**

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

You can have just 1 "**Output;**" statement.

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 extension.

### **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 (only at beginning of line)  
Blank lines and comments are discarded when  
the code file is parsed.

Command lines and blank lines are permitted anywhere throughout file.

### **Example of a simple code file:**

```
# sql7.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.

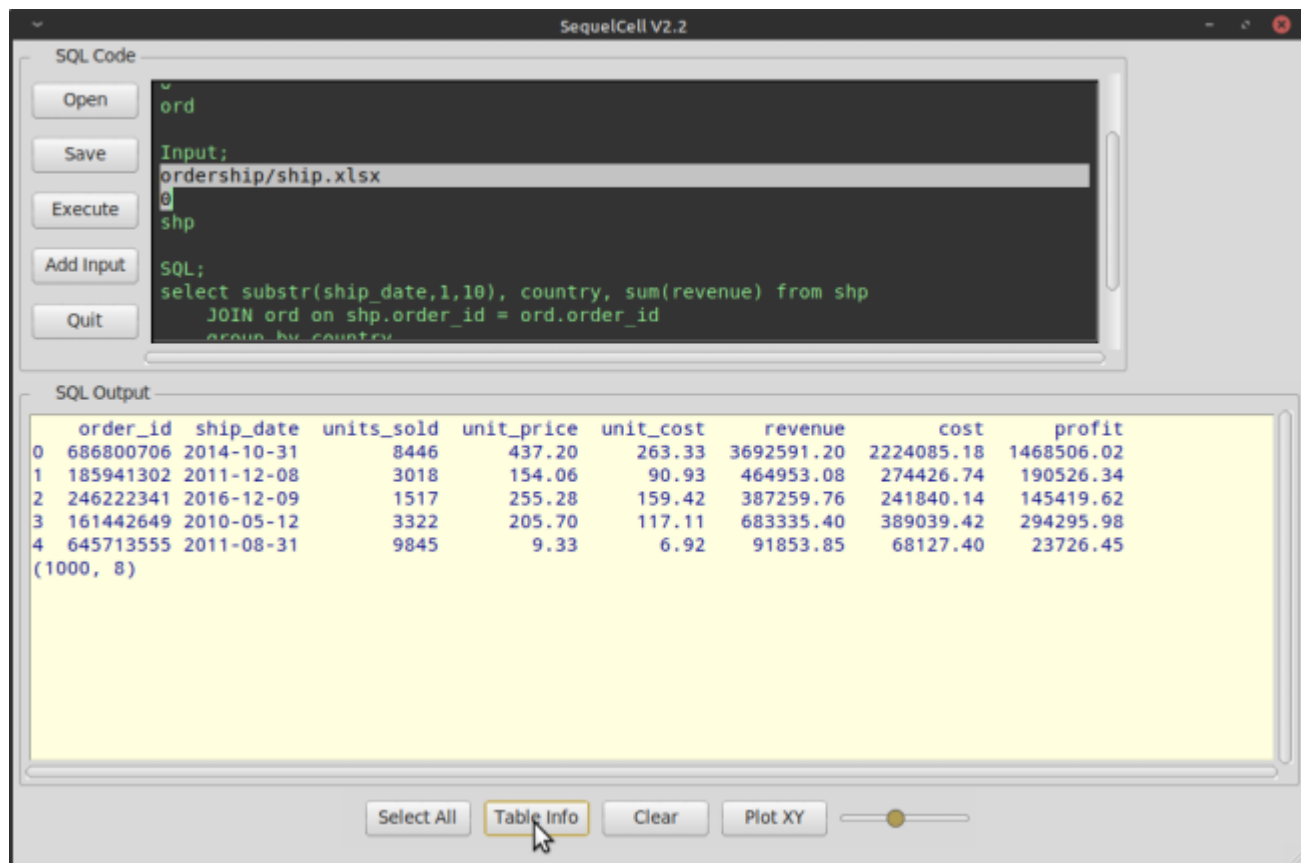
## Preview columns & row data

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.



SequelCell V2.2

SQL Code

Open

Save

Execute

Add Input

Quit

```
ord  
  
Input;  
ordership/ship.xlsx  
0  
shp  
  
SQL;  
select substr(ship_date,1,10), country, sum(revenue) from shp  
JOIN ord on shp.order_id = ord.order_id  
group by country
```

SQL Output

	order_id	ship_date	units_sold	unit_price	unit_cost	revenue	cost	profit
0	686800706	2014-10-31	8446	437.20	263.33	3692591.20	2224085.18	1468506.02
1	185941302	2011-12-08	3018	154.06	90.93	464953.08	274426.74	190526.34
2	246222341	2016-12-09	1517	255.28	159.42	387259.76	241840.14	145419.62
3	161442649	2010-05-12	3322	205.70	117.11	683335.40	389039.42	294295.98
4	645713555	2011-08-31	9845	9.33	6.92	91853.85	68127.40	23726.45
	(1000, 8)							

Select All Table Info Clear Plot XY

## 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 refer 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* after setting up all of the coordinate variables.

## Style

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

```
# sqlcel.ini

# These are for the SQL Code frame Text
Font = Consolas
Size = 11
Backg = #333333
Foreg = lightgreen
Tab = 4
Cursor = lightgreen

# These are for the SQL OUTPUT frame Text
Ofont = Noto Mono
Obg = lightyellow
Ofg = darkblue

end
```