sqawk

or

CLI SQL for CSV:-)

i.e.

making some shell tasks on CSV files easier

Comma-separated Values

→ *lingua franca* of tabular data (e.g., relational databases)

Simple CSV file tasks

CSV is text $\Rightarrow$ work in the shell

a. extract all rows of `myfile` where the value in field #3 is greater than 12.5

b. join `file1` with `file2` on a common field

Shell one-liners

a.
```
$ awk '$3 > 12.5' < myfile
```

b.
```
$ join file1 file2
```
supposing both files are sorted on the join field, in this case #1

Similar tasks

a′. extract all rows of `myfile` where the value in column 3 is above average

b′. join `file1` with `file2`, but on a composite join field (e.g. hospital ID + patient ID)

c′. join more than two files (e.g. genotypes, co-variates, eigenvalues)

No one-liners!

a′. `awk` must have read all of column 3 to compute average

b′. `join` needs both files sorted (extra steps, destroys order); can't use > 1 field

c′. `join` cannot handle > 2 files

Personal postulate

There is a category of tasks that can *almost* be done with one-liners, but not quite

This is a shame ☺

And as a matter of fact. . .

As Database Operations

. . . you can express them as a single SQL query:

a′.
```
SELECT * FROM t1 WHERE f3 >
(SELECT avg(f3));
```

b′.
```
SELECT t1.* FROM t1 JOIN t2 USING
(f1,f2,f3);
```

c′.
```
SELECT ... FROM t1 JOIN t2 ...
JOIN t3 ...;
```

where tables (`t1`, etc.) contain files, columns (`f1`, etc.) contain file fields

but. . .

to do SQL queries on a file, you need:

- to create a database
- to create a table for the file
- to import the file’s data into the table
- (usually) a SQL server

. . .which is a bit unwieldy.

in other words

	pros	cons
shell 1-liners	concise	limited
SQL	expressive	awkward

in other words

	pros	cons
shell 1-liners	concise	limited
SQL	expressive	awkward

→ Can we have both the concision and the expressiveness?

Wish List

the successful candidate will:

- automatically create a database
- automatically create db tables from CSV files
- automatically import content into the tables
- run a SQL query
- print out the result
- be a shell filter

Anything Out There?

SQL-Powered Awk Functions for accessing MySQL databases from Awk

ShellSQL Programs that enable shell scripts to connect to SQL engines

→ not exactly what I want

SQLite

- C library (not server)
- small, fast

$\rightarrow$ if we can automate table creation and population, we’re done.

Results

sqawk

(or "squawk", need a better name anyway. . .clisql?)

1. creates a database (memory -> transient)
2. for each CSV file (or stdin):
   - 2.1. examines first few lines
   - 2.2. creates a table with appropriate names and types
3. runs the SQL query
4. prints out the result rows

Syntax

```
$ sqawk [opts] ([file opts] file)+
SQL
```

Selected options:

- `-i`: specifies index fields
- `-p`: specifies primary key
- `-q`: shows the generated SQL

Examples

a′.
```
$ sqawk myfile.csv 'SELECT *
FROM myfile WHERE f3 > (SELECT
avg(f3))'
```

b′.
```
sqawk file1.csv file2.csv 'SE-
LECT * FROM file1 JOIN file2 USING
(f1,f2,f3);
```

Checking for valid IDs

- file `valid`: list of valid IDs
- file `dubious`: uncertain IDs (among other data)

```
$ ./sqawk valid dubious 'SELECT *
FROM dubious WHERE dubious.spc NOT IN
(SELECT spc FROM valid)'
```

Conclusion

- sqawk makes working with CSV and CSV-like files easier tasks

In an ideal world:

- data is consistently formatted
- data formats are compatible
- data is validated before use
- . . .

In the real world. . .

- data is messy
- there is a plethora of incompatible formats
- nobody checks the data before sending it :-)
- . . .

what can be done?

- export to CSV
- write code to systematically check the data
- . . .

$\Rightarrow$ this is where sqawk might help.

That's all

Thanks