ToiletPaper #140

jambit WHERE INNOVATION WORKS

Executing SQL queries without databases on CSV files using q

Authors: Fionn Fuchs & Maximilian Konzack / Working Student & Software Engineer / Office Leipzig

✕ Problem

Have you ever stared at a CSV file on your screen and wished it was a database executing SQL queries? We've often found ourselves wanting not to directly process CSVs but favoring to run SQL queries.

✔ Solution

With q [1], conventional SQL queries can be executed on CSVs and other tabular files without connecting to a database or to an SQLite database. If needed, q permits exports to an SQLite database. Q supports all common SQL queries types. It is remarkable that q can be used as a Unix command. By default, q reads from STDIN and writes to STDOUT, which allows piping with other Unix commands.

Installation:
All major operating systems are supported and can be installed as usual [2].
Disclaimer: We have only tested q via homebrew on macOS.

```
brew install q
```

➔ Examples

Example 1

Let's take the following table as CSV file:

```
ID,name,yearofbirth,haircolor
1,Max Mustermann,1994,brown
2,Lars Agne,1983,blonde
3,Otto Normal,1995,brown
4,Lieschen Müller,1987,black
5,John Doe,1980,brown
```

We would like to find out how many people have brown hair. This is obviously trivial with such a small table, but in real data sets this would be much more tedious and laborious.

Using q, we directly execute a SQL query on this CSV file to filter by hair color. We expect ; to delimit and that the first line of the CSV describes the column names. Therefore, we use -d to set the delimiter and -H to skip the first line.

```
$> q -H -d ";" "SELECT COUNT(ID) FROM persons.csv WHERE haircolor = 'brown'"

- Result -
3
```

Example 2

Q can also join multiple CSV files on-the-fly. For this purpose, we take this CSV file in addition to the CSV file from example 1 (see above):

```
ID,email
1,max.mustermann@example.com
2,lars.agne@example.com
3,otto.normal@example.com
4,lieschen.mueller@example.com
5,john.doe@example.com
```

Now we want to link both files to output all e-mail addresses of people with brown hair. It would look like this:

```
$> q -H -d ";" "SELECT persons.name, emails.email FROM persons.csv persons
            JOIN emails.csv emails ON (persons.id = emails.id)
            WHERE persons.haircolor = 'brown'"

- Result -
Max Mustermann    max.mustermann@example.com
Otto Normal       otto.normal@example.com
John Doe          john.doe@example.com
```

Example 3

Since q treats text as data, we can also process output from Unix commands with q. For example, you can do the following in q to list all the current process commands that were started on December 24:

```
$> ps aux | q -H "SELECT COMMAND FROM - WHERE STARTED = '24Dez20'"

- Result -
/System/Library/CoreServices/Santa
...
```

Δ Performance and Limitations

The current version of q is significantly faster in running SQL queries on CSV files than comparable Go-based tools, such as Textql and Octosql [3]. We would be interested in comparing q with xsv, which can index and process CSVs. However, we noticed that q does not allow, for example, to execute FROM on a subquery. Another limitation is that q uses SQLite as its single SQL dialect. Further limitations can be found on q's website [1].

✚ Further Aspects

- [1] http://harelba.github.io/q/
- [2] http://harelba.github.io/q/#installation
- [3] https://github.com/harelba/q/blob/master/test/BENCHMARK.md
- [4] https://github.com/BurntSushi/xsv