csvkit CSV command-line toolkit

START HERE

in2csv *<file.xlsx>*
Convert to CSV

csvlook *<file>*
Read it as a table

csvcut -n *<file>*
List columns

csvcut -c *<cols>* *<file>*
Pick columns

csvgrep -c *<col>* **-m** *<text>* *<file>*
Filter rows

csvsort -c *<col>* *<file>*
Sort rows

csvstat *<file>*
Summarize columns

csvsql --query "*<sql>***"** *<file>*
Query with SQL

in2csv / sql2csv xlsx, json, db → CSV → **csvclean / csvcut** fix, select → CSV → **csvgrep / csvsort / csvjoin** filter, order, merge → CSV → **csvlook / csvstat / csvjson** read or export

CONCEPTS

Every tool reads a file or stdin and writes CSV to stdout, so chain them with pipes. Only csvlook, csvstat and csvjson print other formats.

Column arguments take names or 1-based indices: -c 1,3 or -c name,score. Ranges like -c 2-4 work.

Type inference reads the whole file by default; add -I on big files to skip it and keep values as text.

INPUT AND FORMAT FLAGS

-d *<char>*
Input delimiter (e.g. -d ';')

-t
Input is tab-separated

-e *<enc>*
Input encoding (e.g. latin1, cp1252)

-H
Input has no header row
Columns are then named a, b, c, ...

-I
Skip type inference, keep text as is

CONVERT IN (IN2CSV, SQL2CSV)

in2csv *<file.xlsx>*
Convert Excel to CSV

in2csv -n *<file.xlsx>*
List sheet names

in2csv --sheet "*<name>***"** *<file.xlsx>*
Convert one sheet

in2csv -f json *<file>*
Convert JSON (array of objects)

in2csv -f ndjson *<file>*
Convert newline-delimited JSON

cat *<file.xlsx>* **| in2csv -f xlsx**
Convert from stdin (format required)

sql2csv --db *<url>* **--query "***<sql>***"**
Run SQL, get CSV

SELECT AND FILTER

csvcut -n *<file>*
List column names and numbers

csvcut -c *<cols>* *<file>*
Keep columns by name or number

SELECT AND FILTER (cont.)

csvcut -C *<cols>* *<file>*
Drop columns

csvgrep -c *<col>* **-m** *<text>* *<file>*
Rows where the column equals text
-m is an exact match, not a substring; use -r for patterns.

csvgrep -c *<col>* **-r '***<regex>***'** *<file>*
Rows matching a regex

csvgrep -c *<col>* **-i -m** *<text>* *<file>*
Invert: rows that do not match

csvgrep -a -r '*<regex>***'** *<file>*
Match in any column

SORT AND COMBINE

csvsort -c *<col>* *<file>*
Sort by column

csvsort -r -c *<col>* *<file>*
Sort descending

csvjoin -c *<col>* *<a>* *<b>*
Inner join on a shared column

csvjoin --left -c *<col>* *<a>* *<b>*
Left outer join

SORT AND COMBINE (cont.)

csvstack *<a>* *<b>*
Append files with the same columns

csvstack -g *<x>*,*<y>* *<a>* *<b>*
Add a group column per input

csvstack --filenames -n source *.csv
Add each file name as a column

INSPECT AND SUMMARIZE

csvlook *<file>*
Pretty-print as a table

csvstat *<file>*
Type, nulls, min, max, mean per column

csvstat -c *<col>* *<file>*
Stats for one column

csvstat --count *<file>*
Row count only

SQL

csvsql --query "*<sql>***"** *<file>*
Query a file; the table is named after it
From stdin the table is called stdin.

SQL (cont.)

csvsql --query "*<sql>***"** *<a>* *<b>*
Query several files, join them in SQL

csvsql *<file>*
Print CREATE TABLE for the inferred schema

csvsql --db *<url>* **--insert** *<file>*
Load into a database

OUTPUT AND CLEANUP

csvjson *<file>*
Convert to a JSON array

csvjson -i 2 *<file>*
Indented JSON

csvformat -T *<file>*
Re-emit as tab-separated

csvformat -D '*<char>***'** *<file>*
Re-emit with another delimiter

csvclean --length-mismatch *<file>*
Report rows with the wrong column count
Since 2.0 a check or fix flag is required. Errors go to stderr, good rows to stdout.

OUTPUT AND CLEANUP (cont.)

csvclean --fill-short-rows *<file>*
Pad short rows with empty cells

RECIPES

Excel sheet to a quick look

```
in2csv -n <report.xlsx>
in2csv --sheet "Q3"
<report.xlsx> > q3.csv
csvcut -c name,total q3.csv |
csvlook
```

Top ten by a numeric column

```
csvsort -r -c total <file> |
head -n 11 | csvlook
```

Filter, project, export JSON

```
csvgrep -c status -m open
<file> \
  | csvcut -c id,title \
  | csvjson > open.json
```

Combine monthly files and aggregate

```
csvstack --filenames -n month
2026-*.csv > all.csv
csvsql --query "select month,
sum(total) from all group by
month" all.csv | csvlook
```