

SQLite Cheatsheet

By Dejan Panovski • Updated on Oct 4, 2026

SQLite command-line reference for opening databases, running SQL, formatting results, importing CSV files, and making backups.

This cheatsheet covers opening SQLite databases, running queries, formatting results, importing CSV files, and making backups with `sqlite3`. Run `sqlite3` commands in your terminal, and enter SQL and dot commands in the SQLite shell.

Open and Inspect

For worked examples, see our `sqlite3` command guide .

<code>sqlite3 tasks.db</code>	Open a database; create the file on the first write if it is missing
<code>sqlite3 --readonly tasks.db</code>	Inspect an existing database without allowing writes
<code>sqlite3 :memory:</code>	Open a database that disappears when the shell exits
<code>sqlite3 --version</code>	Show the installed SQLite version
<code>.databases</code>	Show open databases and their file paths
<code>.tables</code>	List tables and views
<code>.schema tasks</code>	Show the SQL that defines a table
<code>.help</code>	Show available dot commands
<code>.quit</code>	Exit the SQLite shell

Run SQL

End SQL statements with a semicolon. Dot commands do not need one. Use `WHERE` to limit which rows an update changes.

<code>CREATE TABLE tasks (id INTEGER PRIMARY KEY, title TEXT NOT NULL, done INTEGER DEFAULT 0);</code>	Create a task table with automatically assigned IDs
<code>INSERT INTO tasks (title) VALUES ('Check backups');</code>	Add a task with the default done value of 0
<code>SELECT * FROM tasks LIMIT 10;</code>	Preview up to ten rows
<code>SELECT title FROM tasks WHERE done = 0;</code>	List unfinished tasks
<code>SELECT count(*) FROM tasks;</code>	Count all rows
<code>UPDATE tasks SET done = 1 WHERE id = 1;</code>	Mark one task as completed

Output Formats

Set the mode before changing headers. Column spacing and alignment can vary by SQLite version.

<code>.mode list</code>	Separate columns with pipes
<code>.mode column</code>	Align values in columns
<code>.mode box</code>	Add table borders
<code>.mode json</code>	Print results as JSON
<code>.mode csv</code>	Format results as CSV
<code>.headers on</code>	Include column names where the mode supports them
<code>.headers off</code>	Hide column names where the mode supports it

CSV Import and Export

Imports append rows. Use `--skip 1` for a CSV header when the target table already exists. To export, set `.mode csv` and `.headers on` before `.once`. Choose an unused output filename.

<code>.import --csv contacts.csv contacts</code>	Import CSV; a new table uses the first row as column names
<code>.import --csv --skip 1 contacts.csv contacts</code>	Skip the header when importing into an existing table
<code>.once report.csv</code>	Send the next query result to a file
<code>SELECT id, title FROM tasks;</code>	Run after <code>.once</code> to export the selected columns
<code>.output report.csv</code>	Send query results to a file until output is changed
<code>.output</code>	Send query results back to the terminal

Transactions and Locks

Back up first and preview the matching rows before deleting. Use `ROLLBACK` to discard changes or `COMMIT` to keep them.

<code>BEGIN;</code>	Start a transaction
<code>SELECT * FROM tasks WHERE done = 1;</code>	Preview completed tasks before deletion
<code>DELETE FROM tasks WHERE done = 1;</code>	Delete completed tasks inside the transaction
<code>ROLLBACK;</code>	Undo the uncommitted changes
<code>COMMIT;</code>	Keep the changes; they can no longer be rolled back
<code>.timeout 5000</code>	Wait up to five seconds for a lock; does not release another connection's lock

Backup and Restore

Run these commands in your terminal. Choose unused backup filenames and restore SQL into a new database. Use `.backup` for a live database, including committed data still in its WAL file.

<code>sqlite3 -readonly tasks.db ".backup backup.db"</code>	Create a consistent database copy
<code>sqlite3 -readonly tasks.db .dump > backup.sql</code>	Export the database as SQL statements
<code>sqlite3 -bail restored.db < backup.sql</code>	Restore SQL into a new database and stop on errors
<code>sqlite3 -readonly backup.db "PRAGMA integrity_check; "</code>	Check database structure; ok means the check passed