

SQLite Cheat Sheet

sqlite3 shell commands and everyday SQL. Every line was run with SQLite 3.51.0 on 6 October 2026. Full tested guides: sqldocs.org/sqlite-cheat-sheet/

sqlite3 shell

Open or create a database file	<code>.open shop.db</code>
Turn on column headers	<code>.headers on</code>
Aligned column output	<code>.mode column</code>
List tables	<code>.tables</code>
Show a table's CREATE statement	<code>.schema products</code>
Export to CSV	<code>.mode csv</code>
Send output to a file	<code>.output products.csv</code>
Write the rows	<code>SELECT * FROM products;</code>
Back to the screen	<code>.output stdout</code>
Import a CSV into a new table	<code>.import --csv products.csv products_copy</code>
Dump the database as SQL	<code>.output dump.sql</code> then <code>.dump</code>
Run SQL from a file	<code>.read dump.sql</code>
Quit	<code>.quit</code>

Create and change tables

Create a table	<code>CREATE TABLE products (id INTEGER PRIMARY KEY, name TEXT NOT NULL UNIQUE, price REAL NOT NULL CHECK (price >= 0), added TEXT DEFAULT CURRENT_TIMESTAMP);</code>
Create only if missing	<code>CREATE TABLE IF NOT EXISTS orders (id INTEGER PRIMARY KEY, product_id INTEGER NOT NULL REFERENCES products(id), qty INTEGER NOT NULL DEFAULT 1);</code>
Create an index	<code>CREATE INDEX IF NOT EXISTS idx_orders_product ON orders(product_id);</code>
Add a column	<code>ALTER TABLE products ADD COLUMN stock INTEGER NOT NULL DEFAULT 0;</code>
Rename a column	<code>ALTER TABLE products RENAME COLUMN stock TO in_stock;</code>

Inspect

List tables with SQL	<code>SELECT name FROM sqlite_schema WHERE type = 'table' AND name NOT LIKE 'sqlite_%' ESCAPE '\';</code>
List a table's columns	<code>PRAGMA table_info(products);</code>

Insert, update, delete

Insert rows	<code>INSERT INTO products (name, price) VALUES ('Pen', 1.5), ('Notebook', 4.25), ('Stapler', 9.99);</code>
Insert and return the new row	<code>INSERT INTO orders (product_id, qty) VALUES (1, 3) RETURNING id, product_id, qty;</code>
Insert or update (upsert)	<code>INSERT INTO products (name, price) VALUES ('Pen', 1.75) ON CONFLICT(name) DO UPDATE SET price = excluded.price;</code>
Insert, ignoring duplicates	<code>INSERT OR IGNORE INTO products (name, price) VALUES ('Pen', 99);</code>
Update rows	<code>UPDATE products SET in_stock = in_stock + 10 WHERE name = 'Pen';</code>
Delete rows	<code>DELETE FROM products WHERE name = 'Stapler';</code>

Query

Filter, sort and limit	<code>SELECT name, price FROM products WHERE price < 5 ORDER BY price DESC LIMIT 10;</code>
Pattern match (case-insensitive for ASCII)	<code>SELECT name FROM products WHERE name LIKE 'n%';</code>
Group and aggregate	<code>SELECT product_id, count(*) AS orders, sum(qty) AS units FROM orders GROUP BY product_id HAVING sum(qty) > 0;</code>
Join two tables	<code>SELECT o.id, p.name, o.qty FROM orders AS o JOIN products AS p ON p.id = o.product_id;</code>
Conditional value	<code>SELECT name, iif(price > 2, 'pricey', 'cheap') AS band FROM products;</code>
Replace NULL	<code>SELECT name, coalesce(NULL, 'n/a') AS note FROM products LIMIT 1;</code>
Common table expression	<code>WITH cheap AS (SELECT * FROM products WHERE price < 5) SELECT count(*) FROM cheap;</code>
Window function	<code>SELECT name, price, rank() OVER (ORDER BY price DESC) AS price_rank FROM products;</code>

Strings

Concatenate	<code>SELECT 'SQL' 'ite' AS joined;</code>
Substring, upper, length	<code>SELECT substr('sqldocs', 1, 3) AS part, upper('sql') AS up, length('sqldocs') AS len;</code>
Trim and replace	<code>SELECT trim(' hi ') AS trimmed, replace('a-b-c', '-', '/') AS replaced;</code>

Dates and times

Current UTC date and time	<code>SELECT datetime('now') IS NOT NULL AS has_now;</code>
Add an interval	<code>SELECT date('2026-10-06', '+7 days') AS next_week;</code>
Start of month	<code>SELECT date('2026-10-06', 'start of month') AS month_start;</code>
Format a date	<code>SELECT strftime('%Y-%m', '2026-10-06') AS year_month;</code>
Days between dates	<code>SELECT julianday('2026-12-25') - julianday('2026-10-06') AS days;</code>
Unix seconds to text	<code>SELECT datetime(1791297015, 'unixepoch') AS from_unix;</code>

JSON

Read a JSON field	<code>SELECT '{"a": {"b": 2}}' --> '\$.a.b' AS value;</code>
Set a JSON field	<code>SELECT json_set('{"a": 1}', '\$.b', 2) AS updated;</code>
Expand a JSON array	<code>SELECT value FROM json_each('[10, 20, 30]');</code>

Types

Check a value's type	<code>SELECT typeof(42), typeof('42'), typeof(4.2), typeof(NULL);</code>
Convert a type	<code>SELECT CAST('42' AS INTEGER) + 1 AS n;</code>

Transactions

Start, commit	<code>BEGIN IMMEDIATE; UPDATE products SET price = price * 1.1 WHERE id = 2; COMMIT;</code>
Undo uncommitted changes	<code>BEGIN; DELETE FROM orders; ROLLBACK;</code>

PRAGMAs

Enforce foreign keys (per connection)	PRAGMA foreign_keys = ON;
Switch to WAL mode (persistent)	PRAGMA journal_mode = WAL;
Wait up to 5 s for locks	PRAGMA busy_timeout = 5000;
Check database integrity	PRAGMA integrity_check;

sqldocs.org · Official references: sqlite.org/cli.html, sqlite.org/lang.html, sqlite.org/pragmas.html

Maintenance

Analyze query plan	EXPLAIN QUERY PLAN SELECT * FROM orders WHERE product_id = 1;
Reclaim free space	VACUUM;
Copy a live database to a file	VACUUM INTO 'backup.db';