HighPerformanceSQLite.com

SQLite Reference Guide

Command-Line

Command-line (CLI) and dot commands for interacting with, managing, and exploring your SQLite databases

`sqlite3 <database>` Open/create database

COMMON FLAGS

`-cmd` Run commands on database open

`-header` Enable headers in output

`-csv` Output results in CSV format

`-batch` Suppress prompts and interactive behavior

DOT COMMANDS

`.help` List all available CLI commands

`.open <filename>` Open database file in CLI

`.databases` List open databases

`.backup <filename>` Restore database from file

`.tables` List all tables in database

`.schema <table>` Print schema of table

`.mode <mode>` Set output mode (e.g., box, column, csv, list)

`.headers on|off` Toggle column headers

`.import <file> <table>` Import CSV file

`.exit / .quit / Ctrl+D` Exit the SQLite shell

Data Types

Any data can be stored in any column, adding type affinities affects how SQLite interprets inserted data.

`INTEGER` Whole numbers

`REAL` Floating point numbers

`TEXT` String values

`BLOB` Binary data

`NULL` No value

SQLite has no specific date or time types; dates can be stored as:

`TEXT` Using ISO8601 format (e.g., 2025-01-30)

`INTEGER` Using epoch time (e.g., 1717027200)

`REAL` Using Julian day number (e.g., 2460705.5000000)

Date & Time

Date and time functions, tokens, and modifiers to easily format, calculate offsets, and manipulate temporal data.

Token	Description	Example
`%Y`	Year (4 digits)	`2025`
`%m`	Month (2 digits)	`01`
`%d`	Day of month (2 digits)	`30`
`%H`	Hour (24-hr 2 digits)	`14`
`%M`	Minute (2 digits)	`45`
`%S`	Second (2 digits)	`09`
`%f`	Fractional seconds (6 digits)	`123456`
`%w`	Day of week (0=Sunday)	`4`
`%W`	Week of year (2 digits)	`26`
`%j`	Day of year (3 digits)	`030`
`%s`	Unix timestamp (seconds)	`1738238400`
`%z`	Time zone offset	`+0000 (UTC)`
`%Z`	Time zone abbreviation	`UTC`
`+N minute`	Add N minutes	`+15 minute`
`+N hour`	Add N hours	`+3 hour`
`+N day`	Add N days	`+7 day`
`-N month`	Subtract N months	`-2 month`
`+N year`	Add N years	`+1 year`
`start of week`	Get first day of week	`start of week`
`start of month`	Get first day of month	`start of month`
`weekday N`	Get next Nth weekday	`weekday 0`
`CURRENT_DATE`	Return current date	`2025-01-30`
`CURRENT_TIME`	Return current time	`23:45:15`
`CURRENT_TIMESTAMP`	Return current date & time	`2025-01-30 23:45:15`
`DATE(column, modifier)`	Format a date with optional offset	`DATE(column, '+2 day')`
`TIME(column, modifier)`	Format a time with optional offset	`TIME(column, '+1 hour')`
`DATETIME(column, modifier)`	Format a date & time with optional offset	`DATETIME(column, '-1 day', '+1 hour')`
`STRFTIME('format', datetime)`	Format date with format string	`STRFTIME('%Y-%m-%d', 'now')`

EXAMPLE USAGE

```
SELECT DATE('2025-05-15', '+2 day'); -- Output: 2025-05-17
SELECT STRFTIME('%H', '2025-01-30 14:45:15'); -- Output: 14
SELECT DATE('2025-07-15', 'start of week'); -- Output: "2025-07-14"
```

HighPerformanceSQLite.com

SQLite Reference Guide

Common SQL Commands

Fundamental SQL commands for creating, modifying, and querying your database.

```
-- Create table
CREATE TABLE table_name (
  id INTEGER PRIMARY KEY,
  name TEXT NOT NULL,
  age INTEGER
);

-- Create table without rowid
CREATE TABLE users (
  email TEXT PRIMARY KEY,
  name TEXT
) WITHOUT ROWID;

-- Remove entire table
DROP TABLE table_name;

-- Insert values into table
INSERT INTO table_name (column1, column2)
  VALUES (value 1, value2);

-- Update values
UPDATE table_name
  SET column = value
  WHERE condition;

-- Delete select records
DELETE FROM table_name
  WHERE condition;

-- Select all records
SELECT *
  FROM table_name;

-- Select specific columns
SELECT column1, column2
  FROM table_name
  WHERE condition;

-- Select distinct records
SELECT DISTINCT column1, column2
  FROM table_name;

-- Inner join
SELECT a.col, b.col
  FROM table_a a
  INNER JOIN tableb b ON a.id = b.id;

-- Left join
SELECT a.col, b.col
  FROM table_a a
  LEFT JOIN tableb b ON a.id = b.id;

-- Select non-NULL records
SELECT *
  FROM table_name
  WHERE column is not NULL;

-- Add new column to table
ALTER TABLE table_name
  ADD COLUMN new column TEXT;

-- Rename table
ALTER TABLE old_table_name
  RENAME TO new_table_name;

-- Paginate results
SELECT *
  FROM table_name
  LIMIT 10 OFFSET 20;

-- Create index
CREATE INDEX idx_name ON table_name
 (column);
CREATE INDEX idx_name_column2 ON
 table_name (column1, column2);

-- Enforce uniqueness on one or more columns
CREATE UNIQUE INDEX idx1_name ON  table_name
(column1, column2);

-- Remove index
DROP INDEX idx_name;

-- Subquery
SELECT *
  FROM table_name
  WHERE column =
    (SELECT value FROM other_table);

-- Aggregate functions
SELECT category,COUNT(*)
  FROM table_name
  GROUP BY category;
```

Operators & Functions

SQL operators for refining query conditions and key aggregate functions for summarizing and analyzing data.

CONDITIONAL OPERATORS

`=`	Equals
`!=`, `<>`	Not equal
`<`, `>`, `<=`, `>=`	Comparisons
`LIKE '%pattern%'`	Match text containing pattern
`IN (val1, val2)`	Match any value in list
`BETWEEN val1 and val2`	Inclusive range

AGGREGATE FUNCTIONS

Aggregate functions must be used with GROUP BY

`COUNT(column)`	Count rows
`SUM(column)`	Sum up values
`AVG(column)`	Calculate average
`MAX(column)`	Find maximum
`MIN(column)`	Find minimum

Performance Tips

Key commands to enhance database performance.

Use Explain to debug queries

```
EXPLAIN QUERY PLAN SELCT * FROM table_name;
```

Improve read and write performance

```
PRAGMA journal_mode=wal
```

Reduce locking issues

```
PRAGMA busy_timeout=5000
```

Increase cache size to 10,000 pages

```
PRAGMA cache_size=10000
```

Improve temp data performance

```
PRAGMA temp_store=MEMORY
```