Table of Contents

SQL Commands
SQL Keywords
SQLite Program Dot Commands

SQLite Statements

These SQL Statements are organized by their CRUD function on the table or database - Create, Read, Update, or Delete.

CREATE

CREATE a database

`sqlite3 <database_name>.db`	This statement starts the sqlite3 program with the database file specified open. If the file doesn't exist, a new database file with the specified name is automatically created. If no database file is given, a temporary database is created and deleted when the sqlite3 program closes. **Note** this is a SQLite program statement to open the program (different from SQL commands)	`sqlite3 shelter.db`

CREATE a table

`CREATE TABLE <table_name>(`
`<column_name_1> <data_type_1>,`
`<column_name_2> <data_type_2>,`
`...);`
	Create a table with the specified name containing column names of the specified data types.	`CREATE TABLE pets (`
		` _id INTEGER,`
		` name TEXT,`
		` breed TEXT,`
		` gender INTEGER,`
		` weight INTEGER);`

INSERT data in a table

`INSERT INTO <table_name>( <column_name_1>, <column_name_2>, …) VALUES ( <values_1>, <values_2>, …);`	Insert into a specific table the listed values at the corresponding column names.	`INSERT INTO pets ( _id, name, breed, gender, weight) VALUES ( 1, "Tommy", "Pomeranian", 1, 4);`

READ

SELECT data from a table

`SELECT <columns> FROM <table_name>;`	Select specific column(s) from a table.	`SELECT name, breed from pets;`
`SELECT * FROM <table_name>;`	Select all columns and all rows from a specific table. (Asterisk here means "all columns and all rows").	`SELECT * FROM pets;`

UPDATE

UPDATE data in a table

`UPDATE <table_name> SET <column_name> = <value> WHERE <condition>;`	Update information in an existing row in a table.	`UPDATE pets SET weight = 18 WHERE _id = 5;`

DELETE

DELETE data from a table

`DELETE FROM <table_name> WHERE <condition>;`	Delete data from a table that meet the conditions of the WHERE clause.	`DELETE FROM pets WHERE _id = 1;`

	Different from DROP TABLE because the table definition still remains.	
DROP TABLE		
DROP TABLE <table_name>;	Remove a table definition and all its data.	DROP TABLE pets;

SQLite Keywords

These SQLite keywords are to be used in conjunction with SQL commands.

PRIMARY KEY		
CREATE TABLE <table_name> (<column_1> <data_type_1> **PRIMARY KEY**, <column_2> <data_type_2>, ...);	Ensure uniqueness. There can only be one primary key per table.	CREATE TABLE headphones (_id INTEGER **PRIMARY KEY**, name TEXT, price INTEGER, style INTEGER, in_stock INTEGER, description TEXT);
AUTOINCREMENT		
CREATE TABLE <table_name> (<column_1> <data_type_1> **AUTOINCREMENT**, <column_2> <data_type_2>, ...);	Automatically calculate new integer when row is added. Useful for IDs.	CREATE TABLE headphones (_id INTEGER PRIMARY KEY **AUTOINCREMENT**, name TEXT, price INTEGER, style INTEGER, in_stock INTEGER, description TEXT);
NOT NULL		
CREATE TABLE <table_name> (<column_1> <data_type_1> **NOT NULL**, <column_2> <data_type_2>, ...);	When a value is inserted into the table, it MUST have a value associated with it.	CREATE TABLE headphones (_id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT **NOT NULL**, price INTEGER, style INTEGER, in_stock INTEGER, description TEXT);
DEFAULT <value>		

`CREATE TABLE <table_name> ( <column_1> <data_type_1>` **`DEFAULT <value>,`** `<column_2> <data_type_2>, …);`	When inserting a new row, if no value is provided, the default value will be used.	`CREATE TABLE headphones ( _id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT NOT NULL, price INTEGER, style INTEGER, in_stock INTEGER NOT NULL` **`DEFAULT 0,`** `description TEXT);`
WHERE clause		
Some examples: `SELECT * FROM pets` **`WHERE`** `<condition>;` `UPDATE <table_name> SET <column_name> = <value>` **`WHERE`** `<condition>;` `DELETE FROM <table_name>` **`WHERE`** `<condition>;`	The WHERE clause ensures that only rows that meet the specified criteria are affected. It can be used in conjunction with SELECT, INSERT, UPDATE, or DELETE statements.	`SELECT * FROM pets` **`WHERE`** `_id = 1;` `SELECT * FROM pets` **`WHERE`** `weight >= 15;` `SELECT name, gender FROM pets` **`WHERE`** `breed != "Breed Unknown";` `DELETE FROM pets` **`WHERE`** `_id = <id_of_pet_to_delete>;`
ORDER BY clause		
`SELECT <column_name> FROM <table_name>` **`ORDER BY`** `<column_name> <ASC|DESC>;`	Sort the data in either ascending (ASC) or descending (DESC) order based on the column(s) listed.	`SELECT * FROM pets` **`ORDER BY`** `name ASC;` `SELECT weight FROM pets` **`ORDER BY`** `name DESC;`

SQLite Program Dot Commands

These dot commands are specific to the Sqlite Version 3 program(a database library) to be used in the command prompt/terminal. Don't confuse them with Structured Query Language (SQL) commands.
To see a full list of dot commands, check here.

`.header <on|off>`	Turn display headers on or off
`.help`	Display the help menu listing dot commands
`.mode <mode>`	Set the output mode to one of these options - ascii, csv, column, html, insert, line, list, tabs, tcl

`.open <filename>`	Close the existing database and open the file name given
`.quit`	Exit the program
`.schema <table_name>`	Show the CREATE statement used to generate the table listed
`.tables`	List names of tables