SQLite Cheatsheet

A quick reference guide to SQLite, covering essential commands, data types, and query syntax for database management and manipulation.

Basic Commands & Syntax

Connecting & Creating Databases

`sqlite3 database_name.db` - Creates a new SQLite database or opens an existing one.

`.help` - Displays available SQLite commands.

`.databases` - Lists the currently open databases.

`.exit` or `.quit` - Closes the SQLite connection.

`.show` - Show current settings.

Basic SQL Operations

`CREATE TABLE table_name (column1 datatype, column2 datatype, ...);`	Creates a new table in the database.
`INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...);`	Inserts a new row into the table.
`SELECT column1, column2, ... FROM table_name WHERE condition;`	Retrieves data from the table based on the specified condition.
`UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition;`	Modifies existing data in the table based on the specified condition.
`DELETE FROM table_name WHERE condition;`	Deletes rows from the table based on the specified condition.

Data Types & Constraints

Common Data Types

`INTEGER` : Stores integer values.

`TEXT` : Stores text strings.

`REAL` : Stores floating-point numbers.

`BLOB` : Stores binary data.

`NUMERIC` : Stores numeric values (can behave like INTEGER or REAL).

Constraints

`PRIMARY KEY`	Uniquely identifies each row in a table.
`NOT NULL`	Ensures that a column cannot have a NULL value.
`UNIQUE`	Ensures that all values in a column are distinct.
`CHECK (expression)`	Specifies a condition that must be true for any row in the table.
`DEFAULT value`	Specifies a default value for a column when no value is provided.
`FOREIGN KEY`	Establishes a link between tables.

Advanced Queries

WHERE Clause

`SELECT * FROM table_name WHERE column1 = 'value';` - Filters rows where column1 equals 'value'.

`SELECT * FROM table_name WHERE column2 > 10;` - Filters rows where column2 is greater than 10.

`SELECT * FROM table_name WHERE column3 LIKE 'pattern%';` - Filters rows where column3 starts with 'pattern'.

`SELECT * FROM table_name WHERE column4 IS NULL;` - Filters rows where column4 is NULL.

`SELECT * FROM table_name WHERE column5 BETWEEN 1 AND 10;` - Filters rows where column5 is between 1 and 10 (inclusive).

JOIN Operations

`INNER JOIN`	Returns rows only when there is a match in both tables.
`LEFT JOIN`	Returns all rows from the left table and matching rows from the right table. If no match, the right side will contain nulls.
`RIGHT JOIN`	Returns all rows from the right table and matching rows from the left table. If no match, the left side will contain nulls.
`FULL OUTER JOIN`	Returns all rows when there is a match in one of the tables.
`CROSS JOIN`	Returns all possible combinations of rows from all tables.

GROUP BY and Aggregate Functions

`GROUP BY column_name` - Groups rows that have the same values in a column into summary rows, like how many customers are in each country.

`COUNT(column_name)` - Returns the number of items in a group.

`SUM(column_name)` - Returns the sum of the values in a group.

`AVG(column_name)` - Returns the average of the values in a group.

`MIN(column_name)` - Returns the smallest value in a group.

`MAX(column_name)` - Returns the largest value in a group.

Managing Tables

Altering Tables

`ALTER TABLE table_name ADD COLUMN column_name datatype;` - Adds a new column to an existing table.

`ALTER TABLE table_name RENAME TO new_table_name;` - Renames an existing table.

Dropping Tables

`DROP TABLE table_name;` - Deletes a table and its data.

Indexes

`CREATE INDEX index_name ON table_name (column_name);`	Creates an index on a column to speed up query performance.
`DROP INDEX index_name;`	Removes an index.