

# SQLITE - SYNTAX

[http://www.tutorialspoint.com/sqlite/sqlite\\_syntax.htm](http://www.tutorialspoint.com/sqlite/sqlite_syntax.htm)

Copyright © tutorialspoint.com

SQLite is followed by unique set of rules and guidelines called Syntax. This tutorial gives you a quick start with SQLite by listing all the basic SQLite Syntax.

## Case Sensitivity

Important point to be noted is that SQLite is **case insensitive**, but there are some commands, which are case sensitive like **GLOB** and **glob** have different meaning in SQLite statements.

## Comments

SQLite comments are extra notes, which you can add in your SQLite code to increase its readability and they can appear anywhere; whitespace can occur, including inside expressions and in the middle of other SQL statements but they can not be nested.

SQL comments begin with two consecutive "--" characters *ASCII0x2d* and extend up to and including the next newline character *ASCII0x0a* or until the end of input, whichever comes first.

You can also use C-style comments, which begin with "/\*" and extend up to and including the next "\*/" character pair or until the end of input, whichever comes first. C-style comments can span multiple lines.

```
sqlite>.help -- This is a single line comment
```

## SQLite Statements

All the SQLite statements start with any of the keywords like SELECT, INSERT, UPDATE, DELETE, ALTER, DROP, etc., and all the statements end with a semicolon ;.

### SQLite ANALYZE Statement:

```
ANALYZE;  
or  
ANALYZE database_name;  
or  
ANALYZE database_name.table_name;
```

### SQLite AND/OR Clause:

```
SELECT column1, column2....columnN  
FROM table_name  
WHERE CONDITION-1 {AND|OR} CONDITION-2;
```

### SQLite ALTER TABLE Statement:

```
ALTER TABLE table_name ADD COLUMN column_def...;
```

### SQLite ALTER TABLE Statement *Rename*:

```
ALTER TABLE table_name RENAME TO new_table_name;
```

### SQLite ATTACH DATABASE Statement:

```
ATTACH DATABASE 'DatabaseName' As 'Alias-Name';
```

### SQLite BEGIN TRANSACTION Statement:

```
BEGIN;
```

```
OR  
BEGIN EXCLUSIVE TRANSACTION;
```

## SQLite BETWEEN Clause:

```
SELECT column1, column2,...columnN  
FROM   table_name  
WHERE  column_name BETWEEN val-1 AND val-2;
```

## SQLite COMMIT Statement:

```
COMMIT;
```

## SQLite CREATE INDEX Statement :

```
CREATE INDEX index_name  
ON table_name ( column_name COLLATE NOCASE );
```

## SQLite CREATE UNIQUE INDEX Statement :

```
CREATE UNIQUE INDEX index_name  
ON table_name ( column1, column2,...columnN);
```

## SQLite CREATE TABLE Statement:

```
CREATE TABLE table_name(  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
    ....  
    columnN datatype,  
    PRIMARY KEY( one or more columns )  
);
```

## SQLite CREATE TRIGGER Statement :

```
CREATE TRIGGER database_name.trigger_name  
BEFORE INSERT ON table_name FOR EACH ROW  
BEGIN  
    stmt1;  
    stmt2;  
    ....  
END;
```

## SQLite CREATE VIEW Statement :

```
CREATE VIEW database_name.view_name AS  
SELECT statement....;
```

## SQLite CREATE VIRTUAL TABLE Statement:

```
CREATE VIRTUAL TABLE database_name.table_name USING weblog( access.log );  
OR  
CREATE VIRTUAL TABLE database_name.table_name USING fts3( );
```

## SQLite COMMIT TRANSACTION Statement:

```
COMMIT;
```

## SQLite COUNT Clause:

```
SELECT COUNT(column_name)
FROM   table_name
WHERE  CONDITION;
```

## SQLite DELETE Statement:

```
DELETE FROM table_name
WHERE  {CONDITION};
```

## SQLite DETACH DATABASE Statement:

```
DETACH DATABASE 'Alias-Name';
```

## SQLite DISTINCT Clause:

```
SELECT DISTINCT column1, column2....columnN
FROM   table_name;
```

## SQLite DROP INDEX Statement :

```
DROP INDEX database_name.index_name;
```

## SQLite DROP TABLE Statement:

```
DROP TABLE database_name.table_name;
```

## SQLite DROP VIEW Statement :

```
DROP INDEX database_name.view_name;
```

## SQLite DROP TRIGGER Statement :

```
DROP INDEX database_name.trigger_name;
```

## SQLite EXISTS Clause:

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name EXISTS (SELECT * FROM   table_name );
```

## SQLite EXPLAIN Statement :

```
EXPLAIN INSERT statement...;
or
EXPLAIN QUERY PLAN SELECT statement...;
```

## SQLite GLOB Clause:

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name GLOB { PATTERN };
```

## SQLite GROUP BY Clause:

```
SELECT SUM(column_name)
FROM   table_name
WHERE  CONDITION
GROUP BY column_name;
```

## SQLite HAVING Clause:

```
SELECT SUM(column_name)
FROM   table_name
WHERE  CONDITION
GROUP BY column_name
HAVING (arithmetic function condition);
```

## SQLite INSERT INTO Statement:

```
INSERT INTO table_name( column1, column2....columnN)
VALUES ( value1, value2....valueN);
```

## SQLite IN Clause:

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name IN (val-1, val-2, ...val-N);
```

## SQLite Like Clause:

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name LIKE { PATTERN };
```

## SQLite NOT IN Clause:

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name NOT IN (val-1, val-2, ...val-N);
```

## SQLite ORDER BY Clause:

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION
ORDER BY column_name {ASC|DESC};
```

## SQLite PRAGMA Statement:

```
PRAGMA pragma_name;
```

For example:

```
PRAGMA page_size;
PRAGMA cache_size = 1024;
PRAGMA table_info(table_name);
```

## SQLite RELEASE SAVEPOINT Statement:

```
RELEASE savepoint_name;
```

## SQLite REINDEX Statement:

```
REINDEX collation_name;
REINDEX database_name.index_name;
REINDEX database_name.table_name;
```

## SQLite ROLLBACK Statement:

```
ROLLBACK;
```

```
OR  
ROLLBACK TO SAVEPOINT savepoint_name;
```

## SQLite SAVEPOINT Statement:

```
SAVEPOINT savepoint_name;
```

## SQLite SELECT Statement:

```
SELECT column1, column2....columnN  
FROM table_name;
```

## SQLite UPDATE Statement:

```
UPDATE table_name  
SET column1 = value1, column2 = value2....columnN=valueN  
[ WHERE CONDITION ];
```

## SQLite VACUUM Statement:

```
VACUUM;
```

## SQLite WHERE Clause:

```
SELECT column1, column2....columnN  
FROM table_name  
WHERE CONDITION:
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js