SQLite Commands

The standard SQLite commands to interact with relational databases are similar to SQL. They are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into groups based on their operational nature –

DDL - Data Definition Language

Sr.No.	Command & Description
1	**CREATE:** Creates a new table, a view of a table, or other object in the database.
2	**ALTER:** Modifies an existing database object, such as a table.
3	**DROP:** Deletes an entire table, a view of a table or other object in the database.

DML - Data Manipulation Language

Sr.No.	Command & Description
1	**INSERT:** Creates a record
2	**UPDATE:** Modifies records
3	**DELETE:** Deletes records

DQL - Data Query Language

Sr.No.	Command & Description
1	**SELECT:** Retrieves certain records from one or more tables

Installation of SQLite:

1. Open www.sqlite.org
2. Click on download
3. Select "precompiled Binaries for windows"
4. Click on sqlite-tools-win32-x86-3190300.zip"
5. Extract all files
6. From the extracted 3 exe files select sqlite3.exe
7. And select the path of the file and add it to the environment variables in settings.
8. Then click on ok.
9. Then SQLIte is ready to use.
10. You can view the tables which are created using SQLite in **DB Browser for SQLite.**

Storage Classes and Data types

1. INTEGER. The value is a signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value.
2. REAL. The value is a floating point value, stored as an 8-byte IEEE floating point number.
3. TEXT. The value is a text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE).

To create a DATABASE in SQLite:

Following is the basic syntax of sqlite3 command to create a database: −
Syntax : $sqlite3 DatabaseName.db
Always, database name should be unique within the RDBMS.

To create a table in SQLite:

SQLite CREATE TABLE statement is used to create a new table in any of the given database. Creating a basic table involves naming the table and defining its columns and each column's data type.
Syntax
Following is the basic syntax of CREATE TABLE statement.

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

Let us create another table

```
CREATE TABLE DEPARTMENT(
  ID INT PRIMARY KEY      NOT NULL,
  DEPT          CHAR(50) NOT NULL,
  EMP_ID        INT      NOT NULL
);
```

To DROP a table:

SQLite DROP TABLE statement is used to remove a table definition and all associated data, indexes, triggers, constraints, and permission specifications for that table.
You have to be careful while using this command because once a table is deleted then all the information available in the table would also be lost forever.

Syntax:
Following is the basic syntax of DROP TABLE statement. You can optionally specify the database name along with table name as follows −

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

INSERT values into a table:

SQLite INSERT INTO Statement is used to add new rows of data into a table in the database.

Syntax

Following are the two basic syntaxes of INSERT INTO statement.

```
INSERT INTO TABLE_NAME [(column1, column2, column3,...columnN)]
VALUES (value1, value2, value3,...valueN);
```

Example

Consider you already have created COMPANY table in your testDB.db as follows −

```
sqlite> CREATE TABLE COMPANY(
  ID INT PRIMARY KEY     NOT NULL,
  NAME          TEXT    NOT NULL,
  AGE           INT     NOT NULL,
  ADDRESS       CHAR(50),
  SALARY        REAL
);
```

Now, the following statements would create six records in COMPANY table.

```
INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Paul', 32, 'California', 20000.00 );

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, 'Allen', 25, 'Texas', 15000.00 );

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
```

VALUES (3, 'Teddy', 23, 'Norway', 20000.00);

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00);

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (5, 'David', 27, 'Texas', 85000.00);

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (6, 'Kim', 22, 'South-Hall', 45000.00);
You can create a record in COMPANY table using the second syntax as follows −
INSERT INTO COMPANY VALUES (7, 'James', 24, 'Houston', 10000.00);

SELECT Query:

SQLite SELECT statement is used to fetch the data from a SQLite database table which returns data in the form of a result table. These result tables are also called result sets.

Syntax

Following is the basic syntax of SQLite SELECT statement.
SELECT column1, column2, columnN FROM table_name;

Example : SELECT * FROM COMPANY;
SELECT ID, NAME, SALARY FROM COMPANY;

Connecting to database server using python:

All the methods which are explained below are python specific methods so methods are same for all Databases.
standard steps to communicate with database:

1. import that database specific module
   i.e.,module name varies from the Database
   for database oracle module is cx_Oracle
   Microsoft SQL server pymssql
   MYSQL DB pymysql
   SQLite sqlite3

2. establish connection between python and database
   con=sqlite3.connect(database information)

eg:con=sqlite3.connect(“Test.db")

3 .to execute our SQL query & to hold result some special object is required i.e cursor object

cursor=con.cursor()

4. execute our query.cursor having many methods they are:

1.execute()-->to execute a single query -->cursor.execute(sql query)

2.executescript()-->to execute a string of sql queries separated by semicolon→

cursor.executescript(sql queries)

3. executemany()-->to execute a parameterized query i.e.,queries are same but values are different

5. fetch the results

1.cursor.fetchone()-->to fetch only one row

2.cursor.fetchall() -->to fetch all rows

3.cursor.fetchmany(n) -->to fetch 'n' rows

6. To save all the updates made by you use commit() otherwise they will not reflect on the database.After using commit() you can't undo. Before using commit() you can undo using rollback().

7. close the connection.

cursor.close()

con.close().