Unit-2: Database backup and CSV handling:

SQLite Commands

- The SQLite project provides a command-line tool called SQLite3 to allow users to interact with SQLite databases to perform insert, update, delete, etc. operations by writing SQLite statements based on our requirements.
- The SQLite3 command-line tool provides some special commands which are called as "dot (.) commands" to define output format for tables, examine databases and for other administrative operations.
- These SQLite3 dot commands always start with "dot (.)". Here we need to remember that we should **not** use a semicolon (;) to define termination of the statement.

To see all the available commands in SQLite run following command.

```
sqlite> .help
```

The above ".help" command will list all the available commands in SQLite like as shown below.

No.	Command	Description
1	.auth ON|OFF	Show authorizer callbacks
2	.backup ?DB? FILE	Backup DB (default "main") to FILE
3	.bail on|off	Stop after hitting an error. Default OFF
4	.binary on|off	Turn binary output on or off. Default OFF
5	.changes on|off	Show number of rows changed by SQL
6	.check GLOB	Fail if output since .testcase does not match
7	.clone NEWDB	Clone data into NEWDB from the existing database
8	.databases	List names and files of attached databases
9	.dbinfo ?DB?	Show status information about the database
10	.dump ?TABLE? ...	Dump the database in an SQL text format. If TABLE specified, only dump tables matching LIKE pattern TABLE.
11	.echo on|off	Turn command echo on or off
12	.eqp on|off|full	Enable or disable automatic EXPLAIN QUERY PLAN
13	.exit	Exit this program
14	.explain ?on|off|auto?	Turn EXPLAIN output mode on or off or to automatic
15	.fullschema ?--indent?	Show schema and the content of sqlite_stat tables
16	.headers on|off	Turn display of headers on or off
17	.help	Show this message
18	.import FILE TABLE	Import data from FILE into TABLE
19	.imposter INDEX TABLE	Create an imposter table TABLE on index INDEX

20	.indexes ?TABLE?	Show names of all indexes. If TABLE specified, only show indexes for tables matching LIKE pattern TABLE.
21	.limit ?LIMIT? ?VAL?	Display or change the value of an SQLITE_LIMIT
22	.lint OPTIONS	Report potential schema issues. Options: fkey-indexes. Find missing foreign key indexes
23	.load FILE ?ENTRY?	Load an extension library
24	.log FILE|off	Turn logging on or off. FILE can be stderr/stdout
25	.mode MODE ?TABLE?	Set output mode where **MODE** is one of: ASCII - Columns/rows delimited by 0x1F and 0x1E CSV - Comma-Separated Values column - Left-aligned columns. (See .width) HTML - HTML <table> code insert - SQL insert statements for TABLE line - One value per line list - Values delimited by .separator strings quote - Escape answers as for SQL tabs - Tab-separated values tcl - TCL list elements
26	.nullvalue STRING	Use STRING in place of NULL values
27	.once FILENAME	Output for the next SQL command only to FILENAME
28	.open ?--new? ?FILE?	Close existing database and reopen FILE. The --new starts with an empty file
29	.output ?FILENAME?	Send output to FILENAME or stdout
30	.print STRING...	Print literal STRING
31	.prompt MAIN CONTINUE	Replace the standard prompts
32	.quit	Exit this program
33	.read FILENAME	Execute SQL in FILENAME
34	.restore ?DB? FILE	Restore content of DB (default "main") from FILE
35	.save FILE	Write in-memory database into FILE
36	.scanstats on|off	Turn sqlite3_stmt_scanstatus() metrics on or off
37	.schema ?PATTERN?	Show the CREATE statements matching PATTERN. Add --indent for pretty-printing
38	.separator COL ?ROW?	Change the column separator and optionally the row separator for both the output mode and .import
39	.shell CMD ARGS...	Run CMD ARGS... in a system shell
40	.show	Show the current values for various settings
41	.stats ?on|off?	Show stats or turn stats on or off

42	.system CMD ARGS...	Run CMD ARGS... in a system shell
43	.tables ?TABLE?	List the names of tables. If TABLE specified, only list tables matching LIKE pattern TABLE.
44	.testcase NAME	Begin redirecting output to 'testcase-out.txt'
45	.timeout MS	Try opening locked tables for MS milliseconds
46	.timer on|off	Turn SQL timer on or off
47	.trace FILE|off	Output each SQL statement as it is run
48	.vfsinfo ?AUX?	Information about the top-level VFS
50	.vfslist	List all available VFSes
51	.vfsname ?AUX?	Print the name of the VFS stack
52	.width NUM1 NUM2 ...	Set column widths for "column" mode. Negative values right-justify

2.1 SQLite dump :

2.1.1 Dump specific table into file, Dump only table structure

SQLite Dump Command

- In SQLite Dump command is used to dump the databases, table's schema and tables data based on our requirements.
- By using SQLite Dump command we can save the structure and data of the database tables in SQL format to the disk.

Syntax of SQLite Dump Command

Following is the syntax of SQLite Dump command to dump databases, tables structure and data based on our requirements.

```
sqlite> .dump
```

The above syntax will dump complete database.

SQLite Dump Only Table Schema(Dump only table structure)

To dump only schema of specific table, SQLite offers ".schema" command and we can redirect output of this command to external file using ".output" command.
Following is the example of dumping only schema of "Products" table.

```
sqlite> .open ProductMaster.db

sqlite> .tables
Product     Product_log

sqlite> .output ProductSchema.sql

sqlite> .schema Products

sqlite> .quit
```

When we execute above queries "ProductSchema.sql" is generated at the location where our sqlite3.exe is exist with Product table Structure.

2.1.2 Dump entire database into file

SQLite Dump Whole Database

Suppose if we want to dump whole database instead of just one table, then we need to use dump command without specifying any table name as shown following.

```
sqlite> .open Empdb.db
sqlite> .tables
emp_master
sqlite> .output Wholedb.sql
sqlite> .dump
sqlite> .quit
```

When we execute above queries "Wholedb.sql" is Created at the location where our sqlite3.exe is exist with all the tables structures and Data.

2.1.3 Dump data of one or more tables into a file

Dump data of one table into file:

If we want to dump our table to SQL file then first we need to use ".output" command. After that we need to use ".dump" command to redirect result to defined file.

```
sqlite> .open ProductMaster1.db
sqlite> .tables
Book        Product     Product_log  Publisher
sqlite> .mode column
sqlite> Select * from Product;
pid  pname     amount  quantity
---  --------     ------  --------
1    Marbles      100.0   3
2    KeyBoard     100.0   3
3    Pencil       200.0   40
4    Mouse        400.0   100
sqlite> .output ProductData.sql
sqlite> .dump Product
sqlite> .quit
```

When we execute above queries "ProductData.sql" is Created at the location where our sqlite3.exe is exist with product table Data.

Dump data of one or more tables into file:

First, set the mode to insert using the .mode command as follows:

```
sqlite>.mode insert
```

From now on, every SELECT statement will issue the result as the INSERT statements instead of pure text data.

Second, set the output to a text file instead of the default standard output. The following command sets the output file to the Productdata.sql file.

```
sqlite>.output Productdata.sql
```

Third, issue the SELECT statements to query data from a table that you want to dump. The following command returns data from the artists table.

```
sqlite> select * from Product;
```

Check the content of the Productdata.sql file, if everything is fine.

To dump data from other tables, you need to issue the SELECT statements to query data from those tables.

2.2 CSV files handling:
2.2.1 Import a CSV file into a table

1)In the first scenario, you want to import data from CSV file into a table that does not exist in the SQLite database.
First, the sqlite3 tool creates the table. The sqlite3 tool uses the first row of the CSV file as the names of the columns of the table.

Second, the sqlite3 tool import data from the second row of the CSV file into the table.

We will import a CSV file named city.csv with two columns: name and population. You can Create it with header.

To import the c:\sqlite\city.csv file into the cities table:
First, set the mode to CSV to instruct the command-line shell program to interpret the input file as a CSV file. To do this, you use the .mode command as follows:

```
sqlite> .mode csv
```

Second, use the command .import FILE TABLE to import the data from the city.csv file into the cities table.

```
sqlite>.import c:/sqlite/city.csv cities
```

To verify the import, you use the command .schema to display the structure of the cities table.

```
sqlite> .schema cities
CREATE TABLE cities(
  "name" TEXT,
  "population" TEXT
);
```

To view the data of the cities table, you use the following SELECT statement.

```
SELECT name, population FROM cities;
```

2)In the second scenario, the table is already available in the database and you just need to import the data.
First, drop the cities table that you have created.

```
DROP TABLE IF EXISTS cities;
```

Second, use the following CREATE TABLE statement to create the table cities.

```
CREATE TABLE cities(
  name TEXT NOT NULL,
  population INTEGER NOT NULL
);
```

If the table already exists, the sqlite3 tool uses all the rows, including the first row, in the CSV file as the actual data to import. Therefore, you should delete the first row of the CSV file.
The following commands import the city_without_header.csv file into the cities table.

```
sqlite> .mode csv
sqlite> .import c:/sqlite/city_no_header.csv cities
```

2.2.2 Export a CSV file from table

SQLite Export Data to CSV File

To export data from the SQLite database to a CSV file, you use these steps:

1. Turn on the header of the result set using the .header on command.
2. Set the output mode to CSV to instruct the sqlite3 tool to issue the result in the CSV mode.
3. Send the output to a CSV file.
4. Issue the query to select data from the table to which you want to export.

In SQLite, by using ".output" command we can export data from database tables to CSV or excel external files based on our requirement.

Syntax of SQLite Export Command

```
.output (filename)
```

Example:

We will export "emp_master" table data to Employee.csv file for that write the query like as shown below. Let's look at the example of exporting data of emp_master table to Employee.csv file. This file does not exist. So it will first create and export data into it.

```
sqlite> .header on

sqlite> .mode csv

sqlite> .output Employee.csv

sqlite> SELECT * FROM emp_master;

sqlite> .quit
```

Once we execute the above statements Employee.csv file will create in the folder where our SQLite3.exe file exists with emp_master table data.

When we open the Employee.csv file that will contain all the records of emp_master table.

```
emp_id,first_name,last_name,salary,dept_id
1,Honey,Patel,10100,1
2,Shweta,Jariwala,19300,2
3,Vinay,Jariwala,35100,3
4,Jagruti,Viras,9500,2
5,Shweta,Rana,12000,3
6,Sonal,Menpara,13000,1
7,Yamini,Patel,10000,2
8,Khyati,Shah,50000,3
9,Shwets,Jariwala,19400,2
```