**Experiment no. 19**

**Aim: Write a Python program to create a table and insert some records in that table. Then selects all rows from the table and display the records.**

**Theory:**

Databases offer numerous functionalities by which one can manage large amounts of information easily over the web and high-volume data input and output over a typical file such as a text file. SQL is a query language and is very popular in databases. Many websites use MySQL. SQLite is a “light” version that works over syntax very much similar to SQL. SQLite is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine. It is the most used database engine on the world wide web. Python has a library to access SQLite databases, called sqlite3, intended for working with this database which has been included with Python package since version 2.5. SQLite has the following features.

1. Serverless
2. Self-Contained
3. Zero-Configuration
4. Transactional
5. Single-Database

## **Data Types Available in SQLite for Python**

SQLite for Python offers fewer data types than other SQL implementations. This can be a bit restricting. However, as you’ll see, SQLite makes a lot of other things easier. The data types that are available:

* NULL – Includes a NULL value
* INTEGER – Includes an integer
* REAL – Includes a floating point (decimal) value
* TEXT. – Includes text
* BLOB. – Includes a binary large object that is stored exactly as input.

From this list, you may notice a number of missing data types such as dates. Unfortunately, when using SQLite, you’re restricted to these data types.

**Getting Started with SQLite in Python**

Let’s start off by loading in the library. We can do this by using the following command:

import sqlite3

**Creating a SQLite Database in Python**

In order to create a database in Python with SQLite we’ll create a Connection object that will represent the database. This object is created using SQLite’s connect() function. The connect function creates a connection to the SQLite database and returns an object to represent it.Let’s first create a .db file, as this is a very standard way of actually maintaining a SQLite database. We’ll represent the connection using a variable named conn.

conn = sqlite3.connect('Databasename.db')

With this line of code, we’ve created a new connection object, as well as a new file called Databasename.db in the directory in which you’re working.If the file already exists, the connect function will simply connect to that file.

## **Creating a Cursor Object**

The next task is to create a cursor object. Simply put, a cursor object allows us to execute SQL queries against a database. We’ll create a variable cur to hold our cursor object:

cur = conn.cursor()

## **Creating our Tables in SQLite for Python**

In SQLite database we use the following syntax to create a table:

CREATE TABLE database\_name.table\_name( column1 datatype PRIMARY KEY(one or more columns),column2 datatype, column3 datatype,columnN datatype*);*

* Import the required module
* Establish the connection or create a connection object with the database using the connect() function of the sqlite3 module.
* Create a Cursor object by calling the cursor() method of the Connection object.
* Form table using the CREATE TABLE statement with the execute() method of the Cursor class.

## **Adding Data with SQLite in Python**

The SQL INSERT INTO statement of SQL is used to insert a new row in a table. There are two ways of using the INSERT INTO statement for inserting rows:

Only values: The first method is to specify only the value of data to be inserted without the column names.

INSERT INTO table\_name VALUES (value1, value2, value3,…);

table\_name: name of the table.

value1, value2,.. : value of first column, second column,… for the new record

Column names and values both: In the second method we will specify both the columns which we want to fill and their corresponding values as shown below:

INSERT INTO table\_name (column1, column2, column3,..) VALUES ( value1, value2, value3,..);

table\_name: name of the table.

column1: name of first column, second column …

value1, value2, value3 : value of first column, second column,… for the new record

## **Selecting Data in SQLite with Python**

 Select statement is used to retrieve data from an SQLite table and this returns the data contained in the table. In SQLite the syntax of Select Statement is:

SELECT \* FROM table\_name;

fetchone()- to pull only one result

fetchmany()- to return more than only one result, we could use the fetchmany() function

fetchall()-  to return all the results

## **Deleting Data in SQLite with Python**

to delete data from a table:

DELETE FROM table\_name [WHERE Clause]

program to delete the row in a table:

* Import the required module.
* Establish the connection or create a connection object with the database using to connect() function of the sqlite3 module.
* Create a Cursor object by calling the cursor() method of the Connection object.
* Finally, trigger to execute() method on the cursor object, bypassing a DELETE statement as a parameter to it.

The UPDATE statement in SQL is used to update the data of an existing table in the database. We can update single columns as well as multiple columns using UPDATE statement as per our requirement.

#### Syntax:

UPDATE table\_name SET column1 = value1, column2 = value2,…

WHERE condition;