Practical No 11

Aim: To install and configure a database connector in Python and establish a connection to a relational database system (e.g. SQLite).

Theory:

Database connectivity allows Python programs to interact with databases — to store, retrieve, update, and manage data efficiently.
Through database connectors (like SQLite or MySQL Connector), Python can execute SQL commands directly within a program.

Python provides a built-in module called sqlite3 that enables smooth interaction with SQLite, a lightweight, serverless, and zero-configuration relational database management system (RDBMS).

Database connectivity refers to the process of connecting an application (Python program) to a database so that the program can:

- Create and manage tables
- Insert, modify, and delete data
- Retrieve and display records

This connection acts as a **bridge** between the Python application and the database engine.

Installing and Configuring the Database Connector

- SQLite comes pre-installed with Python.
- To use it, simply import the module:

```
import sqlite3
```

Establish connection

```
conn= sqlite3.connect("college.db")
```

- If the database file doesn't exist, SQLite automatically creates it.

Creating a Cursor Object:

- A cursor object is used to execute SQL commands and fetch results. You create one from the connection object:

```
cursor = conn.cursor()
```

Executing SQL Commands:

- Execute SQL commands using the cursor.execute() method. This can be for creating tables, inserting data, updating, deleting, or querying.

```
# Create a table

cursor.execute('''
```

```
    CREATE TABLE IF NOT EXISTS users (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        name TEXT NOT NULL,
        age INTEGER
    )
'''')
```

 # Insert data

```
cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("Alice", 30))
```

Retrieving Data:

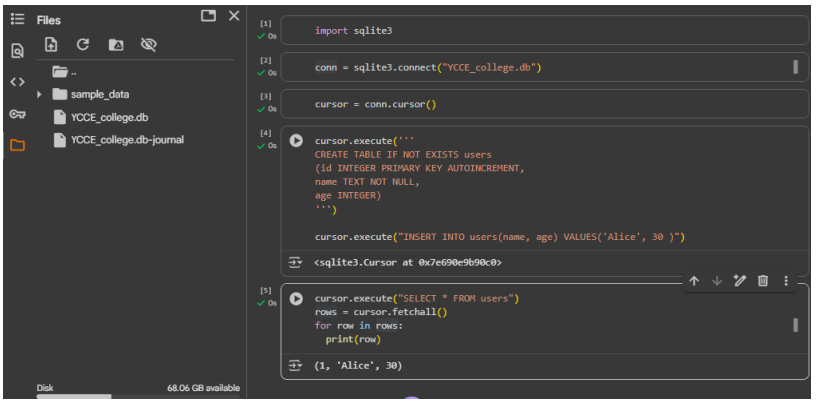
```
cursor.execute("SELECT * FROM users")
rows = cursor.fetchall()
for row in rows:
    print(row)
```

Closing the Connection:

- It's crucial to close the database connection when you're finished to release resources:

```
conn.close()
```

Output:


Files
..
sample_data
YCCE_college.db
YCCE_college.db-journal
Disk
68.06 GB available
[1]
import sqlite3
[2]
conn = sqlite3.connect("YCCE_college.db")
[3]
cursor = conn.cursor()
[4]
cursor.execute('''
CREATE TABLE IF NOT EXISTS users
(id INTEGER PRIMARY KEY AUTOINCREMENT,
name TEXT NOT NULL,
age INTEGER)
''')
cursor.execute("INSERT INTO users(name, age) VALUES('Alice', 30)")
<sqlite3.Cursor at 0x7e690e9b90c0>
[5]
cursor.execute("SELECT * FROM users")
rows = cursor.fetchall()
for row in rows:
print(row)
(1, 'Alice', 30)