Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2024-25

Class:	**SE**	**Semester:**	**IV**
Course Code:	**CSL405**	**Course Name:**	**Skills Based Python Programming Lab**

Name of Student:	**Shravani Sandeep Raut**
Roll No. :	**48**
Experiment No.:	**6**
Title of the Experiment:	**To demonstrate CRUD (create, read, update, delete) operation on a database using python.**
Date of Performance:	**18/02/2025**
Date of Submission:	**04/03/2025**

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty : Mr. Raunak Joshi

Signature :

Date:

Aim: To demonstrate CRUD (create, read, update, delete) operation on a database using python.

Theory:

CRUD is an acronym that stands for **Create**, **Read**, **Update**, and **Delete**—the four fundamental operations of persistent storage in databases. In Python, CRUD operations can be performed using various database management systems (DBMS), with **SQLite** being one of the most popular choices due to its simplicity and built-in support through the sqlite3 module.

Introduction to CRUD Operations:

- **Create:** Involves inserting new data into the database.
- **Read:** Fetches or retrieves data from the database.
- **Update:** Modifies existing data.
- **Delete:** Removes data from the database.

These operations allow for full manipulation of database records, ensuring data integrity and accessibility.

Database Choice: SQLite:

SQLite is a lightweight, serverless, self-contained SQL database engine. It is a popular choice for:

- Local application data storage.
- Rapid prototyping and testing.
- Applications needing a lightweight, zero-configuration database.

Python provides built-in support for SQLite through the sqlite3 module.

Advantages of SQLite:

- No separate server process (everything is in a single file).
- Fast and efficient for small to medium-sized applications.
- Zero configuration needed, making it suitable for beginners.

Connecting to SQLite Database:

To perform CRUD operations, a connection to the database is required. This can be done using the connect() method from the sqlite3 module.

connect() establishes a connection to the database.

cursor() allows execution of SQL commands.

Advantages of Using sqlite3 in Python:

- No separate server required.
- Built-in support in Python, no external installation needed.
- Ideal for small to medium-sized applications.

Implementation:

Importing

In [18]:

```
import sqlite3
```

Connecting to Database

In [19]:

```
connection = sqlite3.connect ('./genericDatabase.db')
cursor = connection.cursor ()
```

Create Table

```
cursor.execute('''
    CREATE TABLE IF NOT EXISTS students (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        name TEXT NOT NULL,
        age INTEGER NOT NULL
    )
''')

connection.commit()
```

CRUD Operations

Create (Insertion)

In [21]:

```
def create_student(name, age):
    cursor.execute('''
        INSERT INTO students (name, age)
        VALUES (?, ?)
    ''', (name, age))
    connection.commit()
    print("Record added successfully!")

create_student("Griffith", 20)
create_student("Guts", 22)
```

```
Record added successfully!
Record added successfully!
```

Read (Retrieve)

```
def read_students():
    cursor.execute('SELECT * FROM students')
    rows = cursor.fetchall()
    print("Student Records:")
```

```
    for row in rows:
        print(row)

read_students()
```

```
Student Records:
(1, 'Griffith', 21)
(2, 'Guts', 22)
(3, 'Griffith', 20)
(4, 'Guts', 22)
(5, 'Griffith', 20)
(6, 'Guts', 22)
```

Update

```
def update_student_age(student_id, new_age):
    cursor.execute('''
        UPDATE students
        SET age = ?
        WHERE id = ?
    ''', (new_age, student_id))
    connection.commit()
    print("Student age updated successfully!")

update_student_age(1, 21)  # Updating Alice's age to 21
read_students()
```

```
Student age updated successfully!
Student Records:
(1, 'Griffith', 21)
(2, 'Guts', 22)
(3, 'Griffith', 20)
(4, 'Guts', 22)
(5, 'Griffith', 20)
(6, 'Guts', 22)
```

Delete

In [26]:

```
def delete_student(student_id):
    cursor.execute('''
        DELETE FROM students
        WHERE id = ?
    ''', (student_id,))
    connection.commit()
    print("Student deleted successfully!")

delete_student(2)  # Deleting Bob's record
read_students()
```

```
Student deleted successfully!
Student Records:
(1, 'Griffith', 21)
(3, 'Griffith', 20)
(4, 'Guts', 22)
(5, 'Griffith', 20)
(6, 'Guts', 22)
```

Closing the database connection

In [ ]:

```
connection.close ()
```

Conclusion

CRUD operations form the backbone of any database-driven application. By using Python's built-in sqlite3 module, developers can efficiently implement these operations with minimal setup. This example demonstrates the simplicity and power of SQLite for managing application data, making it an excellent choice for beginners and small projects.