

Practical 9

9 Database Connection

Aim: Establish a connection to an SQLite database.

- Create a cursor object for executing SQL commands.
- Execute SQL commands like CREATE TABLE, INSERT, SELECT, UPDATE, and DELETE.
- Use cursor.fetchone(), cursor.fetchall(), and cursor.fetchmany() to retrieve query results.
- Use connection.commit() to save changes made to the database.
- Use connection.close() to close the database connection.
- The methods use try and except blocks to handle database-related exceptions.

Code:

```
import sqlite3

def main():
    try:
        conn = sqlite3.connect('tasks.db')
        cursor = conn.cursor()
        print("Connected to SQLite database!")

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

        cursor.execute("INSERT INTO students (name, age) VALUES
('Debdoot', 20)")
        cursor.execute("INSERT INTO students (name, age) VALUES
('Ananya', 20)")
        conn.commit()
        print("Inserted 2 records")

        cursor.execute("SELECT * FROM students")
        all_rows = cursor.fetchall()
        print("\nAll records:")
        for row in all_rows:
            print(row)

        cursor.execute("SELECT * FROM students")
        print("\nFirst record:", cursor.fetchone())

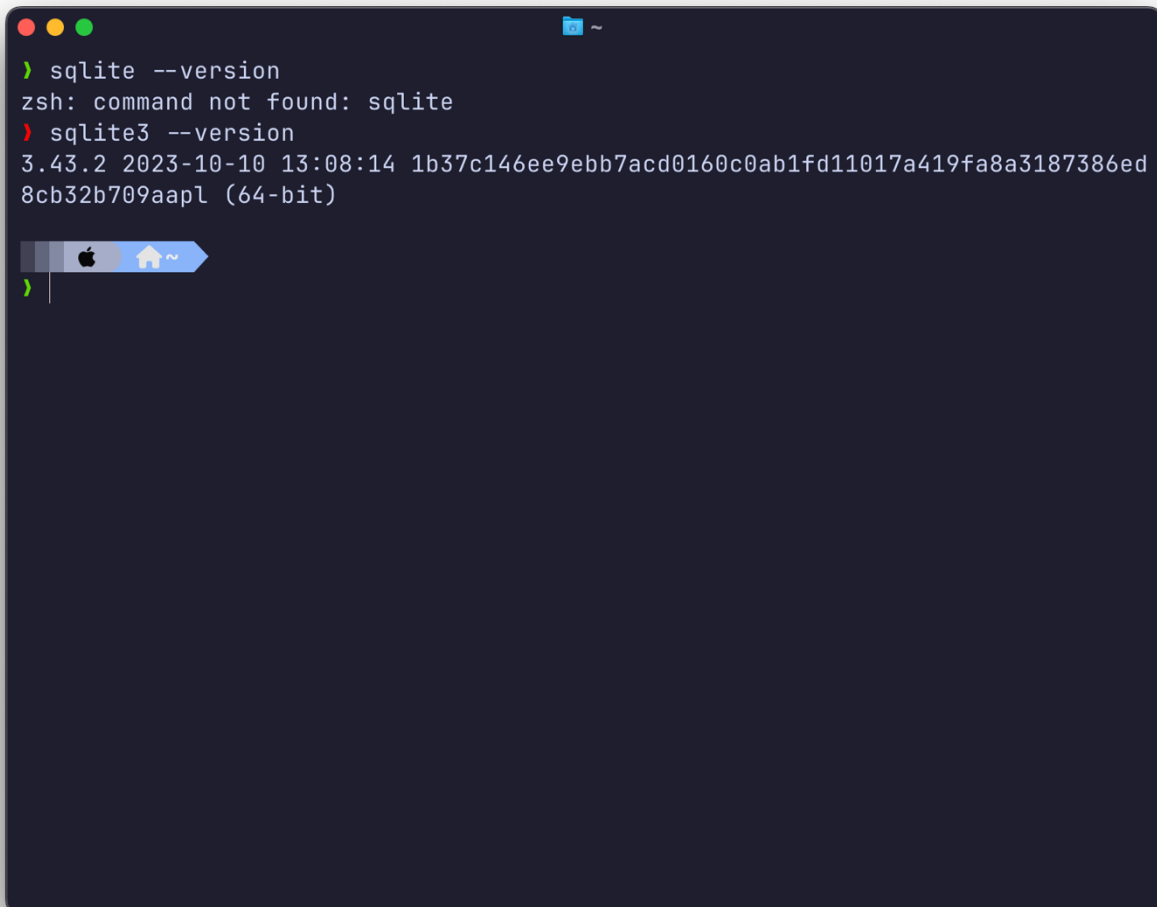
        cursor.execute("UPDATE students SET age = 21 WHERE name =
'Debdoot'")
        conn.commit()
        print("\nUpdated Debdoot's age")
```

```
        cursor.execute("DELETE FROM students WHERE name = 'Ananya'")
        conn.commit()
        print("Deleted Ananya's record")

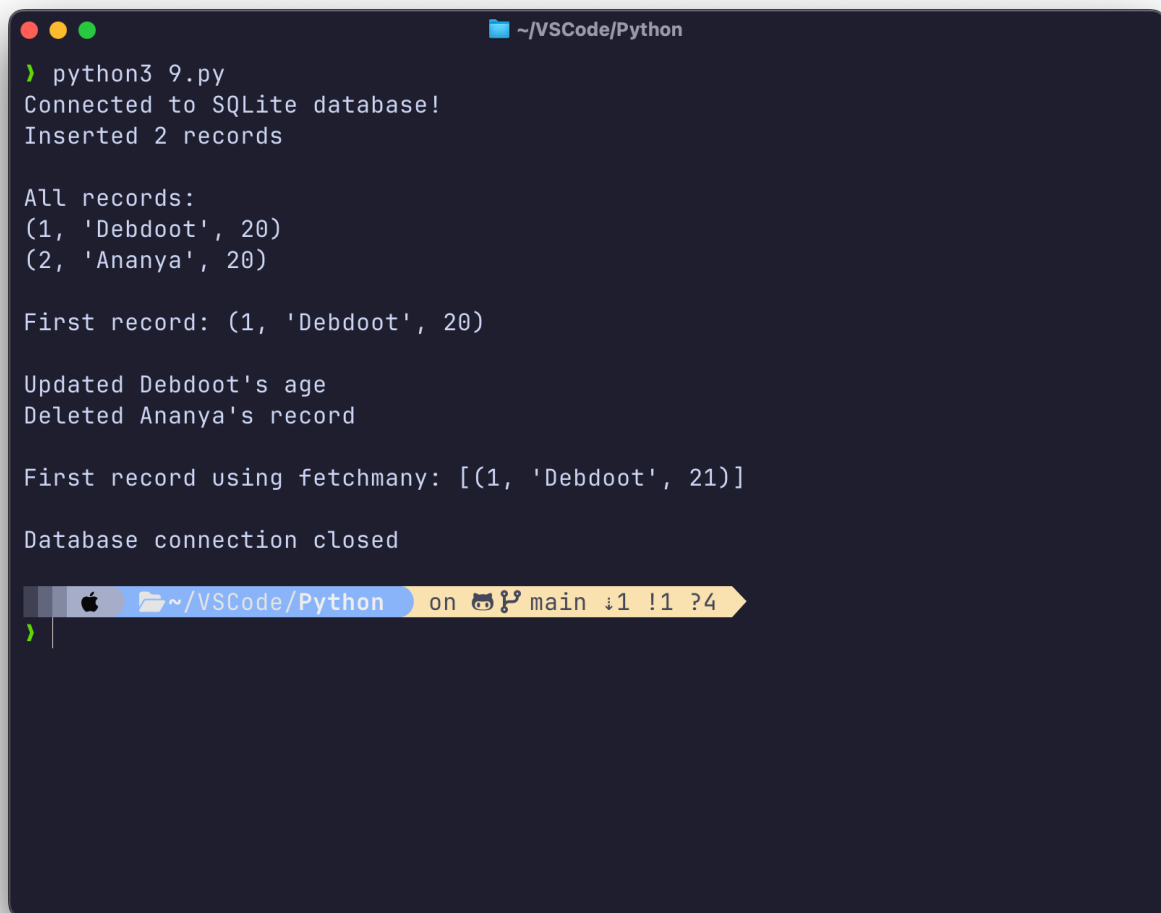
        cursor.execute("SELECT * FROM students")
        some_rows = cursor.fetchmany(1)
        print("\nFirst record using fetchmany:", some_rows)

    except sqlite3.Error as error:
        print("SQLite error:", error)
    finally:
        if conn:
            conn.close()
            print("\nDatabase connection closed")

if __name__ == "__main__":
    main()
```

Output Screenshot:A terminal window with a dark background and light blue text. The window title bar shows standard macOS window controls (red, yellow, green buttons) and a folder icon with a tilde (~). The terminal content shows two commands being executed: 'sqlite --version' which results in an error 'zsh: command not found: sqlite', and 'sqlite3 --version' which outputs the version '3.43.2' along with a timestamp and a long alphanumeric string, followed by '(64-bit)'. At the bottom, there is a prompt character ']' followed by a vertical bar cursor. The window's title bar also includes a status bar with an Apple logo, a home icon, and a tilde (~).

```
] sqlite --version
zsh: command not found: sqlite
] sqlite3 --version
3.43.2 2023-10-10 13:08:14 1b37c146ee9ebb7acd0160c0ab1fd11017a419fa8a3187386ed
8cb32b709aapl (64-bit)
] |
```



```
python3 9.py
Connected to SQLite database!
Inserted 2 records

All records:
(1, 'Debdoot', 20)
(2, 'Ananya', 20)

First record: (1, 'Debdoot', 20)

Updated Debdoot's age
Deleted Ananya's record

First record using fetchmany: [(1, 'Debdoot', 21)]

Database connection closed
```

The screenshot shows a terminal window with a dark background. The title bar indicates the path ~/VSCode/Python. The output of the script shows successful database operations: connecting, inserting two records, querying all records, retrieving the first record, updating a record, deleting a record, and finally querying the first record again using fetchmany. The database connection is closed at the end.

Conclusion/Summary:

Through this project, I successfully demonstrated the fundamentals of interacting with an SQLite database using Python. The objective was to establish a robust connection, perform CRUD operations (Create, Read, Update, Delete), and handle exceptions systematically. Here's a summary of my learnings:

1> Installation & Setup:

Installed SQLite3 on macOS using Homebrew, ensuring a smooth workflow.

Created a tasks.db database and executed SQL commands via a Python script.

2> Database Operations:

Used `cursor.execute()` to create tables, insert sample data (like Alice and Bob), and modify records.

Practiced retrieving results with `fetchone()`, `fetchall()`, and `fetchmany()` to understand data querying nuances.

3> Error Handling:

Wrapped operations in try-except blocks to catch `sqlite3.Error` exceptions, ensuring reliability.

Used commit() to save changes and close() to terminate connections safely.

4> Verification:

Validated results through terminal outputs and direct database inspection using the sqlite3 CLI.

This project strengthened my understanding of database management, SQL syntax, and Python's sqlite3 module. These skills are vital for building data-driven applications, from simple tools to complex systems. I now feel confident integrating databases into future projects while adhering to best practices.

Student Signature & Date	Marks:	Evaluator Signature & Date
-------------------------------------	---------------	---------------------------------------