**Aim:** Understand how to create a SQLite database and perform basic **CRUD** (Create, Read, Update, Delete) operations using Python.

**IDE:**

SQLite3 can be integrated with Python using sqlite3 module. It provides an SQL interface compliant with the DB-API 2.0 specification described by PEP 249. You do not need to install this module separately because it is shipped by default along with Python version 2.5.x onwards. To use sqlite3 module, you must first create a connection object that represents the database and then optionally you can create a cursor object, which will help you in executing all the SQL statements.

Let's enhance the examples with a more practical use case, focusing on **Student Record** **Management**. We will simulate managing student\_record by storing student data like their enrollment, **name**, subject, and mark in the database, and include additional operations like calculating the average mark.

**Install sqlite-database**

pip install sqlite-database

**Database Setup**

We'll set up an SQLite database to manage student record information.

**Example**

import sqlite3

# Connect to database (or create it)

conn = sqlite3.connect('student\_record.db')

# Create a cursor object using the cursor() method

cursor = conn.cursor()

**Create an Student Table**

We'll create a student\_record table to store student details such as Enrollment, name, subject, and Mark.

**Example**

# Create students table if it doesn't exist

cursor.execute('''CREATE TABLE IF NOT EXISTS student\_record (

Enrollment INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

Subject TEXT NOT NULL,

Mark INTEGER NOT NULL

)''')

# Commit the changes

conn.commit()

**Insert Student Data**

Let's insert multiple students into the table.

**Example**

# Insert multiple employee records

student\_record = [

(92301733016,'ASHUTOSH KUMAR SINGH','PWP',95),

(92301733017,'HARSH VISHALBHAI TRIVEDI','PWP',85),

(92301733027,'VIRAJ PRAKASHBHAI VAGHASIYA','PWP',90),

(92301733046,'SHIVAM ATULKUMAR BHATT', 'PWP',93),

(92301733058,'DEVENDRASINH DOLATSINH JADEJA','PWP',75)

]

# Using executemany to insert multiple records

cursor.executemany('''INSERT INTO student\_record (Enrollment, name, subject,Mark)

VALUES (?, ?, ?,?)''', student\_record)

# Commit the changes

conn.commit()

Fetch Student Data

Let’s retrieve and display all student records.

**Example**

# Fetch all student records

cursor.execute('SELECT \* FROM student\_record')

rows = cursor.fetchall()

# Display the results

print("All Student Records:")

for row in rows:

print(row)

**Fetch Data with Specific Criteria**

Let’s fetch employees who earn more than 90.

**Example**

# Fetch student got more than 90

cursor.execute('SELECT name, subject, Mark FROM student\_record WHERE Mark > 90')

high\_marks = cursor.fetchall()

print("\nStudents with Marks greater than 90:")

for student in high\_marks:

print(student)

**Update Student Information**

Suppose a student gets a raise in mark. We can update their mark using an UPDATE statement.

**Example:**

# Update MArk for Ashutosh kumar (PWP)

cursor.execute('''UPDATE student\_record SET Mark = 98

WHERE name = 'ASHUTOSH KUMAR SINGH' AND subject = 'PWP' ''')

# Commit the changes

conn.commit()

# Verify the update

cursor.execute('SELECT name, MArk FROM student\_record WHERE name = "ASHUTOSH KUMAR SINGH"')

updated\_mark = cursor.fetchone()

print(f"\nUpdated Mark for {updated\_mark[0]}: {updated\_mark[1]}")

**Delete a Student**

Let’s remove a student from the database.

**Example:**

# Delete a student record (e.g.,DEVENDRASINH DOLATSINH JADEJA )

cursor.execute('''DELETE FROM student\_record WHERE name = 'DEVENDRASINH DOLATSINH JADEJA' ''')

# Commit the changes

conn.commit()

# Verify the deletion

cursor.execute('SELECT \* FROM student\_record WHERE name = "DEVENDRASINH DOLATSINH JADEJA"')

deleted\_name = cursor.fetchone()

if deleted\_name is None:

print("\nDEVENDRASINH DOLATSINH JADEJA has been successfully deleted.")

**Calculate Average Mark**

Let’s calculate the average mark of all students.

**Example:**

# Calculate the average Mark

cursor.execute('''SELECT AVG(Mark) FROM student\_record''')

avg\_mark = cursor.fetchone()[0]

print(f"\nThe average mark of students is: ${avg\_mark:.2f}")

**Close the Database Connection**

Always close the connection after completing your operations.

**Example**

# Close the connection

conn.close()

A screenshot of a computer screen

AI-generated content may be incorrect.A screen shot of a computer

AI-generated content may be incorrect.

**Post Lab Exercise:**

Modify the system to allow a student to enroll in multiple subjects at once

import sqlite3

# Connect to database (or create it)

conn = sqlite3.connect('multiple\_student\_subjects.db')

# Create a cursor object using the cursor() method

cursor = conn.cursor()

# Drop old table for clean start

cursor.execute('DROP TABLE IF EXISTS multiple\_student\_subjects')

# Create table with composite primary key

cursor.execute('''

    CREATE TABLE multiple\_student\_subjects (

        Enrollment INTEGER,

        name TEXT NOT NULL,

        Subject TEXT NOT NULL,

        Mark INTEGER NOT NULL,

        PRIMARY KEY (Enrollment, Subject)

    )

''')

conn.commit()

# Student records

multiple\_student\_subjects = [

    (92400133189, 'om', 'PWP', 99),

    (92400133189, 'om', 'ICE', 99),

    (92400133189, 'om', 'DMGT', 99),

    (92400133189, 'om', 'DSC', 99),

    (92400133189, 'om', 'SS', 99),

    (92400133189, 'om', 'SPDT', 99),

    (92400133189, 'om', 'APTI', 99),

    (92400133189, 'om', 'COA', 99)

]

cursor.executemany('''

    INSERT INTO multiple\_student\_subjects (Enrollment, name, Subject, Mark)

    VALUES (?, ?, ?, ?)

''', multiple\_student\_subjects)

conn.commit()

# Fetch all records

cursor.execute('SELECT \* FROM multiple\_student\_subjects')

rows = cursor.fetchall()

print("All Student Subjects Records:")

for row in rows:

    print(row)

# Subjects with Marks > 90

cursor.execute('SELECT name, Subject, Mark FROM multiple\_student\_subjects WHERE Mark > 90')

high\_marks = cursor.fetchall()

print("\nSubjects with Marks greater than 90:")

for subject in high\_marks:

    print(subject)

# Update Mark for COA

cursor.execute('''

    UPDATE multiple\_student\_subjects

    SET Mark = 98

    WHERE Enrollment = 92400133189 AND Subject = 'ICE'

''')

conn.commit()

# Verify the update

cursor.execute('''

    SELECT Subject, Mark FROM multiple\_student\_subjects

    WHERE Enrollment = 92400133189 AND Subject = 'ICE'

''')

updated = cursor.fetchone()

print(*f*"\nUpdated Mark for ICE: {updated[1]}")

# Delete marks for 'SS' subject

cursor.execute('''

    DELETE FROM multiple\_student\_subjects

    WHERE Enrollment = 92400133189 AND Subject = 'DMGT'

''')

conn.commit()

# Verify deletion

cursor.execute('''

    SELECT \* FROM multiple\_student\_subjects

    WHERE Enrollment = 92400133189 AND Subject = 'DMGT'

''')

deleted = cursor.fetchone()

if deleted is None:

    print("\n'DMGT' subject record has been successfully deleted")

# Calculate the average Mark

cursor.execute('''SELECT AVG(Mark) FROM multiple\_student\_subjects''')

avg\_mark = cursor.fetchone()[0]

print(*f*"\nThe average mark of students is: {avg\_mark*:.2f*}")

OUPUT :

A screenshot of a computer program

AI-generated content may be incorrect.