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:** | | |
| **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:  1️> 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.  2️> 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.  3️> 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.  4️> 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** |