Mini Project Report:

Student Records Management System

1. Introduction

The Student Records Management System is a mini project developed using **Python** and **SQLite**.
It allows users to perform **CRUD operations** (Create, Read, Update, Delete) to manage student records.
This project demonstrates **database handling in Python** and is suitable for beginners to understand database concepts.

---

2. Objectives

- Demonstrate integration of Python with SQLite.
- Implement CRUD operations for data management.
- Provide a simple and user-friendly interface.
- Showcase database handling and error management.

---

3. Features

- Add new student records with details like **ID, Name, Age, and Grade**.
- View all existing student records.
- Update details of any student using their ID.
- Delete student records permanently.
- Automatically creates database if not present.

---

4. Technology Stack

- **Programming Language:** Python 3
- **Database:** SQLite
- **Libraries:** sqlite3 (built-in)
- **Interface:** Command-line menu

---

5. Source Code

```
import sqlite3

# Connect to SQLite database
conn = sqlite3.connect("students.db")
cursor = conn.cursor()

# Create table if not exists
cursor.execute("""
CREATE TABLE IF NOT EXISTS students (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    age INTEGER NOT NULL,
    grade TEXT NOT NULL
)
""")
conn.commit()

def add_student():
    name = input("Enter student name: ")
    age = int(input("Enter student age: "))
    grade = input("Enter student grade: ")
    cursor.execute("INSERT INTO students (name, age, grade) VALUES (?, ?, ?)", (name, age, grade))
    conn.commit()
    print("Student added successfully!")
```

```
def view_students():
    cursor.execute("SELECT * FROM students")
    records = cursor.fetchall()
    if records:
        for row in records:
            print(f"ID: {row[0]}, Name: {row[1]}, Age: {row[2]}, Grade: {row[3]}")
    else:
        print("No records found.")

def update_student():
    student_id = int(input("Enter student ID to update: "))
    name = input("Enter new name: ")
    age = int(input("Enter new age: "))
    grade = input("Enter new grade: ")
    cursor.execute("UPDATE students SET name=?, age=?, grade=? WHERE id=?", (name,
age, grade, student_id))
    conn.commit()
    print("Student updated successfully!")

def delete_student():
    student_id = int(input("Enter student ID to delete: "))
    cursor.execute("DELETE FROM students WHERE id=?", (student_id,))
    conn.commit()
    print("Student deleted successfully!")

def main():
    while True:
        print("===== Student Records Management System =====")
```

```
        print("1. Add Student")
        print("2. View Students")
        print("3. Update Student")
        print("4. Delete Student")
        print("5. Exit")
        choice = input("Enter your choice (1-5): ")
        if choice == "1":
            add_student()
        elif choice == "2":
            view_students()
        elif choice == "3":
            update_student()
        elif choice == "4":
            delete_student()
        elif choice == "5":
            print("Exiting...")
            break
        else:
            print("Invalid choice!")

if __name__ == "__main__":
    main()
    conn.close()
```

---

6. Conclusion

The Student Records Management System demonstrates **basic CRUD operations** using Python and SQLite.
It serves as a foundation for building **more advanced database applications** and helps

beginners understand **database handling, Python programming, and command-line interfaces**.