Problem Statement:

An educational institution requires a streamlined management system to handle student records, including personal information, academic details, and class assignments. The current manual system is prone to errors, inefficient, and lacks the capability to provide real-time access to student data. To address these issues, a software solution is needed that automates the management of student records, facilitates the updating and retrieval of student information, and enables easy access to search and view student details.

Abstract:

The Student Management System is a comprehensive software solution designed to streamline the administrative tasks involved in managing student records for educational institutions. Utilizing Python's Tkinter for the graphical user interface and MySQL for backend database management, the system provides functionalities for adding, viewing, updating, and deleting student records. It automates the management of student information, ensuring data integrity and ease of access. The system is designed to improve efficiency, reduce manual errors, and enhance data security by using parameterized queries to protect against SQL injection. By providing real-time access to student data and enabling efficient data handling, the Student Management System aims to facilitate better administrative decision-making and improve overall management of student records.

Design Table:

List of tables, attributes and their domains:

1)Table-1: Student Table

- roll (Primary Key)
- name
- age
- class

Detailed Schema:

Student Table

roll: Integer (Primary Key)

Domain: Positive integers

Description: Unique identifier for each student.

name: Varchar(100)

Domain: Alphabetic characters

Description: Full name of the student.

age: Integer

Domain: Positive integer

Description: Age of the student.

class: Varchar(50)

Domain: Alphanumeric character

Description: Class or grade of the student

DDL COMMANDS:

```
mysql> create database vasavi;
Query OK, 1 row affected (0.08 sec)

mysql> use vasavi;
Database changed
mysql> create table stud(roll integer(5), name varchar(20), age integer(2),class varchar(3));
Query OK, 0 rows affected, 2 warnings (0.21 sec)

mysql> desc stud;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| roll  | int         | YES  |     | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
| age   | int         | YES  |     | NULL    |       |
| class | varchar(3)  | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
4 rows in set (0.18 sec)

mysql>
```

DML COMMANDS:

```
mysql> select * from stud;
+------+------------------+------+-------+
| roll | name             | age  | class |
+------+------------------+------+-------+
|   87 | Maharshi Revanth |   10 | 5     |
+------+------------------+------+-------+
1 row in set (0.00 sec)

mysql> insert into stud values(56,"Rahul",13,8);
Query OK, 1 row affected (0.06 sec)

mysql> select * from stud;
+------+------------------+------+-------+
| roll | name             | age  | class |
+------+------------------+------+-------+
|   87 | Maharshi Revanth |   10 | 5     |
|   56 | Rahul            |   13 | 8     |
+------+------------------+------+-------+
2 rows in set (0.06 sec)
```

IMPLEMENTATION:

The provided code implements a Student Management System using Python's Tkinter library for the graphical user interface and MySQL for database management. It allows users to perform CRUD (Create, Read, Update, Delete) operations on student records. The GUI provides input fields for entering student details and buttons for adding, viewing, updating, and deleting records. The application connects to a MySQL database to store and retrieve student data. Error handling mechanisms are implemented to provide feedback to users in case of issues such as database connection errors or invalid inputs. Overall, the implementation provides a user-friendly interface for managing student records efficiently.

APPLICATION CODE:

```
import tkinter as tk

from tkinter import messagebox

import mysql.connector as x

# Database connection

con = x.connect(host='localhost', user='root', password='Maha@2004', database='vasavi')

cur = con.cursor()

# Functions

def stuInsert():

    roll = roll_entry.get()

    name = name_entry.get()

    age = age_entry.get()

    clas = class_entry.get()
```

```
    if roll and name and age and clas:
        try:
            sql = "INSERT INTO stud (roll, name, age, class) VALUES (%s, %s, %s, %s)"
            cur.execute(sql, (roll, name, age, clas))
            con.commit()
            messagebox.showinfo("Success", "Record inserted successfully")
            clear_entries()
        except Exception as e:
            messagebox.showerror("Error", str(e))
    else:
        messagebox.showwarning("Input Error", "Please fill all fields")

def stuview():
    try:
        cur.execute("SELECT * FROM stud")
        results = cur.fetchall()
        output_text.delete(1.0, tk.END)
        for row in results:
            output_text.insert(tk.END, f"{row}\n")
    except Exception as e:
        messagebox.showerror("Error", str(e))

def searchstu():
    condition = search_entry.get()
```

```
    try:
        cur.execute(f"SELECT * FROM stud WHERE {condition}")
        results = cur.fetchall()
        output_text.delete(1.0, tk.END)
        for row in results:
            output_text.insert(tk.END, f"{row}\n")
    except Exception as e:
        messagebox.showerror("Error", str(e))


def removestu():
    roll = roll_entry.get()
    try:
        cur.execute("DELETE FROM stud WHERE roll=%s", (roll,))
        con.commit()
        messagebox.showinfo("Success", "Record deleted successfully")
        clear_entries()
    except Exception as e:
        messagebox.showerror("Error", str(e))


def updatestu():
    roll = roll_entry.get()
    name = name_entry.get()
    age = age_entry.get()
    clas = class_entry.get()
```

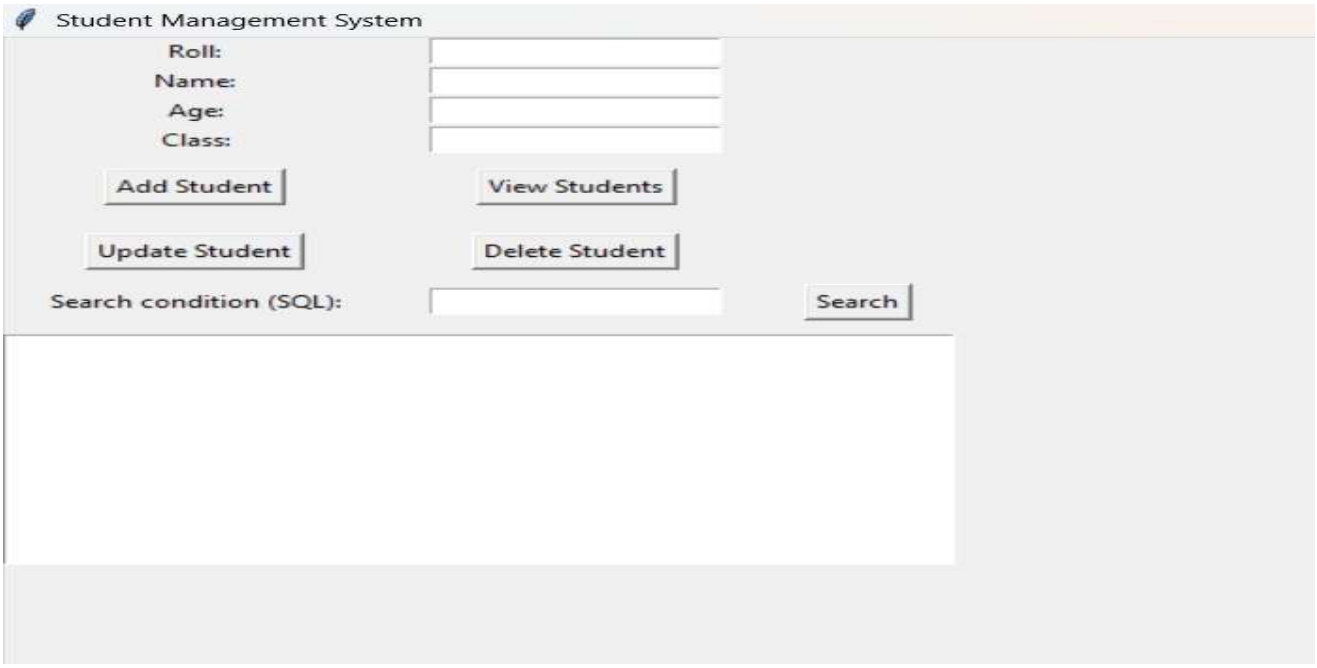
```
    if roll:
        try:
            # Fetch the record to ensure it exists
            cur.execute("SELECT * FROM stud WHERE roll=%s", (roll,))
            results = cur.fetchall()

            if not results:
                messagebox.showwarning("No Record", "No matching details available")
            else:
                # Update the record
                cur.execute("UPDATE stud SET name=%s, age=%s, class=%s WHERE roll=%s", (name,
age, clas, roll))
                con.commit()
                messagebox.showinfo("Success", "Record updated successfully")
                clear_entries()
        except Exception as e:
            messagebox.showerror("Error", str(e))
    else:
        messagebox.showwarning("Input Error", "Please enter roll number")

def clear_entries():
    roll_entry.delete(0, tk.END)
    name_entry.delete(0, tk.END)
```

```
    age_entry.delete(0, tk.END)
    class_entry.delete(0, tk.END)
    search_entry.delete(0, tk.END)

# GUI setup
root = tk.Tk()
root.title("Student Management System")

# Entry fields
tk.Label(root, text="Roll:").grid(row=0, column=0)
roll_entry = tk.Entry(root)
roll_entry.grid(row=0, column=1)

tk.Label(root, text="Name:").grid(row=1, column=0)
name_entry = tk.Entry(root)
name_entry.grid(row=1, column=1)

tk.Label(root, text="Age:").grid(row=2, column=0)
age_entry = tk.Entry(root)
age_entry.grid(row=2, column=1)

tk.Label(root, text="Class:").grid(row=3, column=0)
class_entry = tk.Entry(root)
class_entry.grid(row=3, column=1)
```

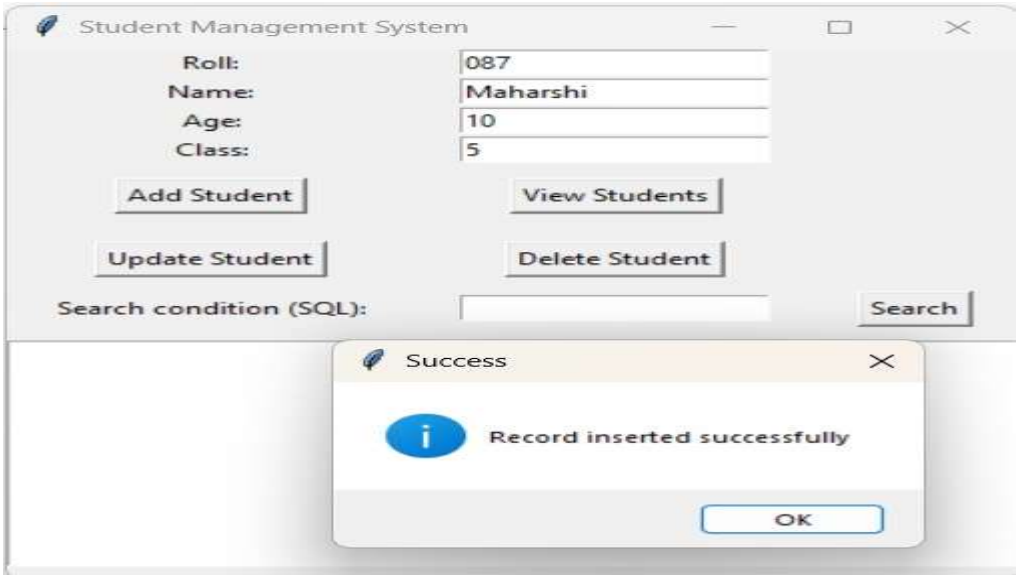
```
# Buttons
tk.Button(root, text="Add Student", command=stuInsert).grid(row=4, column=0, pady=10)
tk.Button(root, text="View Students", command=stuview).grid(row=4, column=1, pady=10)
tk.Button(root, text="Update Student", command=updatestu).grid(row=5, column=0, pady=10)
tk.Button(root, text="Delete Student", command=removestu).grid(row=5, column=1, pady=10)

# Search field
tk.Label(root, text="Search condition (SQL):").grid(row=6, column=0)
search_entry = tk.Entry(root)
search_entry.grid(row=6, column=1)
tk.Button(root, text="Search", command=searchstu).grid(row=6, column=2)

# Output text box
output_text = tk.Text(root, height=10, width=50)
output_text.grid(row=7, columnspan=3, pady=10)

# Main loop
root.mainloop()
```

OUTPUT:

INTERFACE:

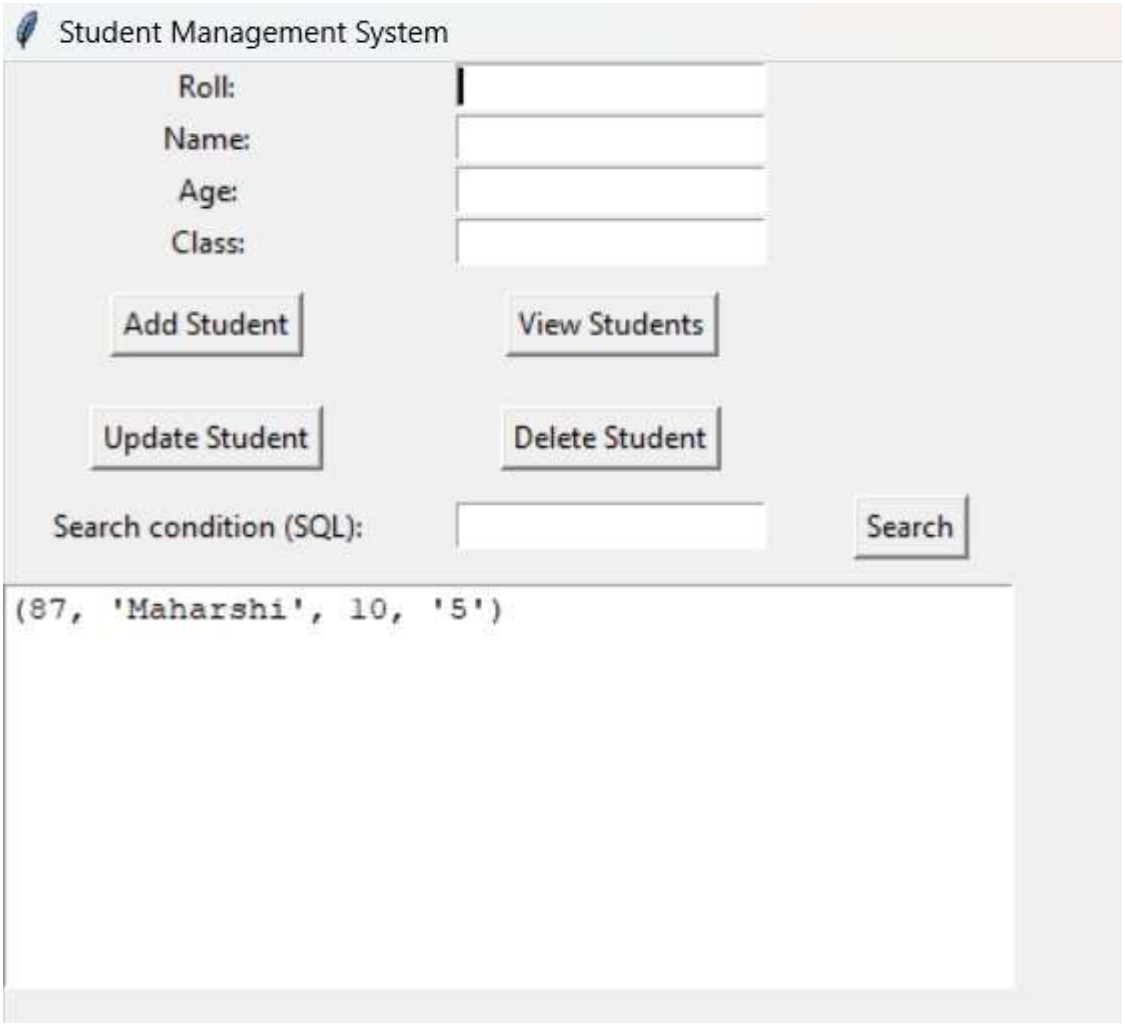


INSERT STUDENT DETAILS:



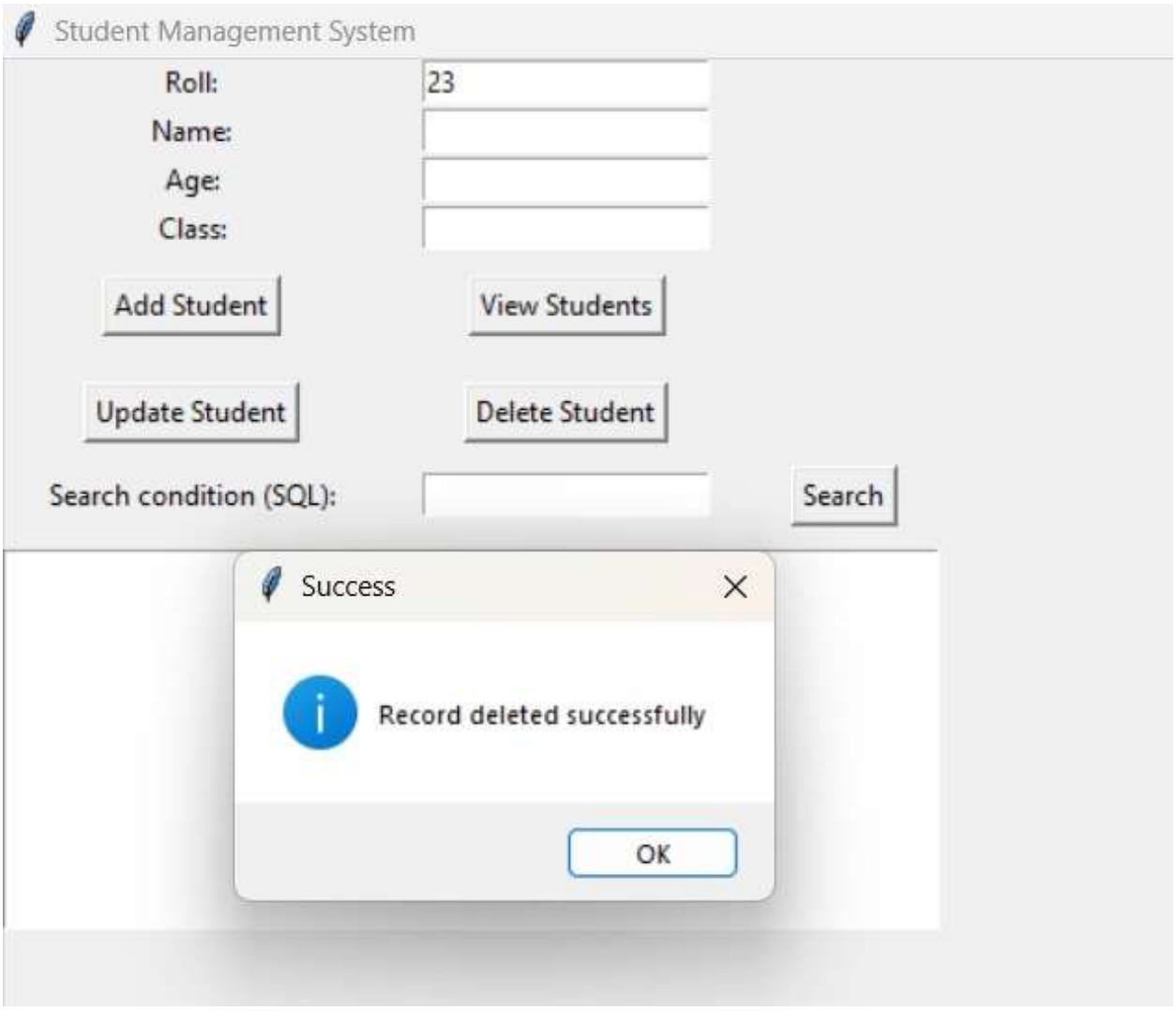
```
mysql> select * from stud;
+------+----------+------+-------+
| roll | name     | age  | class |
+------+----------+------+-------+
|   87 | Maharshi |   10 | 5     |
+------+----------+------+-------+
1 row in set (0.05 sec)
```

VIEW STUDENT DETAILS:

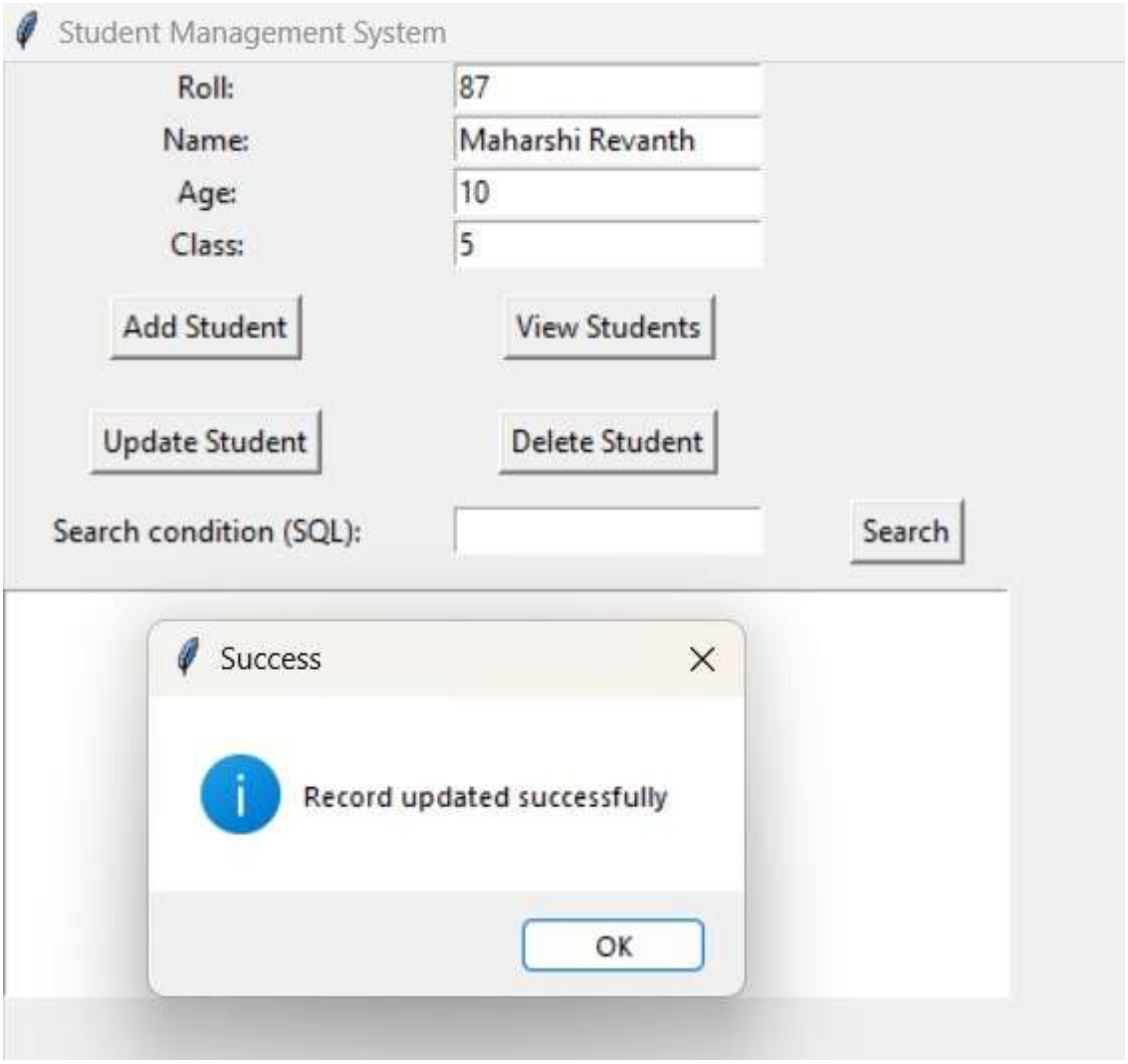


DELETE STUDENT DETAILS:

```
mysql> select * from stud;
+------+----------+------+-------+
| roll | name     | age  | class |
+------+----------+------+-------+
|   87 | Maharshi |   10 | 5     |
|   23 | Ram      |   15 | 10    |
+------+----------+------+-------+
2 rows in set (0.00 sec)
```



UPDATE STUDENT DETAILS:




```
mysql> select * from stud;
+------+------------------+------+-------+
| roll | name             | age  | class |
+------+------------------+------+-------+
|   87 | Maharshi Revanth |   10 | 5     |
+------+------------------+------+-------+
1 row in set (0.00 sec)
```

RESULT:

I have successfully completed the mini project “School Management System”.

DISCUSSION AND FUTURE WORKS:

The Student Management System project offers an intuitive interface for educational institutions to efficiently manage student records. Implemented using Python's Tkinter library and MySQL database, the system enables users to seamlessly add, view, update, and delete student information. It ensures data integrity and security through parameterized queries and error handling mechanisms. Future enhancements could include user authentication and authorization features, advanced search functionalities, integration with Learning Management Systems, mobile application development, data visualization, feedback channels, and internationalization support. These improvements aim to transform the system into a comprehensive platform that not only streamlines administrative tasks but also enhances communication, collaboration, and student engagement within the educational community.

REFERENCES:

➤ https://docs.oracle.com/javase/7/docs/api/

➤ https://www.javatpoint.com/java-swing

➤ https://stackoverflow.com/