Report of mini project

Title of project: Student Management System.

Name of student: Prerna karn bhosale.

Roll no:13124.

Aim: To design and develop a database system to manage student details such as name, roll number, department, marks, and attendance.

Use Case: This system is used by school/college administrators to store, view, and manage student records efficiently.

Mysql tables:

Microsoft Windows [Version 10.0.26200.6725]

(c) Microsoft Corporation. All rights reserved.

C:\Program Files\MySQL\MySQL Server 9.3\bin>mysql -h localhost -u root -p

Enter password: ****

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 9

Server version: 9.3.0 MySQL Community Server - GPL

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
mysql> CREATE DATABASE student_db;
Query OK, 1 row affected (1.339 sec)

mysql> USE student_db;
Database changed
mysql> CREATE TABLE students(student_id INT AUTO_INCREMENT PRIMARY KEY,name VARCHAR(100),roll_no VARCHAR(20),department VARCHAR(50));
Query OK, 0 rows affected (1.630 sec)

mysql> CREATE TABLE courses (course_id INT AUTO_INCREMENT PRIMARY KEY,course_name VARCHAR(100),department VARCHAR(50));
```

Query OK, 0 rows affected (0.467 sec)

mysql> CREATE TABLE marks (mark_id INT AUTO_INCREMENT PRIMARY KEY,student_id INT,course_id INT,marks_obtained INT,FOREIGN KEY (student_id) REFERENCES students(student_id),FOREIGN KEY (course_id) REFERENCES courses(course_id));

Query OK, 0 rows affected (1.053 sec)

mysql> CREATE TABLE attendance (att_id INT AUTO_INCREMENT PRIMARY KEY,student_id INT,total_classes INT,attended INT,FOREIGN KEY (student_id) REFERENCES students(student_id));

Query OK, 0 rows affected (0.792 sec)

Source code in python:

```
import mysql.connector


# Connect to MySQL
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root",
    database="student_db"
```

```
)
# Create a cursor object to execute SQL queries
cursor = conn.cursor()
# Step 2: Functions for each operation
def add_student():
    name = input("Enter name: ")
    roll_no = input("Enter roll number: ")
    department = input("Enter department: ")

    cursor.execute(
        "INSERT INTO students (name, roll_no, department) VALUES (%s, %s, %s)",
        (name, roll_no, department)
    )
    conn.commit()
    print("✅ Student added successfully!\n")

def view_students():
    cursor.execute("SELECT * FROM students")
    records = cursor.fetchall()
    print("\n--- Student List ---")
```

```
    for row in records:
        print(f"ID: {row[0]}, Name: {row[1]}, Roll No: {row[2]}, Department: {row[3]}")
    print()

def update_student():
    sid = input("Enter student ID to update: ")
    name = input("Enter new name: ")
    dept = input("Enter new department: ")
    cursor.execute(
        "UPDATE students SET name=%s, department=%s WHERE student_id=%s",
        (name, dept, sid)
    )
    conn.commit()
    print("✅ Record updated successfully!\n")

def delete_student():
    sid = input("Enter student ID to delete: ")
    cursor.execute("DELETE FROM students WHERE student_id=%s", (sid,))
    conn.commit()
```

```
    print("🗑️ Student deleted successfully!\n")


def add_marks():
    sid = input("Enter student ID: ")
    cid = input("Enter course ID: ")
    marks = input("Enter marks obtained: ")
    cursor.execute(
        "INSERT INTO marks (student_id, course_id, marks_obtained)
VALUES (%s, %s, %s)",
        (sid, cid, marks)
    )
    conn.commit()
    print("✅ Marks added successfully!\n")


def view_marks():
    cursor.execute(
        "SELECT s.name, c.course_name, m.marks_obtained "
        "FROM marks m "
        "JOIN students s ON m.student_id = s.student_id "
        "JOIN courses c ON m.course_id = c.course_id"
    )
```

```
    records = cursor.fetchall()
    print("\n--- Marks Report ---")
    for row in records:
        print(f"Student: {row[0]}, Course: {row[1]}, Marks: {row[2]}")
    print()

def add_attendance():
    sid = input("Enter student ID: ")
    total = int(input("Enter total classes: "))
    attended = int(input("Enter attended classes: "))
    cursor.execute(
        "INSERT INTO attendance (student_id, total_classes, attended)
VALUES (%s, %s, %s)",
        (sid, total, attended)
    )
    conn.commit()
    print("✅ Attendance added successfully!\n")

def view_attendance():
    cursor.execute(
        "SELECT s.name, a.total_classes, a.attended "
```

```
        "FROM attendance a "
        "JOIN students s ON a.student_id = s.student_id"
    )
    records = cursor.fetchall()
    print("\n--- Attendance Report ---")
    for row in records:
        percent = (row[2] / row[1]) * 100 if row[1] > 0 else 0
        print(f"Student: {row[0]}, Attendance: {percent:.2f}%")
    print()
# Step 3: Menu-driven program
while True:
    print("""
===== Student Management System =====
1. Add Student
2. View Students
3. Update Student
4. Delete Student
5. Add Marks
6. View Marks
7. Add Attendance
8. View Attendance
```

```
9. Exit
""")

    choice = input("Enter your choice: ")

    if choice == '1':
        add_student()
    elif choice == '2':
        view_students()
    elif choice == '3':
        update_student()
    elif choice == '4':
        delete_student()
    elif choice == '5':
        add_marks()
    elif choice == '6':
        view_marks()
    elif choice == '7':
        add_attendance()
    elif choice == '8':
        view_attendance()
```

```
    elif choice == '9':
        print("👋 Exiting...")
        break
    else:
        print("❌ Invalid choice! Try again.")
```

output:

```
PS C:\Users\HP>  python
"C:\Users\HP\OneDrive\Desktop\student_management.py"


===== Student Management System =====
1. Add Student
2. View Students
3. Update Student
4. Delete Student
5. Add Marks
6. View Marks
7. Add Attendance
8. View Attendance
9. Exit


Enter your choice: 1
```

Enter name: prerna

Enter roll number: 13124

Enter department: computer

✅ Student added successfully!

===== Student Management System =====

1. Add Student

2. View Students

3. Update Student

4. Delete Student

5. Add Marks

6. View Marks

7. Add Attendance

8. View Attendance

9. Exit

Enter your choice: 9

👋 Exiting...

Front end code:

```
import tkinter as tk
```

```
from tkinter import messagebox, ttk
import mysql.connector

# ===== MySQL Connection =====
import mysql.connector

conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root",  # <- put your MySQL password here
    database="student_db"
)

cursor = conn.cursor()

# ===== Functions =====
def add_student():
    name = name_entry.get()
    roll = roll_entry.get()
    dept = dept_entry.get()
    if not name or not roll or not dept:
```

```
        messagebox.showerror("Error", "All fields are required!")
        return
    cursor.execute(
        "INSERT INTO students (name, roll_no, department) VALUES (%s, %s, %s)",
        (name, roll, dept)
    )
    conn.commit()
    messagebox.showinfo("Success", "Student added successfully!")
    name_entry.delete(0, tk.END)
    roll_entry.delete(0, tk.END)
    dept_entry.delete(0, tk.END)
    view_students()

def view_students():
    for row in tree.get_children():
        tree.delete(row)
    cursor.execute("SELECT * FROM students")
    records = cursor.fetchall()
    for r in records:
        tree.insert("", tk.END, values=r)
```

```
def delete_student():
    selected_item = tree.selection()
    if not selected_item:
        messagebox.showerror("Error", "Select a student first")
        return
    sid = tree.item(selected_item)["values"][0]
    cursor.execute("DELETE FROM students WHERE student_id=%s",
(sid,))
    conn.commit()
    messagebox.showinfo("Success", "Student deleted successfully!")
    view_students()

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

# Input fields
tk.Label(root, text="Name").grid(row=0, column=0, padx=5, pady=5)
tk.Label(root, text="Roll No").grid(row=1, column=0, padx=5, pady=5)
tk.Label(root, text="Department").grid(row=2, column=0, padx=5,
pady=5)
```

```
name_entry = tk.Entry(root)
roll_entry = tk.Entry(root)
dept_entry = tk.Entry(root)

name_entry.grid(row=0, column=1, padx=5, pady=5)
roll_entry.grid(row=1, column=1, padx=5, pady=5)
dept_entry.grid(row=2, column=1, padx=5, pady=5)

# Buttons
tk.Button(root, text="Add Student",
command=add_student).grid(row=3, column=0, pady=5)
tk.Button(root, text="Delete Student",
command=delete_student).grid(row=3, column=1, pady=5)
tk.Button(root, text="View Students",
command=view_students).grid(row=3, column=2, pady=5)

# Treeview for student list
columns = ("ID", "Name", "Roll No", "Department")
tree = ttk.Treeview(root, columns=columns, show="headings")
for col in columns:
    tree.heading(col, text=col)
```

tree.grid(row=4, column=0, columnspan=3, padx=5, pady=5)

view_students() # Load initial data

root.mainloop()

output:

PS C:\Users\HP> python "C:\Users\HP\OneDrive\Desktop\student_management_gui.py"

Screenshot: