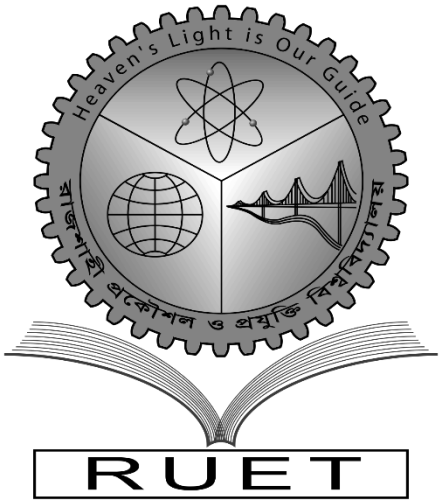
"Heaven's Light is Our Guide

Rajshahi University of Engineering & Technology
Rajshahi, Bangladesh



Department of Electrical & Computer Engineering (ECE-21)

Course Code: ECE 2216
Course Title: Database Systems Sessional
Experiment No: 02
Date of Submission: 30/09/2024

Submitted To:

Oishi Jyoti
Assistant Professor
Rajshahi University of Engineering & Technology

Submitted By:

Md Sharif Hossain
Roll: 2110053
ECE-21 Series

Experiment No. 01

Experiment Name: Managing Student Database and Conditional Data Logging in MySQL

Theory:

Efficient management of structured data is key in today's database systems for various practical uses. This experiment explores the core database tasks, including creating, updating, deleting, and conditionally modifying records in a MySQL relational database within a XAMPP setup.[1] It utilizes SQL (Structured Query Language) commands to organize and manipulate student data in a structured table format. [2]

Problem Statement:

1. Find students who are older than 20 and have a GPA above the average GPA of all students
2. Find the top 5 students with the highest fees paid, ordered by GPA (in descending order)
as a tiebreaker
3. List students who belong to the "Engineering" department, have a GPA greater than 3.5,
and are enrolled after 2020
4. Find students who are not active (i.e., enrollment_status = 'inactive') and have not paid
any fees (fees_paid = 0)
5. Calculate the total fees paid and average GPA for each department, but only for departments with more than 10 students

Software Used:

1. Xampp Control Panel
2. MySQL

Task: Creating database and table

```sql
CREATE DATABASE students_db;
USE students_db;
```

```sql
USE students_db;

CREATE TABLE students (
    student_id INT PRIMARY KEY,
    student_name VARCHAR(50),
    age INT,
    GPA DECIMAL(3, 2),
    department VARCHAR(50),
    year_of_admission INT,
    fees_paid DECIMAL(10, 2),
    credits_earned INT,
    enrollment_status VARCHAR(10)
);
```

```sql
INSERT INTO students (student_id, student_name, age, GPA, department, year_of_admission, fees_paid, credits_earned, enrollment_status)
VALUES
(1, 'Eleven', 21, 3.8, 'Engineering', 2021, 10000, 120, 'active'),
(2, 'Dustin', 22, 3.9, 'Science', 2020, 9000, 110, 'active'),
(3, 'Will', 19, 3.4, 'Science', 2022, 8500, 95, 'active'),
(4, 'Mike', 23, 3.7, 'Science', 2021, 9500, 115, 'inactive'),
(5, 'Max', 20, 3.5, 'Engineering', 2020, 12000, 130, 'active'),
(6, 'Eddie', 22, 4.0, 'Arts', 2019, 8000, 140, 'active'),
(7, 'Billy', 24, 2.9, 'Engineering', 2022, 5000, 60, 'active'),
(8, 'Alexei', 25, 3.2, 'Business', 2018, 7500, 100, 'inactive'),
(9, 'Steve', 21, 3.8, 'Science', 2021, 10500, 120, 'active'),
(10, 'Robin', 20, 3.6, 'Engineering', 2022, 11000, 125, 'active'),
(11, 'Lucas', 18, 2.7, 'Engineering', 2023, 4000, 50, 'active'),
(12, 'Nancy', 23, 3.9, 'Business', 2019, 9500, 135, 'active');
```

Output:

	student_id	student_name	age	GPA	department	year_of_admission	fees_paid	credits_earned	enrollment_status
Edit Copy Delete	1	Eleven	21	3.80	Engineering	2021	10000.00	120	active
Edit Copy Delete	2	Dustin	22	3.90	Science	2020	9000.00	110	active
Edit Copy Delete	3	Will	19	3.40	Science	2022	8500.00	95	active
Edit Copy Delete	4	Mike	23	3.70	Science	2021	9500.00	115	inactive
Edit Copy Delete	5	Max	20	3.50	Engineering	2020	12000.00	130	active
Edit Copy Delete	6	Eddie	22	4.00	Arts	2019	8000.00	140	active
Edit Copy Delete	7	Billy	24	2.90	Engineering	2022	5000.00	60	active
Edit Copy Delete	8	Alexei	25	3.20	Business	2018	7500.00	100	inactive
Edit Copy Delete	9	Steve	21	3.80	Science	2021	10500.00	120	active
Edit Copy Delete	10	Robin	20	3.60	Engineering	2022	11000.00	125	active
Edit Copy Delete	11	Lucas	18	2.70	Engineering	2023	4000.00	50	active
Edit Copy Delete	12	Nancy	23	3.90	Business	2019	9500.00	135	active

Task 1: Find students older than 20 with GPA above the average GPA of all students

Code:

```
SELECT *
FROM students
WHERE age > 20
AND GPA > (SELECT AVG(GPA) FROM students);

```

Output:

	student_id	student_name	age	GPA	department	year_of_admission	fees_paid	credits_earned	enrollment_status
Edit Copy Delete	1	Eleven	21	3.80	Engineering	2021	10000.00	120	active
Edit Copy Delete	2	Dustin	22	3.90	Science	2020	9000.00	110	active
Edit Copy Delete	4	Mike	23	3.70	Science	2021	9500.00	115	inactive
Edit Copy Delete	6	Eddie	22	4.00	Arts	2019	8000.00	140	active
Edit Copy Delete	9	Steve	21	3.80	Science	2021	10500.00	120	active
Edit Copy Delete	12	Nancy	23	3.90	Business	2019	9500.00	135	active

Task 2: Find the top 5 students with the highest fees paid, ordered by GPA (as a tiebreaker)

Code:

```
SELECT *
FROM students
ORDER BY fees_paid DESC, GPA DESC
LIMIT 5;

```

Output:

	student_id	student_name	age	GPA 2	department	year_of_admission	fees_paid 1	credits_earned	enrollment_status
Edit Copy Delete	5	Max	20	3.50	Engineering	2020	12000.00	130	active
Edit Copy Delete	10	Robin	20	3.60	Engineering	2022	11000.00	125	active
Edit Copy Delete	9	Steve	21	3.80	Science	2021	10500.00	120	active
Edit Copy Delete	1	Eleven	21	3.80	Engineering	2021	10000.00	120	active
Edit Copy Delete	12	Nancy	23	3.90	Business	2019	9500.00	135	active

Task 3: List students from the "Engineering" department with a GPA greater than 3.5 and enrolled after 2020

Code:

```sql
SELECT *
FROM students
WHERE department = 'Engineering'
AND GPA > 3.5
AND year_of_admission > 2020;
```

Output:

	student_id	student_name	age	GPA	department	year_of_admission	fees_paid	credits_earned	enrollment_status
Edit Copy Delete	1	Eleven	21	3.80	Engineering	2021	10000.00	120	active
Edit Copy Delete	10	Robin	20	3.60	Engineering	2022	11000.00	125	active

Task 4: Find students who are not active and have not paid any fees (fees_paid = 0)

Code:

```sql
SELECT *
FROM students
WHERE enrollment_status = 'inactive'
AND fees_paid = 0;
```

Output:

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0005 seconds.)

```
SELECT * FROM students WHERE enrollment_status = 'inactive' AND fees_paid = 0;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

student_id	student_name	age	GPA	department	year_of_admission	fees_paid	credits_earned	enrollment_status

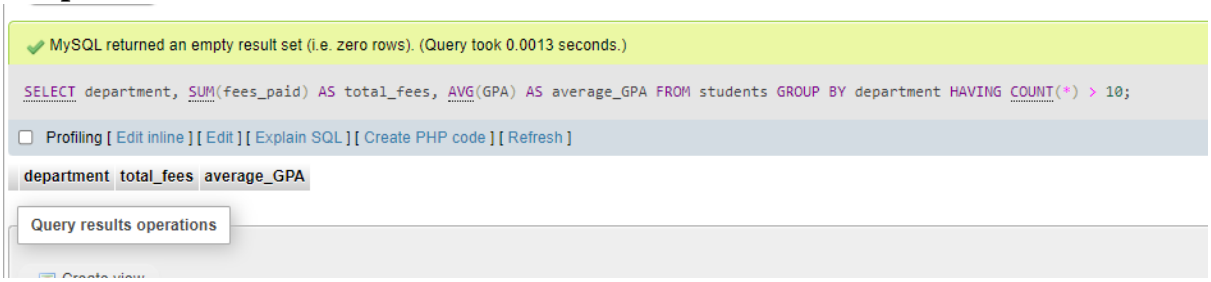
Query results operations

Task 5: Calculate the total fees paid and average GPA for each department with more than 10 students

Code:

```
SELECT department, SUM(fees_paid) AS total_fees, AVG(GPA) AS average_GPA
FROM students
GROUP BY department
HAVING COUNT(*) > 10;
```

Output:



Discussion:

This experiment looked at basic database tasks using MySQL in XAMPP. We started by creating a database called “student_db” and a table named “students.” We changed a column name from “favorite_subject” to “major” to show how to manage changes in the table.[3] We deleted records for students who scored below 30 marks to keep the data relevant. We also added a new column called “log” and filled it with values based on the semester. Overall, this experiment helped us understand important tasks like creating, changing, and managing data in a simple way, making the database more useful and accurate.[4]

References:

[1] “MySQL | Common MySQL Queries - GeeksforGeeks.” Accessed: Sep. 30, 2024. [Online]. Available: https://www.geeksforgeeks.org/mysql-common-mysql-queries/

[2] “MySQL ? Queries.” Accessed: Sep. 30, 2024. [Online]. Available: https://www.tutorialspoint.com/mysql/mysql-queries.htm

[3] “MySQL SQL.” Accessed: Sep. 30, 2024. [Online]. Available: https://www.w3schools.com/mysql/mysql_sql.asp

[4] “MySQL Queries - javatpoint.” Accessed: Sep. 30, 2024. [Online]. Available: https://www.javatpoint.com/mysql-queries