"Heaven's Light is Our Guide"

Rajshahi University of Engineering & Technology

Department of Electrical & Computer Engineering

Course Title

Data Base Systems Sessional

Course No: ECE 2216

Lab Report No. 01

Date of submission: 16/09/2024

Submitted to:	**Submitted by:**
Oishi Jyoti Assistant Professor, Department of ECE, RUET	Md. Sabbir Hosen Roll: 2110019 Reg. No: 1073/2021-2022 Department of ECE, RUET

Experiment No. : 02

Experiment Name : Basic queries of MySQL

Task:

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

Objectives:

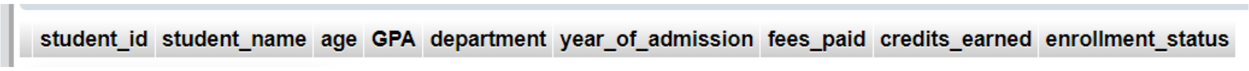
The primary objective of this lab is to create a SQL-based system for storing and managing student information in a structured database. The system is designed to:

- Record details of students such as student_id, name, age, GPA, department, year_of_admission, fees_paid, credits_earned, and enrollment_status.
- Perform queries for various use cases such as finding students based on GPA, fees, and enrollment status.
- Display the data in a table format with SQL queries.
- Generate reports for administrative and analytical purposes.

Queries & Output :

Create Table:

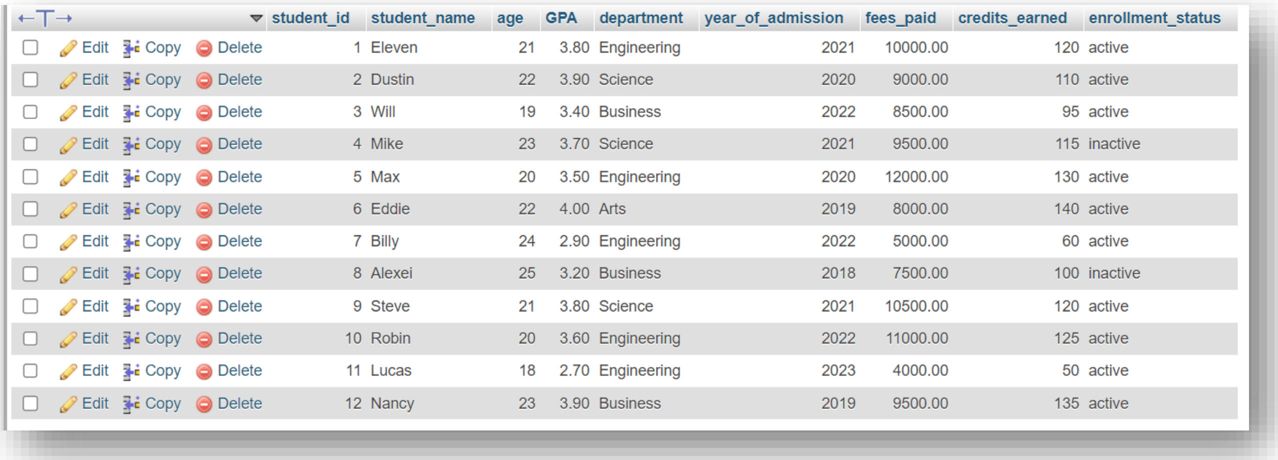
```
CREATE TABLE students_info (

    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)
);
```



student_id	student_name	age	GPA	department	year_of_admission	fees_paid	credits_earned	enrollment_status

Insert Data :

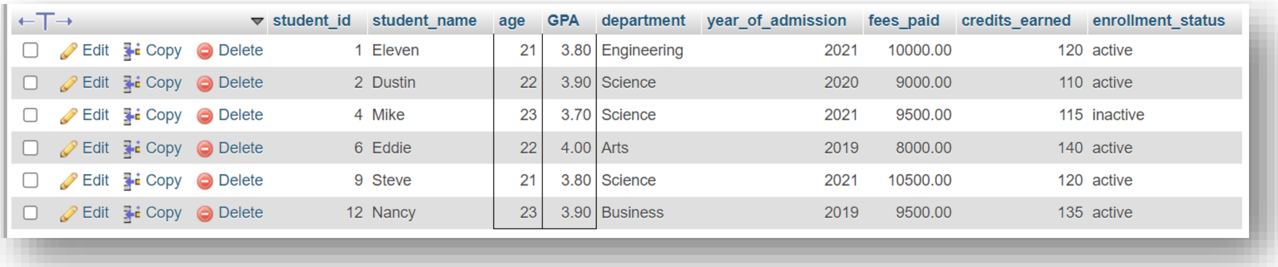
```
INSERT INTO students_info (student_id, student_name, age, GPA,
department, year_of_admission, fees_paid, credits_earned,
enrollment_status)

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, 'Business', 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');
```



←T→	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	Business	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 who are older than 20 and have a GPA above the average GPA of all students.

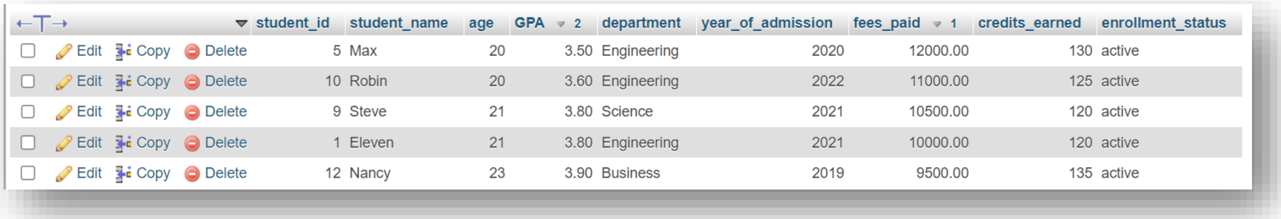
```
SELECT *

FROM students_info

WHERE age > 20

  AND GPA > (SELECT AVG(GPA) FROM students_info);
```



←T→			▼	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 (in descending order) as a tiebreaker

```
SELECT *

FROM students_info

ORDER BY fees_paid DESC, GPA DESC

LIMIT 5;
```

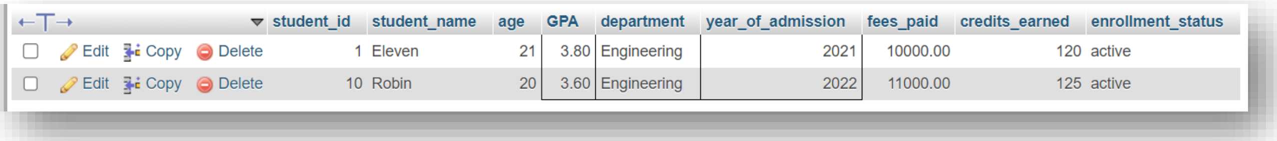


←T→			▼	student_id	student_name	age	GPA ▼ 2	department	year_of_admission	fees_paid ▼ 1	credits_earned	enrollment_status
□	Edit	Copy	Delete	5	Max	20	3.50	Engineering	2020	12000.00	130	active
□	Edit	Copy	Delete	10	Robin	20	3.60	Engineering	2022	11000.00	125	active
□	Edit	Copy	Delete	9	Steve	21	3.80	Science	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 who belong to the "Engineering" department, have a GPA greater than 3.5, and are enrolled after 2020

```
SELECT *
```

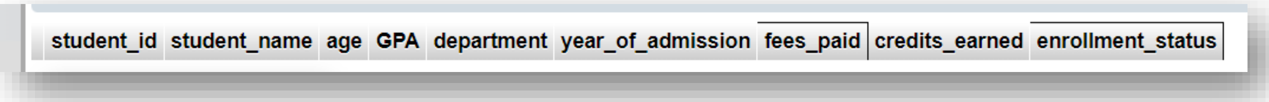
```
FROM students_info
WHERE department = 'Engineering'
  AND GPA > 3.5
  AND year_of_admission > 2020;
```



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

Task 4 : Find students who are not active (i.e., enrollment_status = 'inactive') and have not paid any fees (fees_paid = 0)

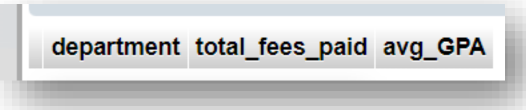
```
SELECT *
FROM students_info
WHERE enrollment_status = 'inactive'
  AND fees_paid = 0;
```



student_id	student_name	age	GPA	department	year_of_admission	fees_paid	credits_earned	enrollment_status

Task 5 : Calculate the total fees paid and average GPA for each department, but only for departments with more than 10 students.

```
SELECT department, SUM(fees_paid) AS total_fees_paid, AVG(GPA) AS
avg_GPA
FROM students_info
GROUP BY department
HAVING COUNT(student_id) > 10;
```



department	total_fees_paid	avg_GPA

Discussion:

The lab demonstrated the use of SQL for database management and analysis of student data. The queries were designed to provide insights based on conditions such as age, GPA, enrollment status, and fees paid. These queries help in organizing data efficiently and retrieving valuable information for administrative purposes.

The student data was carefully structured to ensure that it met all the criteria for testing various SQL queries. This structured approach can be extended to larger datasets for more complex queries, enabling scalability for university databases or other educational institutions.

Additionally, the use of aggregate functions such as SUM, AVG, and COUNT provided a deeper understanding of how SQL can be utilized for statistical and analytical purposes, making it a valuable tool for academic data management.

Conclusion:

The SQL-based student information management system successfully stored, queried, and analyzed data based on various conditions. The lab helped to solidify knowledge about SQL operations, including conditional queries, aggregate functions, and the importance of data structure in database management.

References:

[1] W3Schools, *SQL Tutorial*, https://www.w3schools.com/sql/, accessed October 2024.

[2] Codecademy, *Learn SQL*, https://www.codecademy.com/learn/learn-sql, accessed October 2024.

[3] TutorialsPoint, *SQL Tutorial*, https://www.tutorialspoint.com/sql/index.htm, accessed October 2024.

[4] J. Celko, *SQL for Smarties: Advanced SQL Programming*, 4th ed. Morgan Kaufmann, 2015.