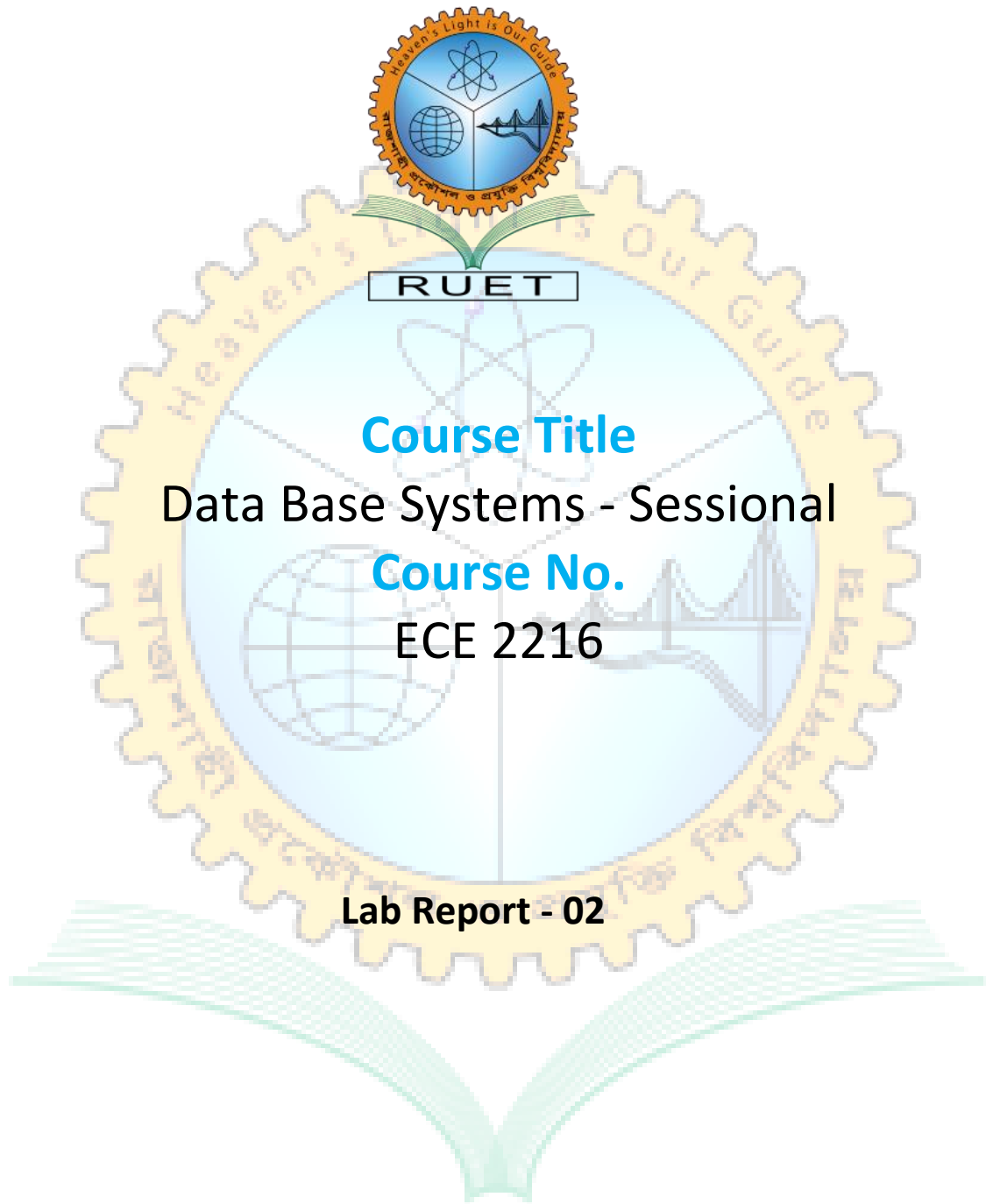


*Havens Light is Our Guide*

# Rajshahi University of Engineering & Technology



## Course Title

Data Base Systems - Sessional

## Course No.

ECE 2216

## Lab Report - 02

### Submitted By

Prattay Deb Soom

Roll: 2110056

Reg No: 1110/2021-2022

Department of ECE, RUET

### Submitted To

Oishi Jyoti

Assistant Professor

Department of ECE, RUET

## Experiment No. 02

### Experiment Name: Analysis of Student Data Using SQL Queries.

#### Theory:

The lab revolves around fundamental SQL concepts, including the use of SQL clauses such as SELECT, WHERE, GROUP BY, HAVING, and ORDER BY. These clauses facilitate filtering, sorting, and grouping data to obtain relevant information from relational databases. The WHERE clause is essential for filtering rows based on specific conditions, while GROUP BY aggregates data into groups sharing common attributes. The HAVING clause allows for filtering based on aggregate results, applied after the grouping operation. Subqueries are also a key feature, enabling dynamic filtering of results when aggregate conditions are required in the WHERE clause. This theoretical framework underpins the SQL operations performed in the experiment.

#### 1<sup>st</sup> step: Creating the given database and table.

#### SQL Commands:

```
1 USE students;
2
3 CREATE TABLE student_info_3 (
4     student_id INT,
5     student_name VARCHAR(100),
6     age INT,
7     GPA DECIMAL(3,2),
8     department VARCHAR(100),
9     year_of_admission INT,
10    fees_paid INT,
11    credits_earned INT,
12    enrollment_status VARCHAR(100)
13 );
14 INSERT INTO student_info_3 (student_id, student_name, age, GPA, department, year_of_admission, fees_paid, credits_earned, enrollment_status)
15 VALUES
16 (1, 'Eleven', 21, 3.80, 'Engineering', 2021, 10000, 120, 'active'),
17 (2, 'Dustin', 22, 3.90, 'Science', 2020, 9000, 110, 'active'),
18 (3, 'Will', 19, 3.40, 'Business', 2022, 8500, 95, 'active'),
19 (4, 'Mike', 23, 3.70, 'Science', 2021, 9500, 115, 'inactive'),
20 (5, 'Max', 20, 3.50, 'Engineering', 2020, 12000, 130, 'active'),
21 (6, 'Eddie', 22, 4.00, 'Arts', 2019, 8000, 140, 'active'),
22 (7, 'Billy', 24, 2.90, 'Engineering', 2022, 5000, 60, 'active'),
23 (8, 'Alexei', 25, 3.20, 'Business', 2018, 7500, 100, 'inactive'),
24 (9, 'Steve', 21, 3.80, 'Science', 2021, 10500, 120, 'active'),
25 (10, 'Robin', 20, 3.60, 'Engineering', 2022, 11000, 125, 'active'),
26 (11, 'Lucas', 18, 2.70, 'Engineering', 2023, 4000, 50, 'active'),
27 (12, 'Nancy', 23, 3.90, 'Business', 2019, 9500, 135, 'active');
```

### Output:

student_id	student_name	age	GPA	department	year_of_admission	fees_paid	credits_earned	enrollment_status
1	Eleven	21	3.80	Engineering	2021	10000	120	active
2	Dustin	22	3.90	Science	2020	9000	110	active
3	Will	19	3.40	Business	2022	8500	95	active
4	Mike	23	3.70	Science	2021	9500	115	inactive
5	Max	20	3.50	Engineering	2020	12000	130	active
6	Eddie	22	4.00	Arts	2019	8000	140	active
7	Billy	24	2.90	Engineering	2022	5000	60	active
8	Alexei	25	3.20	Business	2018	7500	100	inactive
9	Steve	21	3.80	Science	2021	10500	120	active
10	Robin	20	3.60	Engineering	2022	11000	125	active
11	Lucas	18	2.70	Engineering	2023	4000	50	active
12	Nancy	23	3.90	Business	2019	9500	135	active

**Task 1.** Find students who are older than 20 and have a GPA above the average GPA of all students.

### SQL Commands:

```
1 SELECT student_name, age, GPA FROM student_info_3
2 WHERE age > 20 AND GPA > (SELECT AVG(GPA) FROM student_info_3);
3
```

### Output:

student_name	age	GPA
Eleven	21	3.80
Dustin	22	3.90
Mike	23	3.70
Eddie	22	4.00
Steve	21	3.80
Nancy	23	3.90

**Task 2.** Find the top 5 students with the highest fees paid, ordered by GPA (in descending order) as a tiebreaker.

### SQL Commands:

```
1 SELECT student_name, fees_paid, GPA FROM student_info_3
2 ORDER BY fees_paid DESC, GPA DESC LIMIT 5;
```

**Output:**

student_name	fees_paid	GPA
Max	12000	3.50
Robin	11000	3.60
Steve	10500	3.80
Eleven	10000	3.80
Nancy	9500	3.90

**Task 3.** List students who belong to the "Engineering" department, have a GPA greater than 3.5, and are enrolled after 2020.

**SQL Commands:**

```
1 SELECT student_name, GPA, year_of_admission FROM student_info_3
2 WHERE department = 'Engineering' AND GPA > 3.5 AND year_of_admission > 2020;
3
```

**Output:**

student_name	GPA	year_of_admission
Eleven	3.80	2021
Robin	3.60	2022

**Task 4.** Find students who are not active (i.e., enrollment\_status = 'inactive') and have not paid any fees (i.e., fees\_paid = 0).

**SQL Commands:**

```
1 SELECT student_name, enrollment_status, fees_paid FROM student_info_3
2 WHERE enrollment_status = 'inactive' AND fees_paid = 0;
3
```

**Output:**

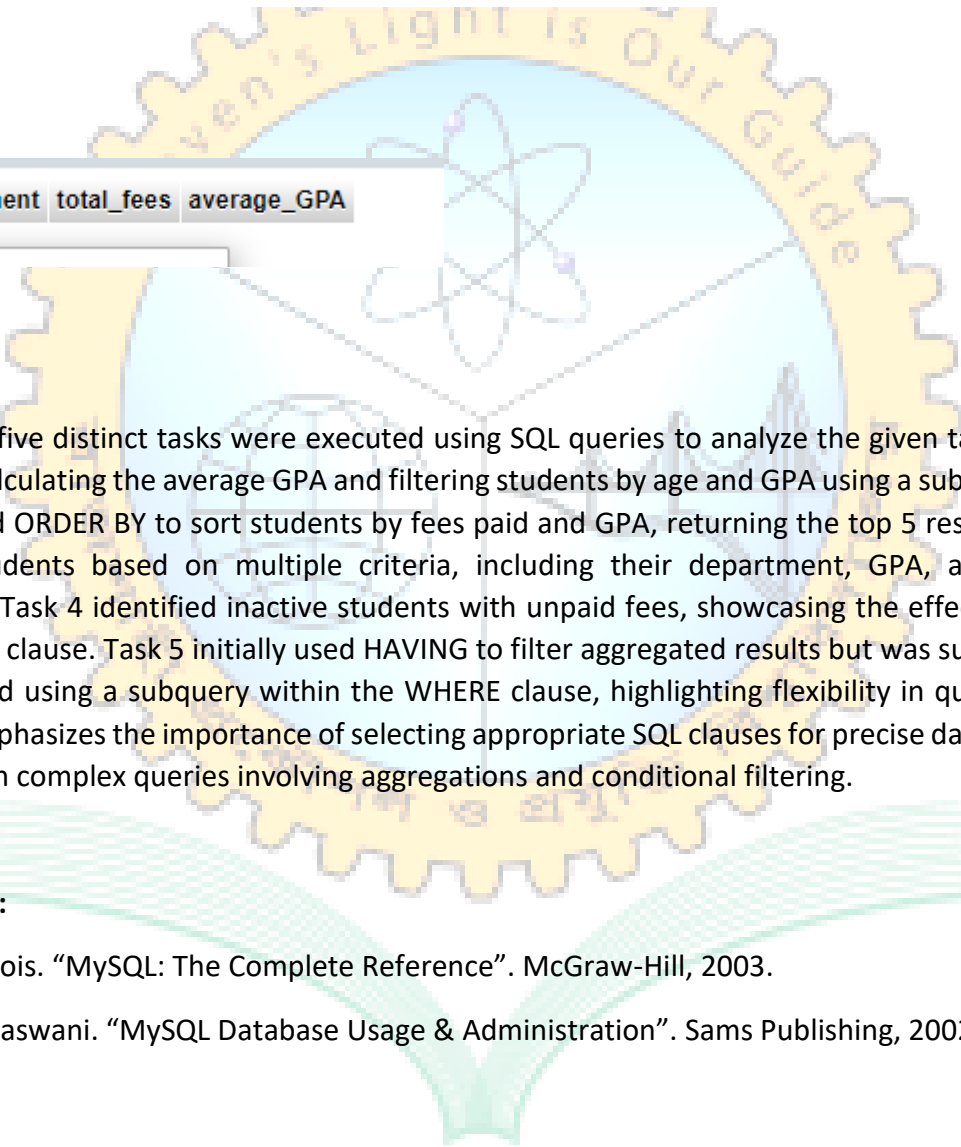
student_name	enrollment_status	fees_paid
--------------	-------------------	-----------

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

#### SQL Commands:

```
1 SELECT department, SUM(fees_paid) AS total_fees, AVG(GPA) AS average_GPA FROM student_info_3
2 GROUP BY department HAVING COUNT(student_id) > 10;
3
```

#### Output:



department	total_fees	average_GPA

#### Discussion:

In this lab, five distinct tasks were executed using SQL queries to analyze the given table. Task 1 involved calculating the average GPA and filtering students by age and GPA using a subquery. Task 2 employed ORDER BY to sort students by fees paid and GPA, returning the top 5 results. Task 3 filtered students based on multiple criteria, including their department, GPA, and year of admission. Task 4 identified inactive students with unpaid fees, showcasing the effective use of the WHERE clause. Task 5 initially used HAVING to filter aggregated results but was subsequently restructured using a subquery within the WHERE clause, highlighting flexibility in query design. This lab emphasizes the importance of selecting appropriate SQL clauses for precise data retrieval, especially in complex queries involving aggregations and conditional filtering.

#### References:

1. Paul DuBois. "MySQL: The Complete Reference". McGraw-Hill, 2003.
2. Vikram Vaswani. "MySQL Database Usage & Administration". Sams Publishing, 2002.

RUET