Heaven's Light is Our Guide

Rajshahi University of Engineering and Technology

Department of Electrical & Computer Engineering

Lab Report

Course Code : ECE 2216

Course Title : Data Base System Sessional

Lab Report NO. : 02

Submission Date: 30 September, 2024

Submitted To
Oishi Jyoti
Assistant Professor
Department of Electrical and
Computer Engineering, RUET

Submitted By
Name: Md. Abu Nayeem
Roll: 2110037
Department of Electrical and
Computer Engineering, RUET

Problem Statement:

Students Table

student_id	student_name	age	GPA	department	year_of_admission	fees_paid	credits_earned	enrollment_status
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

Tasks:

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.

Required Tools:

1. XAMPP Control Panel
2. Laptop

Task 1

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

Solving Procedure:

1. **Set the conditions**: Students older than 20 with a GPA higher than the average GPA.

2. **Use SELECT statement:** Choose the relevant columns.

3. **Apply WHERE clause**: Filter using age > 20 and use a sub-query to calculate the average GPA (GPA > (SELECT AVG(GPA) FROM StudentsTable)).

4. **Execute the query**: Retrieve the filtered data.

.

Query:

```
SELECT * from studentstable WHERE age>20 AND  GPA>(SELECT Avg(GPA) FROM studentstable);
```

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
4	Mike	23	3.70	Science	2021	9500	115	inactive
6	Eddie	22	4.00	Arts	2019	8000	140	active
9	Steve	21	3.80	Science	2021	10500	120	active
12	Nancy	23	3.90	Business	2019	9500	135	active

Task 2

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

Solving Procedure:

1. The SELECT statement retrieves columns of interest from the Students table.
2. The ORDER BY clause sorts the results by fees_paid in descending order and then by GPA in descending order for tie-breaking.
3. The LIMIT 5 clause restricts the output to the top 5 students.

Query:

```
1 SELECT * from studentstable 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
5	Max	20	3.50	Engineering	2020	12000	130	active
10	Robin	20	3.60	Engineering	2022	11000	125	active
9	Steve	21	3.80	Science	2021	10500	120	active
1	Eleven	21	3.80	Engineering	2021	10000	120	active
12	Nancy	23	3.90	Business	2019	9500	135	active

Task 3

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

Solving Procedure:

1. **Set the conditions**: Students from the Engineering department, GPA > 3.5, admitted after 2020.

2. **Use SELECT statement**: Choose the relevant columns.

3. **Apply WHERE clause:** Filter using department = 'Engineering', GPA > 3.5, and year_of_admission > 2020.

4. **Execute the query**: Retrieve the filtered data.

Query:

```
1 SELECT * from studentstable 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
1	Eleven	21	3.80	Engineering	2021	10000	120	active
10	Robin	20	3.60	Engineering	2022	11000	125	active

Task 4

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

Solving Procedure:

1. The SELECT statement retrieves columns of interest from the Students table.
2. The WHERE clause filters the students to only those who are inactive (enrollment_status = 'inactive') and have not paid any fees (fees_paid = 0).

Query:

```
1 SELECT * from studentstable where enrollment_status='inactive' And fees_paid=0;
```

Output:

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, but only for departments with more than 10 students.

Solving Procedure:

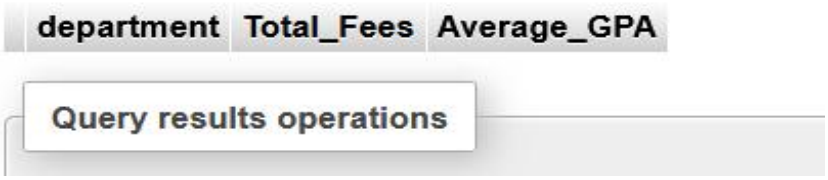
1.Group by department: Use SUM(fees_paid) and AVG(gpa) to calculate total fees and average GPA per department.

2.Filter results: The HAVING COUNT(*) > 10 ensures only departments with more than 10 students are included.

Query:

```
1 SELECT department,sum(fees_paid) as Total_Fees,AVG(gpa) As Average_GPA FROM studentstable group by department HAVING COUNT(*)>10;
2
```

Output:



Discussion:

During the lab, five SQL tasks were performed to analyze the 'Students Table', with challenges including writing SELECT statements correctly and understanding the concept of a tiebreaker, which is used to resolve ties in sorting. In Task 1, calculating the average GPA in a WHERE clause required a subquery. Task 2 introduced the tiebreaker concept to sort fees and GPA. Task 3 involved selecting students based on multiple conditions, initially challenging due to difficulties in using SELECT in one line. Task 4 successfully filtered inactive students who hadn't paid fees. Task 5 required a subquery to calculate total fees alongside department-wise groupings. These tasks emphasized the importance of query structure and syntax.

Reference:

[1] W3Schools.com, "SQL Syntax," *W3Schools*, https://www.w3schools.com/sql/sql_syntax.asp (accessed Sep. 30, 2024).

[2] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 7th ed. Boston, MA, USA: Pearson, 2016, pp. 235-240.