"Heaven's Light is Our Guide"

Rajshahi University of Engineering & Technology, Rajshahi

Department of Electrical & Computer Engineering

Course Code : ECE 2216

Course Title : Database Systems Sessional

Experiment No. : 02

Experiment Date : 23 September 2024

Submission Date : 30 September 2024

Submitted To-
Oishi Jyoti
Assistant Professor,
ECE, RUET

Submitted By-
Proma Sarker
Roll Number: 2110049
Reg. Number: 1103
Session: 2021-2022

Experiment Number: 02

Experiment Name: Database Management with MySQL: Query Operations on Students Table

Objectives:

To perform various SQL queries to analyze student information based on different criteria and to derive meaningful insights from the data.

Theory:

This experiment focuses on querying a student database to retrieve information based on specific conditions. SQL commands will be utilized to find students by age, GPA, fees, and enrollment status. Additionally, we will calculate aggregate values for further analysis.

Creating Database and Table:

SQL Commands:

```sql
CREATE DATABASE StudentDB;
USE StudentDB;

CREATE TABLE Students (
    student_id INT,
    student_name VARCHAR(50),
    age INT,
    GPA FLOAT,
    department VARCHAR(50),
    year_of_admission INT,
    fees_paid INT,
    credits_earned INT,
    enrollment_status VARCHAR(20)
);

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, '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');
```

Output:

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	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

Figure 1: Database and table creation output

Problem Statements:

1. Find students older than 20 with a GPA above the average GPA

SQL Command:

```
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
1	Eleven	21	3.8	Engineering	2021	10000	120	active
2	Dustin	22	3.9	Science	2020	9000	110	active
4	Mike	23	3.7	Science	2021	9500	115	inactive
6	Eddie	22	4	Arts	2019	8000	140	active
9	Steve	21	3.8	Science	2021	10500	120	active
12	Nancy	23	3.9	Business	2019	9500	135	active

Figure 2: Students older than 20 with GPA above average

2. Find the top 5 students with the highest fees paid, ordered by GPA

SQL Command:

```
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
5	Max	20	3.5	Engineering	2020	12000	130	active
10	Robin	20	3.6	Engineering	2022	11000	125	active
9	Steve	21	3.8	Science	2021	10500	120	active
1	Eleven	21	3.8	Engineering	2021	10000	120	active
12	Nancy	23	3.9	Business	2019	9500	135	active

Figure 3: Top 5 students with the highest fees paid

3. List students in Engineering with a GPA greater than 3.5, enrolled after 2020

SQL Command:

```
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
1	Eleven	21	3.8	Engineering	2021	10000	120	active
10	Robin	20	3.6	Engineering	2022	11000	125	active

Figure 4: Engineering students with GPA greater than 3.5

4. Find inactive students with no fees paid

SQL Command:

```
SELECT * FROM Students
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

Figure 5: Inactive students with no fees paid

5. Calculate total fees paid and average GPA for departments with more than 10 students

SQL Command:

```
SELECT department, SUM(fees_paid) AS Total_Fees, AVG(GPA) AS Avg_GPA
FROM Students
GROUP BY department
HAVING COUNT(*) > 10;
```

Output:

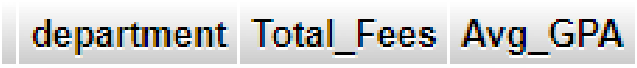

Figure 6: Total fees and average GPA by department

Discussion:

This experiment demonstrated the use of SQL queries to analyze student data effectively. We explored multiple queries to filter students based on age, GPA, fees paid, and enrollment status. Furthermore, we calculated aggregate values to understand the financial contributions of departments. Such operations are essential for managing educational databases and making informed decisions based on the analyzed data.

References

[1] MySQL Documentation, *MySQL Reference Manual*, [Online]. Available: `https://dev.mysql.com/doc/`. Accessed: September 2024.

[2] W3Schools, *SQL Tutorial*, [Online]. Available: `https://www.w3schools.com/sql/sql_intro.asp`. Accessed: September 2024.