Heaven's Light is Our Guide

Rajshahi University of Engineering & Technology

Department of Electrical & Computer Engineering

ECE-21

Course Code: ECE 2216

Course Title: Data Base Systems Sessional

Experiment No: 02

Submitted by :

MD Tonmay Hossain Jifat
Roll: 2110012
Dept. of ECE
RUET

Submitted to:
Oishi Jyoti
Assistant Professor
Dept. of ECE
RUET

Experiment Name: Database Query using MySQL.

Theory: An effective tool for organizing and dealing with relational databases is SQL (Structured Query Language). It enables users to carry out a variety of tasks, including data retrieval, addition, updating, and removal. SQL is essential to database queries because it provides a consistent approach to managing structured data. SQL queries range from basic data retrieval to sophisticated data filtering, grouping, and aggregation. Data Query Language (DQL), Data Manipulation Language (DML), Data Definition Language (DDL), and Data Control Language (DCL) are the categories into which the language is separated, and each one offers varying degrees of control over database management.

The ability of SQL to carry out intricate data operations with straightforward commands is one of its main features. While more complex features like JOIN, GROUP BY, HAVING, and ORDER BY enable effective data structure and filtering, basic commands like SELECT, UPDATE, INSERT, and DELETE allow users to manage data directly. AVG(), COUNT(), and SUM() are examples of aggregate methods that are commonly used to extract insights from data sets. Additionally, SQL is appropriate for more complex data analysis since it includes conditional logic and subqueries. Because of its adaptability, SQL is crucial in fields including scientific research, web development, and corporate intelligence.

Objective:

- To Learn about Aggregate Functions.
- Learing about Subqueries and how to Use them.

Tasks:

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

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

Query and Output:

Creating a new students table:

```sql
CREATE TABLE students (
    student_id INT PRIMARY KEY,
    student_name VARCHAR(50),
    age INT,
    GPA DECIMAL(2,1),
    department VARCHAR(50),
    year_of_admission INT,
    fees_paid DECIMAL(10,2),
    credits_earned INT,
    enrollment_status VARCHAR(10)
);
```

Query to create a new tables:

```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, '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, 3.8, '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:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**student_id**	int(11)			No	*None*			Change Drop More
2	**student_name**	varchar(50)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
3	**age**	int(11)			Yes	*NULL*			Change Drop More
4	**GPA**	decimal(2,1)			Yes	*NULL*			Change Drop More
5	**department**	varchar(50)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
6	**year_of_admission**	int(11)			Yes	*NULL*			Change Drop More
7	**fees_paid**	decimal(10,2)			Yes	*NULL*			Change Drop More
8	**credits_earned**	int(11)			Yes	*NULL*			Change Drop More
9	**enrollment_status**	varchar(10)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More

←T→	student_id	student_name	age	GPA	department	year_of_admission	fees_paid	credits_earned	enrollment_status
Edit Copy Delete	1	Eleven	21	3.8	Engineering	2021	10000.00	120	active
Edit Copy Delete	2	Dustin	22	3.9	Science	2020	9000.00	110	active
Edit Copy Delete	3	Will	19	3.4	Business	2022	8500.00	95	active
Edit Copy Delete	4	Mike	23	3.7	Science	2021	9500.00	115	inactive
Edit Copy Delete	5	Max	20	3.5	Engineering	2020	12000.00	130	active
Edit Copy Delete	6	Eddie	22	4.0	Arts	2019	8000.00	140	active
Edit Copy Delete	7	Billy	24	2.9	Engineering	2022	5000.00	60	active
Edit Copy Delete	8	Alexei	25	3.8	Business	2018	7500.00	100	inactive
Edit Copy Delete	9	Steve	21	3.8	Science	2021	10500.00	120	active
Edit Copy Delete	10	Robin	20	3.6	Engineering	2022	11000.00	125	active
Edit Copy Delete	11	Lucas	18	2.7	Engineering	2023	4000.00	50	active
Edit Copy Delete	12	Nancy	23	3.9	Business	2019	9500.00	135	active

Task -1 :

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

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

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

←T→	student_id	student_name	age	GPA	department	year_of_admission	fees_paid	credits_earned	enrollment_status
Edit Copy Delete	1	Eleven	21	3.8	Engineering	2021	10000.00	120	active
Edit Copy Delete	2	Dustin	22	3.9	Science	2020	9000.00	110	active
Edit Copy Delete	4	Mike	23	3.7	Science	2021	9500.00	115	inactive
Edit Copy Delete	6	Eddie	22	4.0	Arts	2019	8000.00	140	active
Edit Copy Delete	8	Alexei	25	3.8	Business	2018	7500.00	100	inactive
Edit Copy Delete	9	Steve	21	3.8	Science	2021	10500.00	120	active
Edit Copy Delete	12	Nancy	23	3.9	Business	2019	9500.00	135	active

Task - 2:

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

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

Extra options

	student_id	student_name	age	GPA	department	year_of_admission	fees_paid	credits_earned	enrollment_status
Edit Copy Delete	5	Max	20	3.5	Engineering	2020	12000.00	130	active
Edit Copy Delete	10	Robin	20	3.6	Engineering	2022	11000.00	125	active
Edit Copy Delete	9	Steve	21	3.8	Science	2021	10500.00	120	active
Edit Copy Delete	1	Eleven	21	3.8	Engineering	2021	10000.00	120	active
Edit Copy Delete	12	Nancy	23	3.9	Business	2019	9500.00	135	active

Task - 3:

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

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

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	student_id	student_name	age	GPA	department	year_of_admission	fees_paid	credits_earned	enrollment_status
Edit Copy Delete	1	Eleven	21	3.8	Engineering	2021	10000.00	120	active
Edit Copy Delete	10	Robin	20	3.6	Engineering	2022	11000.00	125	active

Task - 4:

```
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

Create view

Task - 5:

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

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

department	total_fees_paid	average_GPA

Query results operations

Create view

Discussion:

In this experiment, we successfully demonstrated how to apply various MySQL commands to query and manage data within a student database. We started by creating a "Students" table with key attributes like student ID, name, age, GPA, department, year of admission, fees paid, credits earned, and enrollment status. During the experiment, we carried out several tasks using SQL queries to analyze and extract specific information.

For instance, we used conditional filtering to find students over the age of 20 with a GPA higher than the average, showcasing the use of subqueries and comparison operators. Additionally, we ranked students based on fees paid, using GPA as a tiebreaker to retrieve the top five records, demonstrating the utility of the `ORDER BY` and `LIMIT` clauses.

Furthermore, we employed aggregation functions such as `SUM()` and `AVG()` to calculate the total fees paid and the average GPA for each department, applying filters for departments with more than 10 students. This task emphasized the importance of **grouping** data and utilizing the `HAVING` clause.

These exercises provided valuable hands-on experience with Data Manipulation Language (DML) commands like `SELECT`, `ORDER BY`, and `GROUP BY`, illustrating how SQL can be effectively used for advanced data querying and analysis.

Reference:

[1] W3Schools. (n.d.). SQL Tutorial. Retrieved from https://www.w3schools.com/sql/