Experiment Number : **02**

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

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

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

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

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

---

2.1Objectives:

1. To know about database creation and manipulation, using basic SQL commands.
2. To know how to manage student information, make changes to column names and data types, and perform specific updates and deletions based on conditions.

2.2 Theory:

In a local hosting environment using XAMPP, the database system is centered around MariaDB, an open-source relational database management system included with XAMPP. MariaDB serves as a powerful and reliable database solution that allows developers to create, manage, and interact with databases directly on their local machines. XAMPP also includes phpMyAdmin, a web-based tool that provides an intuitive graphical user interface for managing MariaDB databases. This makes it easy to perform tasks such as creating and deleting databases, running SQL queries, managing tables and records, and handling user permissions without needing extensive knowledge of SQL commands.

2.3 Required Apparatus:

1. Computer ;
2. XAMPP
3. Google chrome browser;

2.4 Solving Task :

The database of given values.

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

Fig. 2.1: Database table

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

Solving Procedure:

i) Selected the table and age was more than 20.
ii) With age, gpa was also added whose gpa were more then average.

Code:

```
SELECT *
FROM student_table
WHERE age > 20
  AND gpa > (SELECT AVG(gpa) FROM student_table);

```

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

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

Code:

```
SELECT *
FROM student_table
ORDER BY fees_paid DESC, gpa DESC
LIMIT 5;

```

Output:

student_id	student_name	age	GPA	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

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

Code:

```
SELECT *
FROM student_table
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

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

Code:

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

```

Output: Empty.

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

Solving Procedure:

iii) Selected the department and add all fees with average GPA.
iv) Table was grouped whose have more than 10 students

Code:

```
SELECT department,
    SUM(fees_paid) AS total_fees_paid,
    AVG(gpa) AS average_gpa
FROM student_table GROUP BY department HAVING COUNT(*) > 10;

```

Output: No Row found.

1.5 Discussion:

we explored a variety of SQL tasks related to managing and querying data from a student_table in a MySQL database using XAMPP. We covered updating column values, renaming tables, and filtering students based on specific criteria such as age, GPA, and enrollment status. Additionally, we delved into advanced querying techniques like finding the top 5 students by fees with GPA as a tiebreaker, listing students from specific departments, and calculating total fees and average GPA per department while filtering out those with fewer than 10 students. These tasks demonstrate how SQL can be used to effectively manipulate and analyze data, making it an essential tool for database management and data analysis.

Reference:

[1]

"W3Schools.com," *W3schools.com*, 2024. https://www.w3schools.com/sql/sql_syntax.asp (accessed Sep. 29, 2024).

[2]

"SQL Commands: DDL, DML, DCL, TCL, DQL - javatpoint," *www.javatpoint.com*, 2024. https://www.javatpoint.com/dbms-sql-command (accessed Sep. 29, 2024).

[3]

Codecademy, "SQL Commands," *Codecademy*, 2024. https://www.codecademy.com/article/sql-commands (accessed Sep. 29, 2024).