Heaven's Light is Our Guide

Rajshahi University of Engineering & Technology

Course Title

Data Base Systems-Sessional

Course No.

ECE-2216

Report

02

Date of Submission: 30.09.2024

Submitted By	**Submitted To**
Ripon Ghosh	Oishi Jyoti
Roll: 2110043	Assistant Professor,
Reg. No.:1097/2021-2022	Dept. of ECE, RUET

Experiment No.: 02

Experiment Name: Create a Database containing following info for 12 students.

1. st_id
2. st_Name
3. age
4. GPA
5. department
6. year_of_admission
7. fees_paid
8. credit_earned
9. enrollment_status

Theory:

In SQL (Structured Query Language), various queries enable users to retrieve specific data from a database. Conditional queries using `WHERE` allow filtering based on multiple conditions. Aggregate functions like `SUM`, `AVG`, and `COUNT` assist in calculations across rows. The `GROUP BY` clause groups rows sharing a property, while `HAVING` is used to apply conditions on aggregated data. Moreover, `ORDER BY` allows sorting results, and `LIMIT` restricts the number of records returned. In this context, these SQL features help answer common analytical questions, such as identifying top performers, active/inactive students, and department-level statistics based on GPA and fees paid.

Tools:

1. XAMPP control Panel
2. MySQL
3. Laptop
4. Internet

Code:

```
CREATE TABLE student_table(
    st_Id INT(10),
    st_Name TEXT(20),
    age INT(20),
    GPA VARCHAR(15),
    department TEXT(15),
    year_of_admission INT(15),
    fees_paid INT(20),
    credits_earn INT(15),
    enrollment_status VARCHAR(20)
    );
```

```
INSERT INTO student_table (st_Id,st_Name,age, GPA, department, year_of_admission, fees_paid, credits_earn, 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', 2021,10500,100,'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:

st_Id	st_Name	age	GPA	department	year_of_admission	fees_paid	credits_earn	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	100	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

Code:

```
SELECT st_Name, age, GPA FROM student_table WHERE age>20 AND GPA>
(SELECT AVG(GPA) FROM student_table);
```

Output:

st_Name	age	GPA
Eleven	21	3.8
Dustin	22	3.9
Mike	23	3.7
Eddie	22	4.0
Steve	21	3.8
Nancy	23	3.9

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

Code:

```
SELECT st_Name,GPA, fees_paid FROM student_table ORDER BY fees_paid DESC, GPA DESC LIMIT 5;
```

Output:

st_Name	GPA ▾ 2	fees_paid ▾ 1
Max	3.5	12000
Robin	3.6	11000
Steve	3.8	10500
Eleven	3.8	10000
Nancy	3.9	9500

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

Code:

```
SELECT st_Name, GPA, department, year_of_admission FROM student_table
WHERE department = 'Engineering'
AND GPA > 3.5
AND year_of_admission > 2020;
```

Output:

st_Name	GPA	department	year_of_admission
Eleven	3.8	Engineering	2021
Robin	3.6	Engineering	2022

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

Code:

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

Output:

st_Name	fees_paid	enrollment_status

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

Code:

```
24
25 SELECT department,
26        SUM(fees_paid) AS total_fees_paid,
27        AVG(gpa) AS average_gpa,
28        COUNT(*) AS student_count
29 FROM student_table
30 GROUP BY department
31 HAVING COUNT(*) > 10;
```

Output:

department	total_fees_paid	average_gpa	student_count

Discussion:

The dataset presented highlights common attributes like student age, GPA, department, fees paid, and enrollment status. SQL queries address real-world tasks, such as finding students based on age and GPA or analyzing departmental data. For instance, identifying inactive students with unpaid fees or departments with high GPA students leverages conditional filtering. Aggregation functions like `SUM` and `AVG`, combined with grouping and having clauses, efficiently generate department-wise insights, aiding decision-making in academic or financial contexts. SQL's power to handle such data-driven queries showcases its utility in database management, improving resource allocation and performance analysis.

References:

[1] W3Schools, "SQL CREATE TABLE Statement," [Online]. Available: https://www.w3schools.com/sql/sql_create_table.asp. [Accessed: 24-Sep-2024].

[2] GeeksforGeeks, "SQL | DDL, DQL, DML, DCL, TCL Commands," [Online]. Available: https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/. [Accessed: 24-Sep-2024].