"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	: 24 September 2024
Submission Date	: 1 October, 2024

Submitted To-
Oishi Jyoti
Assistant Professor,
ECE, RUET

Submitted By-
Mst .Samia Aktar Sathi
Roll Number: 2110003
Reg. Number: 1057
Session: 2021-2022

Experiment Number: 02

Experiment Name: Database Management with MySQL: Query Op-erations on Students Table

Objectives:

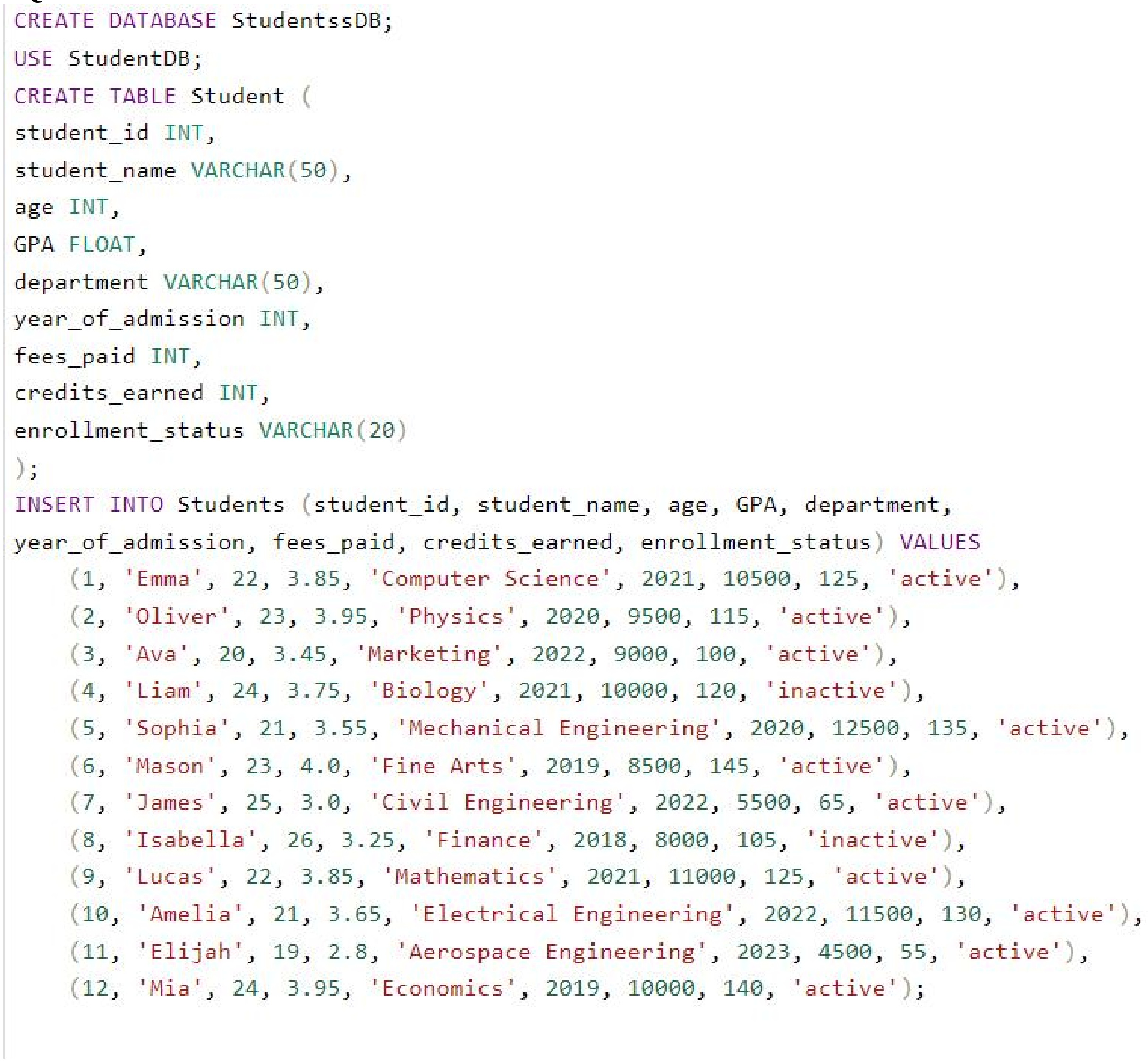
To execute various SQL queries to analyze student data based on multiple criteria and extract meaningful insights from the information.

Theory:

This experiment involves querying a student database to retrieve and analyze information based on specific parameters such as age, GPA, fees, and enrollment status. SQL commands will be used to filter and retrieve data, while aggregate functions will be applied to derive comprehensive insights for further analysis.

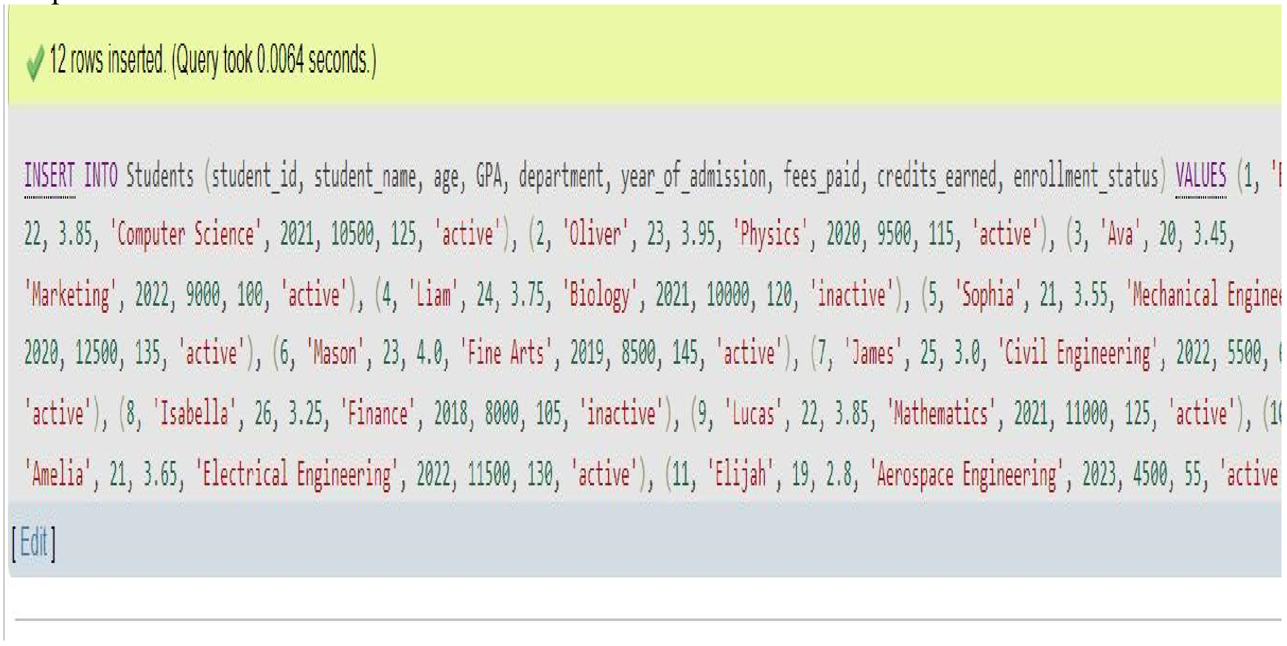
Creating Database and Table:

SQL Commands:



```
CREATE DATABASE StudentssDB;
USE StudentDB;
CREATE TABLE Student (
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, 'Emma', 22, 3.85, 'Computer Science', 2021, 10500, 125, 'active'),
    (2, 'Oliver', 23, 3.95, 'Physics', 2020, 9500, 115, 'active'),
    (3, 'Ava', 20, 3.45, 'Marketing', 2022, 9000, 100, 'active'),
    (4, 'Liam', 24, 3.75, 'Biology', 2021, 10000, 120, 'inactive'),
    (5, 'Sophia', 21, 3.55, 'Mechanical Engineering', 2020, 12500, 135, 'active'),
    (6, 'Mason', 23, 4.0, 'Fine Arts', 2019, 8500, 145, 'active'),
    (7, 'James', 25, 3.0, 'Civil Engineering', 2022, 5500, 65, 'active'),
    (8, 'Isabella', 26, 3.25, 'Finance', 2018, 8000, 105, 'inactive'),
    (9, 'Lucas', 22, 3.85, 'Mathematics', 2021, 11000, 125, 'active'),
    (10, 'Amelia', 21, 3.65, 'Electrical Engineering', 2022, 11500, 130, 'active'),
    (11, 'Elijah', 19, 2.8, 'Aerospace Engineering', 2023, 4500, 55, 'active'),
    (12, 'Mia', 24, 3.95, 'Economics', 2019, 10000, 140, 'active');
```

Output:



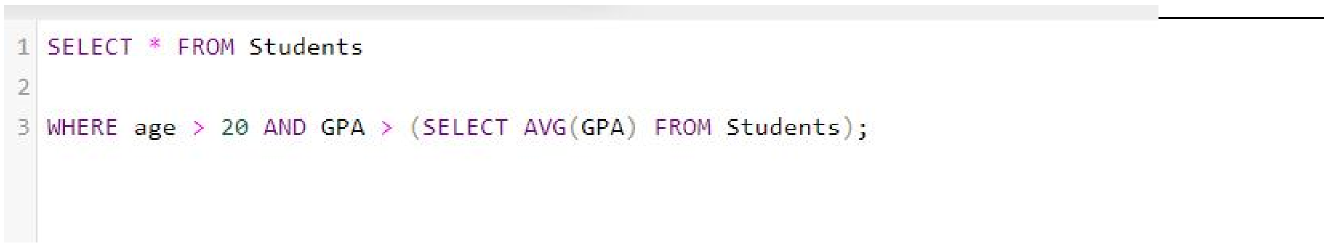
```
✓ 12 rows inserted. (Query took 0.0064 seconds.)

INSERT INTO Students (student_id, student_name, age, GPA, department, year_of_admission, fees_paid, credits_earned, enrollment_status) VALUES (1, '
22, 3.85, 'Computer Science', 2021, 10500, 125, 'active'), (2, 'Oliver', 23, 3.95, 'Physics', 2020, 9500, 115, 'active'), (3, 'Ava', 20, 3.45,
'Marketing', 2022, 9000, 100, 'active'), (4, 'Liam', 24, 3.75, 'Biology', 2021, 10000, 120, 'inactive'), (5, 'Sophia', 21, 3.55, 'Mechanical Enginee
2020, 12500, 135, 'active'), (6, 'Mason', 23, 4.0, 'Fine Arts', 2019, 8500, 145, 'active'), (7, 'James', 25, 3.0, 'Civil Engineering', 2022, 5500,
'active'), (8, 'Isabella', 26, 3.25, 'Finance', 2018, 8000, 105, 'inactive'), (9, 'Lucas', 22, 3.85, 'Mathematics', 2021, 11000, 125, 'active'), (1
'Amelia', 21, 3.65, 'Electrical Engineering', 2022, 11500, 130, 'active'), (11, 'Elijah', 19, 2.8, 'Aerospace Engineering', 2023, 4500, 55, 'active
```

Figure 1: Database and table creation output

Problem Statements:

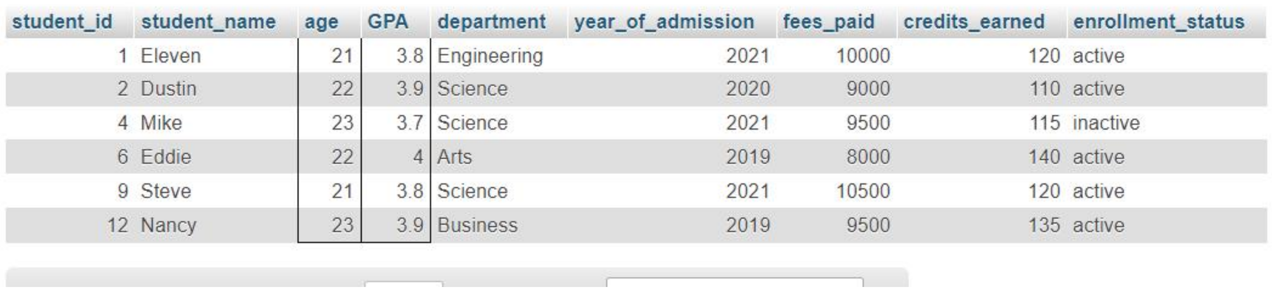
1. Find students older than 20 with a GPA above the average GPA SQL Command:



```sql
SELECT * FROM Students

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

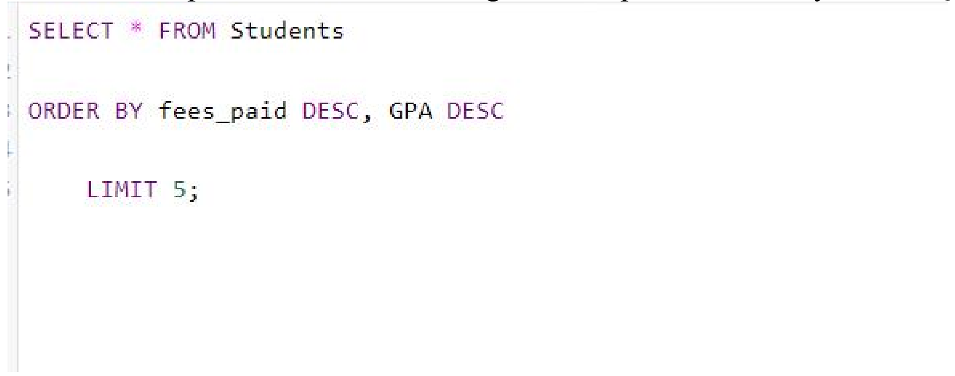
Output:



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:

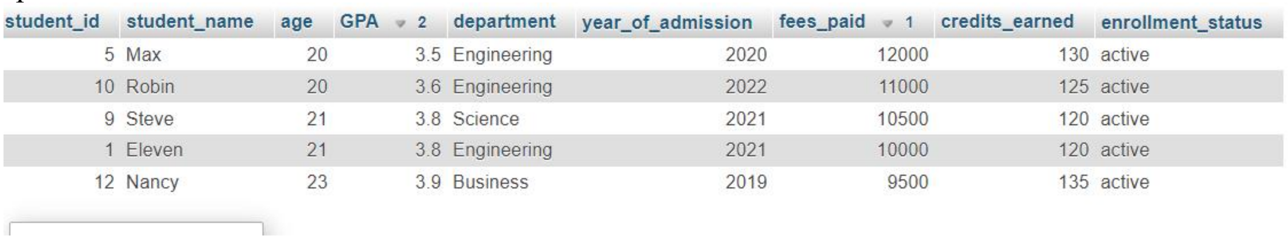


```
SELECT * FROM Students

ORDER BY fees_paid DESC, GPA DESC

    LIMIT 5;
```

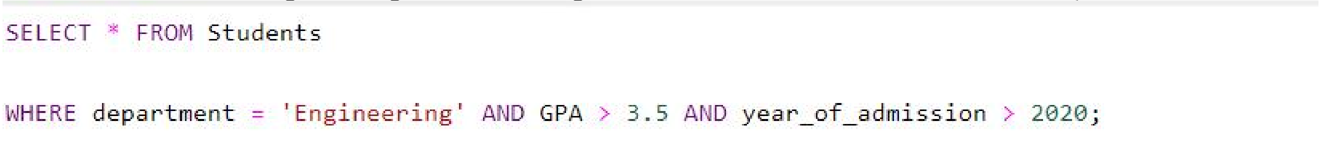
Output:



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

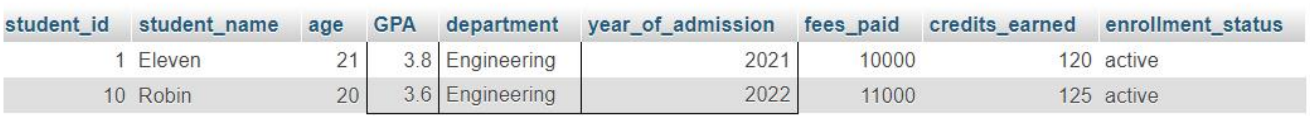
3. List students in Engineering with a GPA greater than 3.5, enrolled after 2020 SQL Command:



```
SELECT * FROM Students

WHERE department = 'Engineering' AND GPA > 3.5 AND year_of_admission > 2020;
```

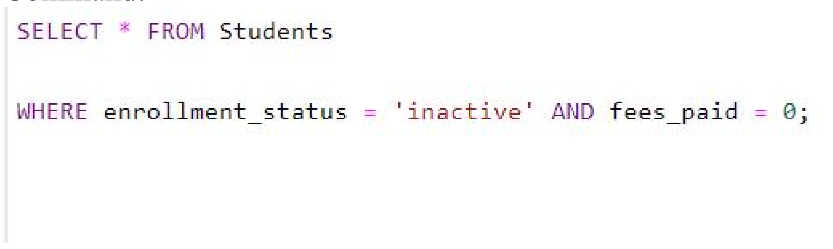
Output:



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

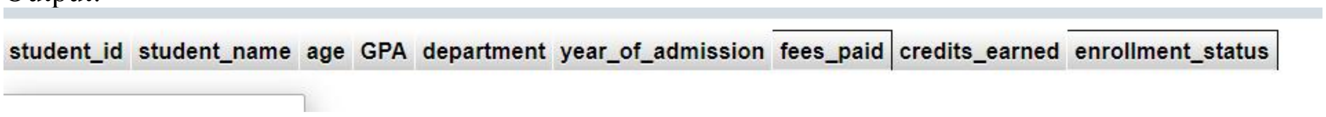
4. Find inactive students with no fees paid SQL Command:



```
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:

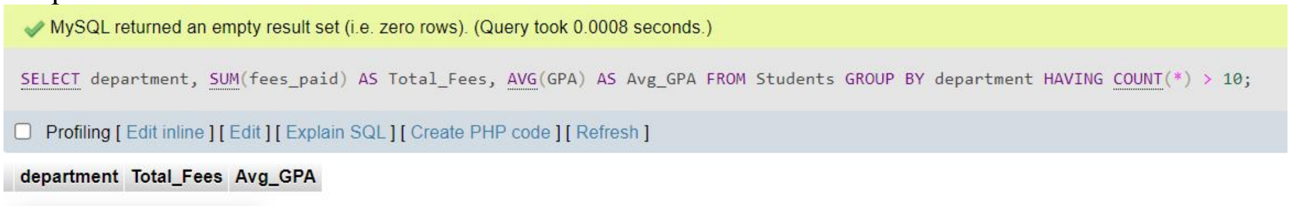
```
SELECT department, SUM(fees_paid) AS Total_Fees, AVG(GPA) AS Avg_GPA

FROM Students

GROUP BY department

    HAVING COUNT(*) > 10;
```

Output:



```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0008 seconds.)

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

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

department	Total_Fees	Avg_GPA

Figure 6: Total fees and average GPA by department

Discussion:

This experiment showcased the effective use of SQL queries to analyze student data. We applied various queries to filter students based on criteria such as age, GPA, fees paid, and enrollment status. Additionally, we computed aggregate values to gain insights into the financial contributions of different departments. These operations are crucial for managing educational databases and making data-driven decisions.