“Heaven’s Light is Our Guide

Rajshahi University of Engineering & Technology
Rajshahi, Bangladesh

Department of Electrical & Computer Engineering
(ECE-21)

Course Code: ECE 2216
Course Title: Database Systems Sessional
Experiment No: 02
Date of Submission: 30/09/2024

Submitted To:

Oishi Jyoti
Assistant Professor
Rajshahi University of Engineering & Technology

Submitted By:

Md. Shahin Alam
Roll: 2110018
ECE-21 Series

Experiment No. 02

Experiment Name: Managing Student Database and Conditional Data Logging in MySQL.

Theory:

In today's database systems, efficient administration of structured data is essential for a number of useful applications. In a XAMPP environment, this exercise investigates basic database operations such as inserting, removing, updating, and conditionally altering records in a relational MySQL database. It arranges and manipulates student data in a structured table format by using SQL (Structured Query Language) instructions.

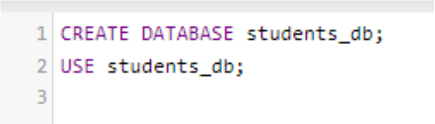
Problem Statement:

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

Software Used:

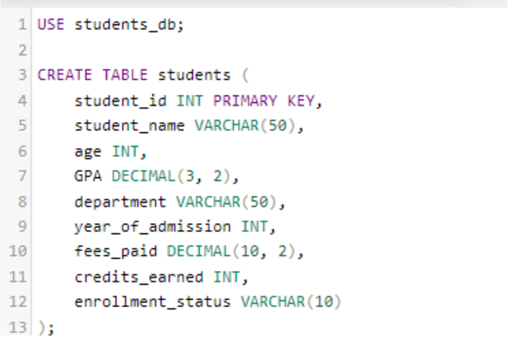
1. Xampp Control Panel
2. MySQL

Task: Creating database and table



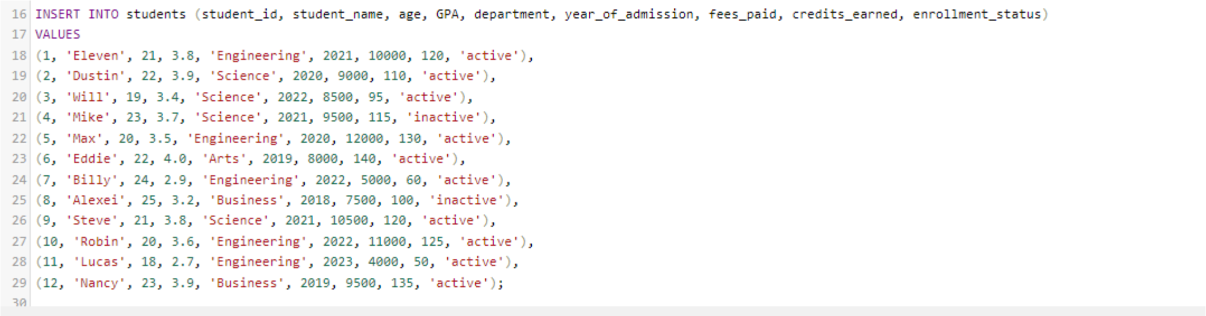
```sql
CREATE DATABASE students_db;
USE students_db;

```



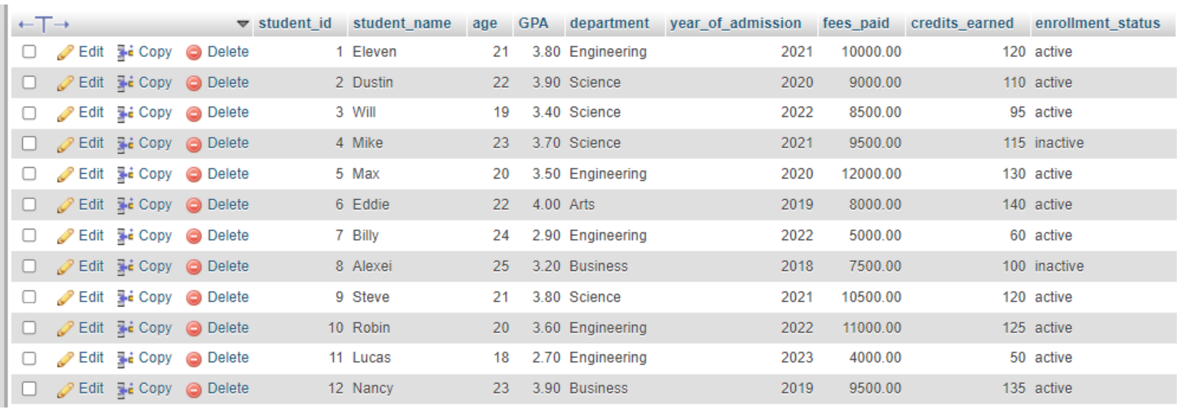
```sql
USE students_db;

CREATE TABLE students (
    student_id INT PRIMARY KEY,
    student_name VARCHAR(50),
    age INT,
    GPA DECIMAL(3, 2),
    department VARCHAR(50),
    year_of_admission INT,
    fees_paid DECIMAL(10, 2),
    credits_earned INT,
    enrollment_status VARCHAR(10)
);
```



```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, 'Science', 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');
```

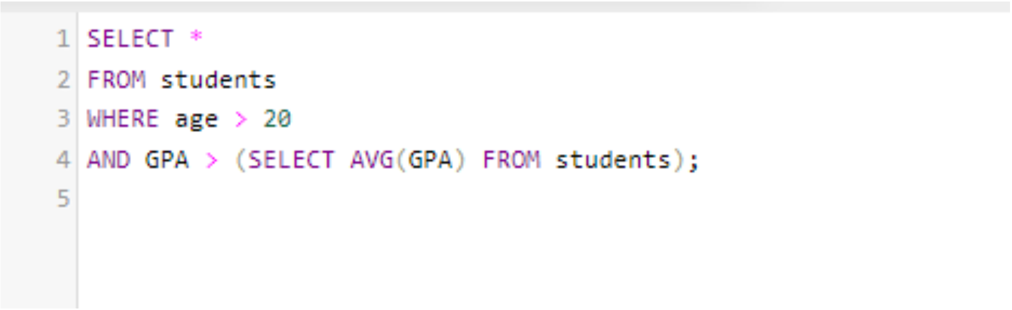
Output:



←T→			▼	student_id	student_name	age	GPA	department	year_of_admission	fees_paid	credits_earned	enrollment_status
□	Edit	Copy	Delete	1	Eleven	21	3.80	Engineering	2021	10000.00	120	active
□	Edit	Copy	Delete	2	Dustin	22	3.90	Science	2020	9000.00	110	active
□	Edit	Copy	Delete	3	Will	19	3.40	Science	2022	8500.00	95	active
□	Edit	Copy	Delete	4	Mike	23	3.70	Science	2021	9500.00	115	inactive
□	Edit	Copy	Delete	5	Max	20	3.50	Engineering	2020	12000.00	130	active
□	Edit	Copy	Delete	6	Eddie	22	4.00	Arts	2019	8000.00	140	active
□	Edit	Copy	Delete	7	Billy	24	2.90	Engineering	2022	5000.00	60	active
□	Edit	Copy	Delete	8	Alexei	25	3.20	Business	2018	7500.00	100	inactive
□	Edit	Copy	Delete	9	Steve	21	3.80	Science	2021	10500.00	120	active
□	Edit	Copy	Delete	10	Robin	20	3.60	Engineering	2022	11000.00	125	active
□	Edit	Copy	Delete	11	Lucas	18	2.70	Engineering	2023	4000.00	50	active
□	Edit	Copy	Delete	12	Nancy	23	3.90	Business	2019	9500.00	135	active

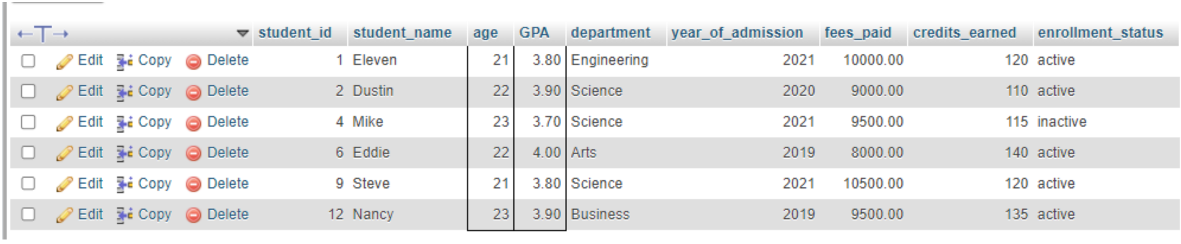
Task 1: Find students older than 20 with GPA above the average GPA of all students

Code:



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

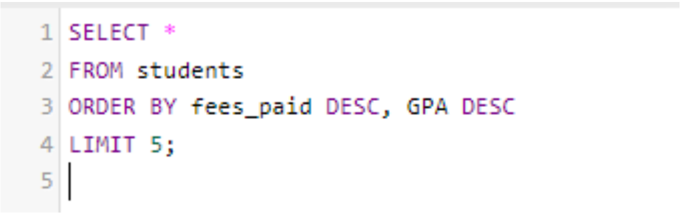
Output:



←T→ ▼	student_id	student_name	age	GPA	department	year_of_admission	fees_paid	credits_earned	enrollment_status
□ Edit Copy Delete	1	Eleven	21	3.80	Engineering	2021	10000.00	120	active
□ Edit Copy Delete	2	Dustin	22	3.90	Science	2020	9000.00	110	active
□ Edit Copy Delete	4	Mike	23	3.70	Science	2021	9500.00	115	inactive
□ Edit Copy Delete	6	Eddie	22	4.00	Arts	2019	8000.00	140	active
□ Edit Copy Delete	9	Steve	21	3.80	Science	2021	10500.00	120	active
□ Edit Copy Delete	12	Nancy	23	3.90	Business	2019	9500.00	135	active

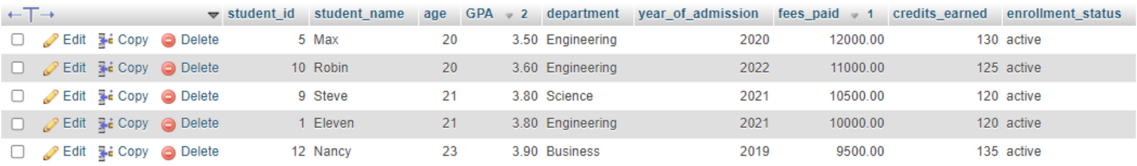
Task 2: Find the top 5 students with the highest fees paid, ordered by GPA (as a tiebreaker)

Code:



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

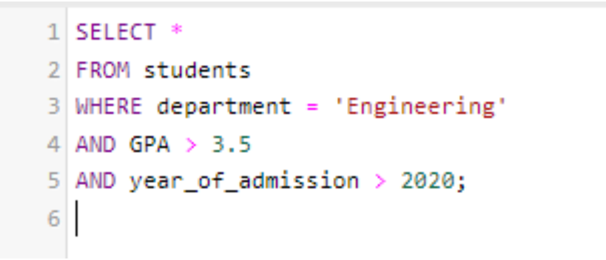
Output:



←T→ ▼	student_id	student_name	age	GPA ▾ 2	department	year_of_admission	fees_paid ▾ 1	credits_earned	enrollment_status
□ Edit Copy Delete	5	Max	20	3.50	Engineering	2020	12000.00	130	active
□ Edit Copy Delete	10	Robin	20	3.60	Engineering	2022	11000.00	125	active
□ Edit Copy Delete	9	Steve	21	3.80	Science	2021	10500.00	120	active
□ Edit Copy Delete	1	Eleven	21	3.80	Engineering	2021	10000.00	120	active
□ Edit Copy Delete	12	Nancy	23	3.90	Business	2019	9500.00	135	active

Task 3: List students from the "Engineering" department with a GPA greater than 3.5 and enrolled after 2020

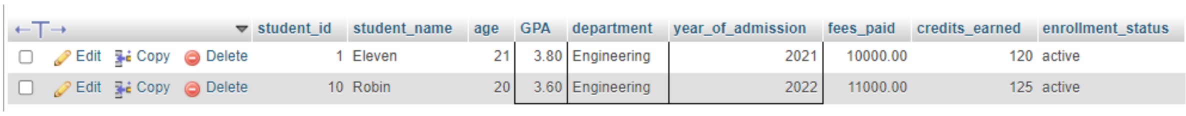
Code:



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

```

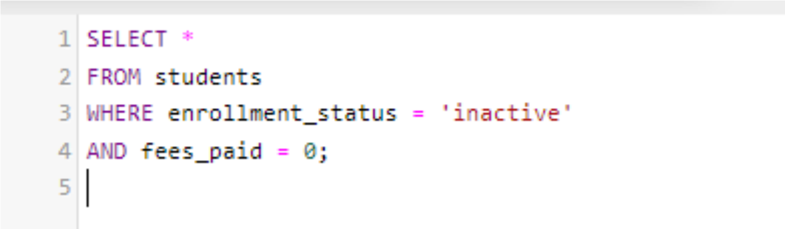
Output:



	student_id	student_name	age	GPA	department	year_of_admission	fees_paid	credits_earned	enrollment_status
□ Edit Copy Delete	1	Eleven	21	3.80	Engineering	2021	10000.00	120	active
□ Edit Copy Delete	10	Robin	20	3.60	Engineering	2022	11000.00	125	active

Task 4: Find students who are not active and have not paid any fees (fees_paid = 0)

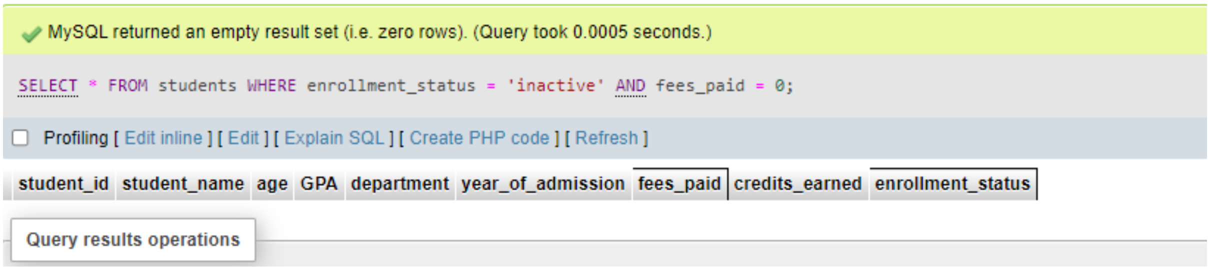
Code:



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

```

Output:



✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0005 seconds.)

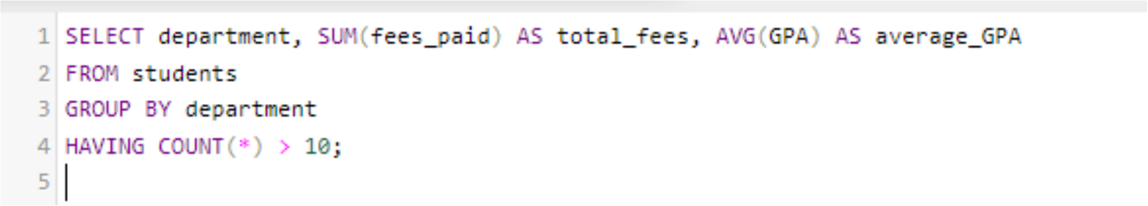
```sql
SELECT * FROM students WHERE enrollment_status = 'inactive' AND fees_paid = 0;
```

□ 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

Task 5: Calculate the total fees paid and average GPA for each department with more than 10 students

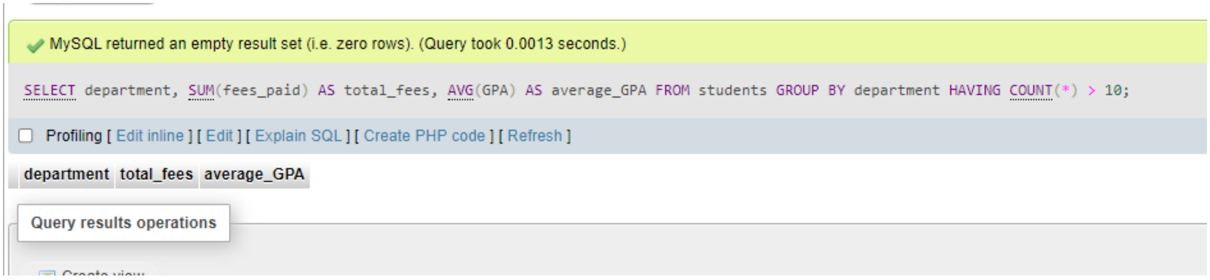
Code:



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

```

Output:



✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0013 seconds.)

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

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

department	total_fees	average_GPA

Discussion:

This experiment used MySQL with XAMPP to examine simple database operations. First, we made a table called "students" and a database called "student_db." We renamed a column "favorite_subject" to "major" in order to demonstrate how to handle database modifications. To maintain the data's relevance, we removed the records of students who received less than 30 marks. Additionally, we created a new column named "log" and entered data in it according to the semester. Overall, this experiment made the database more accurate and valuable by assisting us in comprehending critical activities like adding, editing, and managing data in a straightforward manner.To input text, tap or click this link.