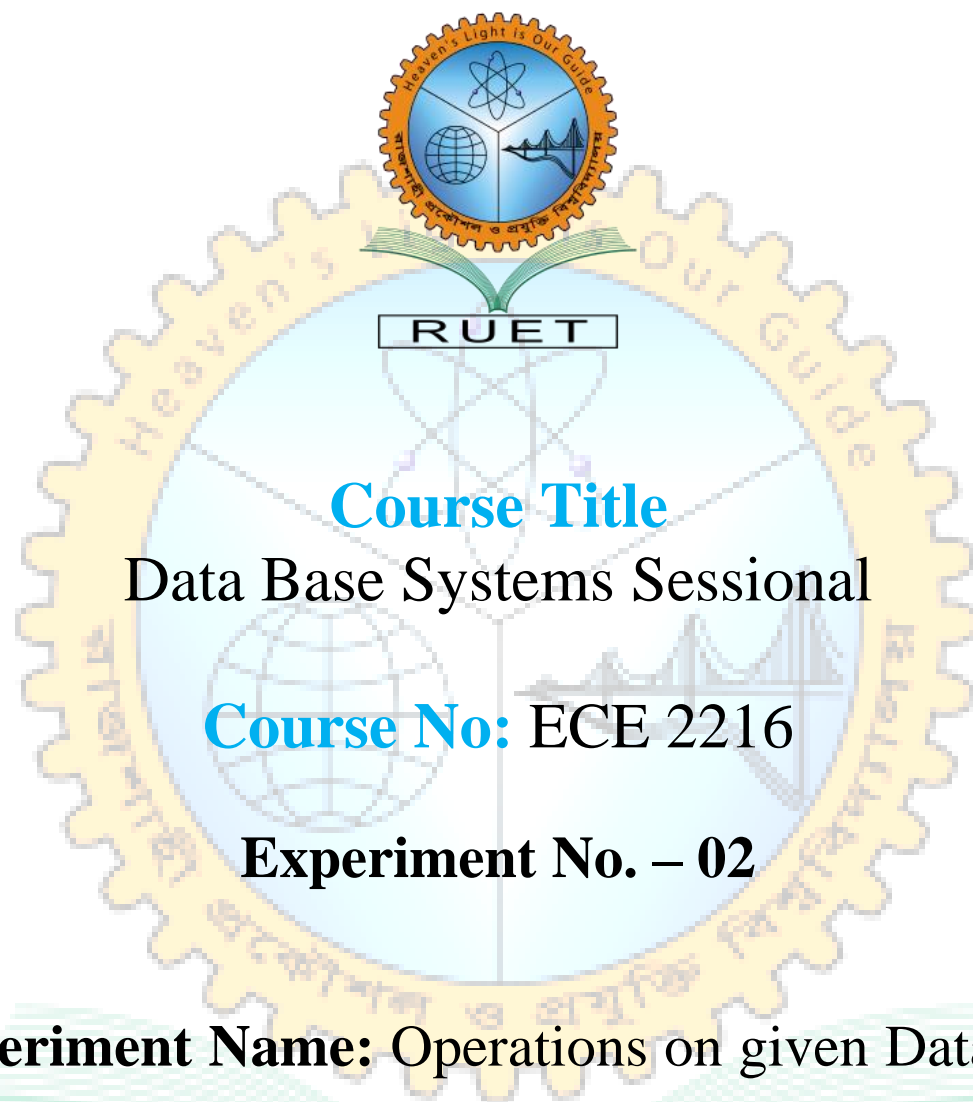


Havens Light is Our Guide
Rajshahi University of Engineering & Technology



Experiment Name: Operations on given Database Table.

Date of submission: 01/10/2024

<u>Submitted to:</u>	<u>Submitted by:</u>
Oishi Jyoti Assistant Professor Department of ECE, RUET	Muhammad Nisar Uddin Roll: 2110027 Reg. No: 1081/2021-2022 ECE-21 series, RUET

Experiment No: 02

Students Table

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:

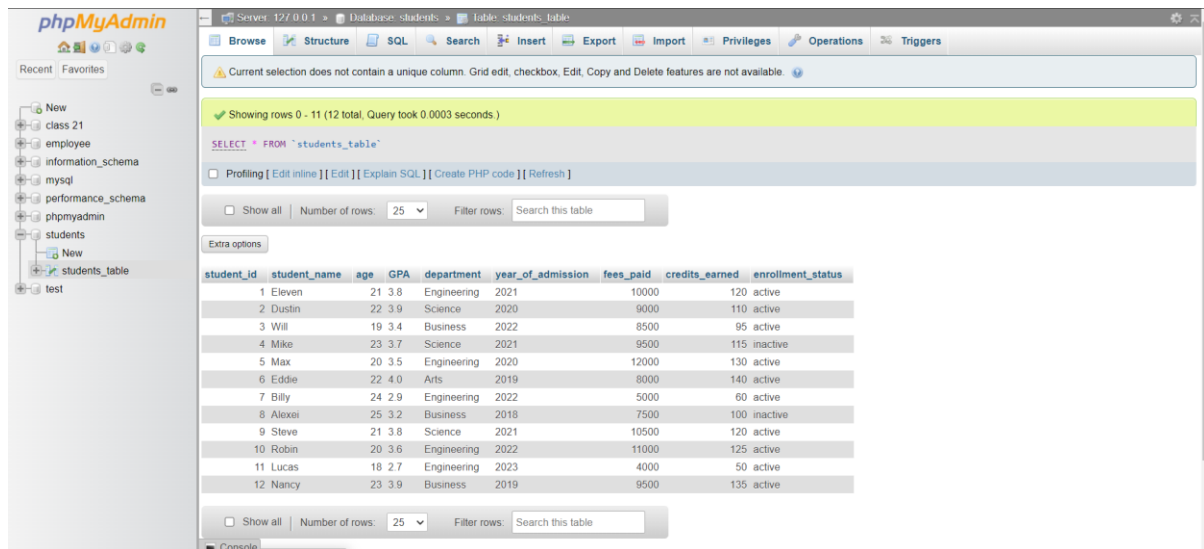
1. Find students who are older than 20 and have a GPA above the average GPA of all students
2. Find the top 5 students with the highest fees paid, ordered by GPA (in descending order) as a tiebreaker
3. List students who belong to the "Engineering" department, have a GPA greater than 3.5, and are enrolled 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

Objective:

- i. To learn about the database system using XAMP.
- ii. To know about SQL and NoSQL database system.
- iii. To differentiate between DML and DDL in SQL.
- iv. To learn about the query language to create tables and inputs in a database system.
- v. To apply queries to solve real life scenarios.

Query & Output:

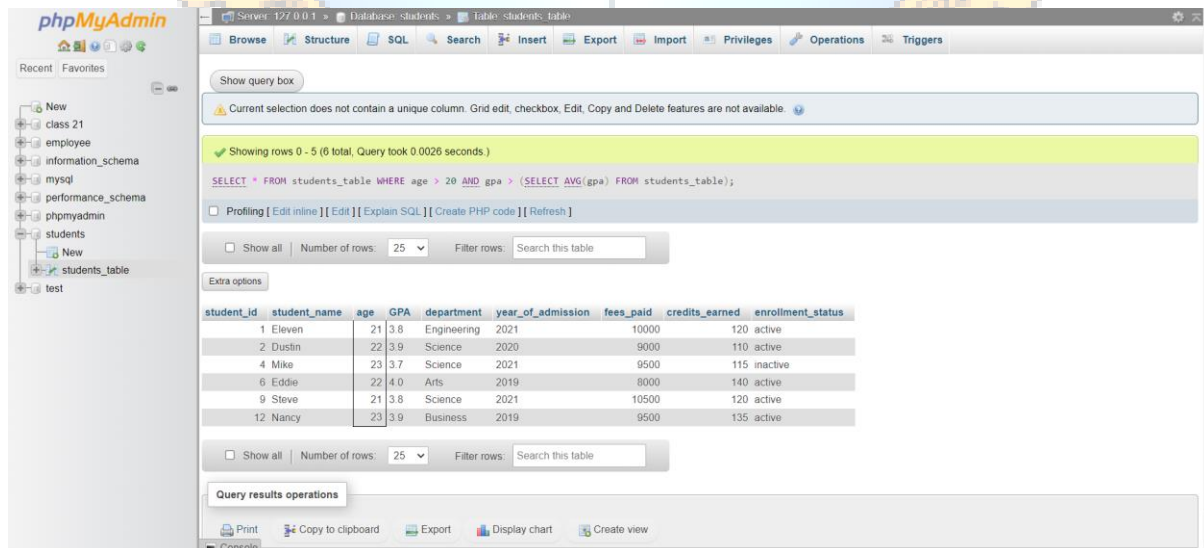
Given Table:



The screenshot shows the phpMyAdmin interface for the 'students' table. The table structure is as follows:

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.



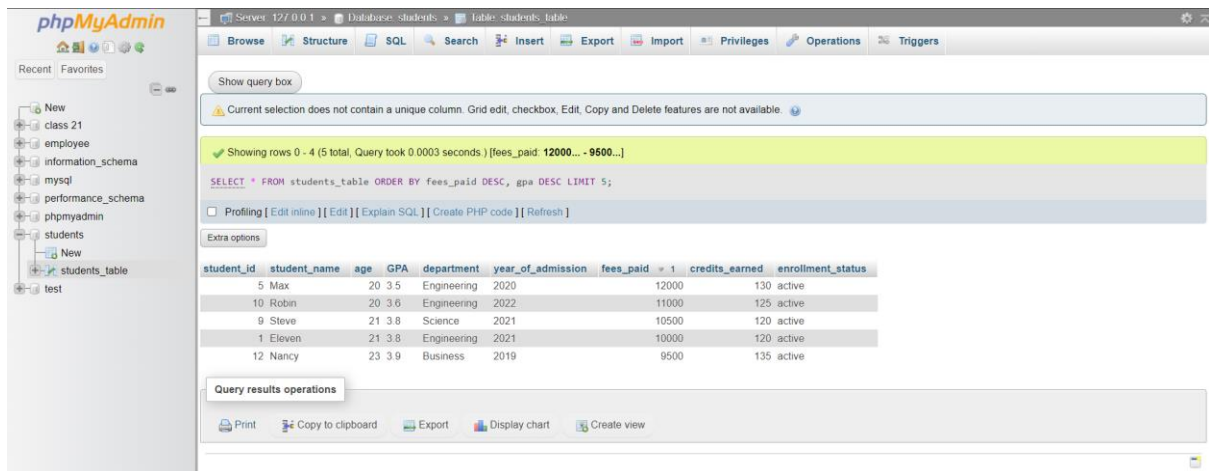
The screenshot shows the phpMyAdmin interface with the following SQL query entered in the query box:

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

The query results show 6 rows of data:

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	4.0	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

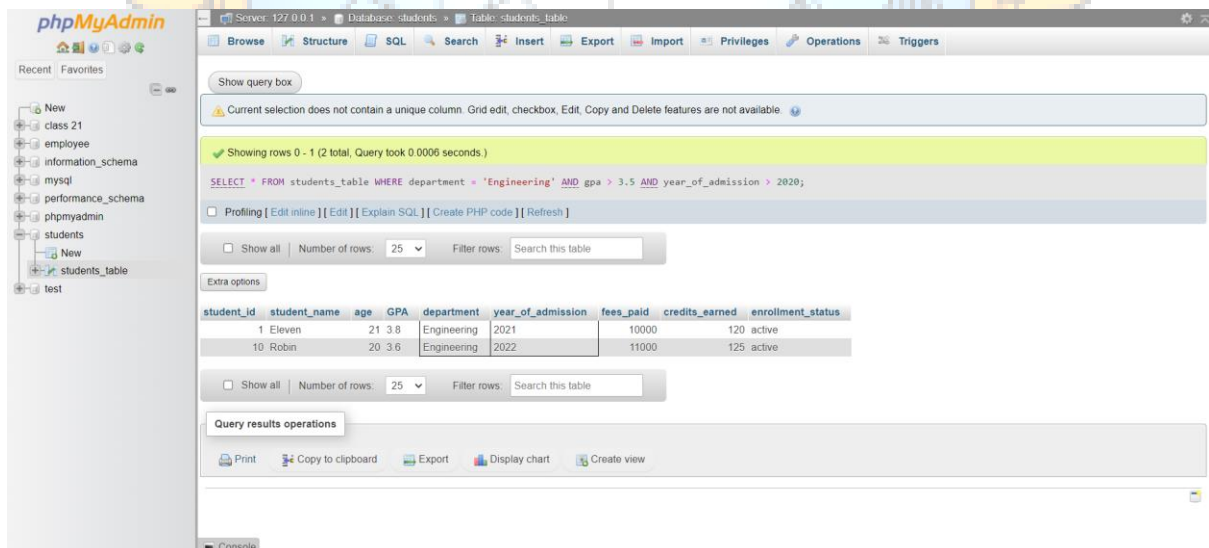
Task 2: Find the top 5 students with the highest fees paid, ordered by GPA (in descending order) as a tiebreaker.



The screenshot shows the phpMyAdmin interface with the 'students' database selected. A SQL query is executed, displaying the top 5 students with the highest fees paid, ordered by GPA in descending order as a tiebreaker. The query results are shown in a table format.

student_id	student_name	age	GPA	department	year_of_admission	fees_paid	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

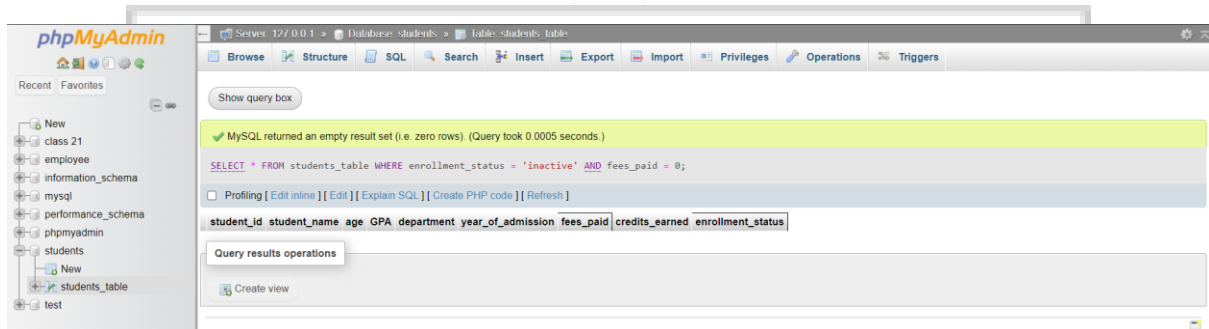
Task 3: List students who belong to the "Engineering" department, have a GPA greater than 3.5, and are enrolled after 2020.



The screenshot shows the phpMyAdmin interface with the 'students' database selected. A SQL query is executed, displaying students who belong to the "Engineering" department, have a GPA greater than 3.5, and are enrolled after 2020. The query results are shown in a table format.

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).



The screenshot shows the phpMyAdmin interface with the 'students' database selected. A SQL query is executed, displaying students who are not active (enrollment_status = 'inactive') and have not paid any fees (fees_paid = 0). The query results are shown in a table format.

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, but only for departments with more than 2 students.

phpMyAdmin

Server: 127.0.0.1 » Database: students » Table: students_table

Recent Favorites

- New
- class 21
- employee
- information_schema
- mysql
- performance_schema
- phpmyadmin
- students
- New
- students_table
- test

Show query box

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 2 (3 total, Query took 0.0146 seconds)

```
SELECT department, SUM(fees_paid) AS total_fees, AVG(gpa) AS average_gpa, COUNT(student_id) AS total_students FROM students_table GROUP BY department HAVING COUNT(student_id) > 2;
```

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

Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

department	total_fees	average_gpa	total_students
Business	25500	3.5	3
Engineering	42000	3.3	5
Science	29000	3.7999999999999994	3

Show all | Number of rows: 25 | Filter rows: Search this table

Query results operations

Print Copy to clipboard Export Display chart Create view

Console

