

# **ASSIGNMENT**

**STUDENT INFORMATION SYSTEM  
(SQL IMPLEMENTATION)**

JASWANTH KUMAR

**BY**

**JASWANTH KUMAR S**

**BATCH -4  
(PYTHON FOUNDATION)**

# Task 1. Database Design:

\*\*\* I HAVE TAKEN A FEW SCREENSHOT TO SHOWCASE ALTOGETHER ONLY FOR THIS TASK [TASK\_1] AND TO LOOK UPON THE INSERTED VALUE , I'VE INCLUDED THE SQL SCRIPTS IN GTIHUB REPOSITORY\*\*\*

1. Create the database named "SISDB"
2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships. a. Students b. Courses c. Enrollments d. Teacher e. Payments
4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.

The screenshot shows the MySQL Workbench interface. In the central workspace, there is a code editor window titled 'table creation' containing the following SQL script:

```
1 • CREATE DATABASE SISDB;
2 • USE SISDB;
3
4 • CREATE TABLE Students
5   (
6     student_id INT AUTO_INCREMENT PRIMARY KEY,
7     first_name VARCHAR(100) NOT NULL,
8     last_name VARCHAR(100) NOT NULL,
9     date_of_birth DATE NOT NULL,
10    email VARCHAR(100) UNIQUE NOT NULL,
11    phone_number VARCHAR(15) UNIQUE NOT NULL);
12 • CREATE TABLE Teachers
13   (
14     teacher_id INT AUTO_INCREMENT PRIMARY KEY,
```

Below the code editor, the 'Output' pane displays the results of the executed queries:

#	Time	Action	Message	Duration / Fetch
90	09:58:10	CREATE DATABASE SISDB	1 row(s) affected	0.016 sec
91	09:58:10	USE SISDB	0 row(s) affected	0.000 sec
92	09:58:10	CREATE TABLE Students (student_id INT AUTO_INCREMENT PRIMARY KEY, fir...	0 row(s) affected	0.109 sec
93	09:58:10	CREATE TABLE Teachers (teacher_id INT AUTO_INCREMENT PRIMARY KEY, ...)	0 row(s) affected	0.125 sec
94	09:58:10	CREATE TABLE Courses (course_id INT AUTO_INCREMENT PRIMARY KEY, co...	0 row(s) affected	0.141 sec
95	09:58:10	CREATE TABLE Payments (payment_id INT AUTO_INCREMENT PRIMARY KEY, ...)	0 row(s) affected	0.078 sec
96	09:58:10	CREATE TABLE Enrollments (enrollment_id INT AUTO_INCREMENT PRIMARY KEY...)	0 row(s) affected	0.109 sec

## STUDENTS:

The screenshot shows the MySQL Workbench interface with the 'hexa (sisdb)' database selected. In the Navigator pane, the 'Tables' section is expanded, showing 'courses', 'enrollments', 'payments', 'students', and 'teachers'. The 'students' table is selected. In the main query editor, a SQL script is running, inserting 10 rows of student data into the 'Students' table. The 'Result Grid' pane displays the inserted data, which includes columns: student\_id, first\_name, last\_name, date\_of\_birth, email, and phone\_number. The data is as follows:

student_id	first_name	last_name	date_of_birth	email	phone_number
1	Arun	Kumar	1998-05-12	arun.kumar@example.com	9876543210
2	Karthik	Raj	1997-11-20	karthik.raj@example.com	9823456710
3	Priya	Subramani	1999-02-25	priya.subramani@example.com	9812345678
4	Deepak	Ravi	1996-07-18	deepak.ravi@example.com	987654321
5	Meena	Lakshmanan	2000-10-05	meena.lakshmanan@example.com	9834567890
6	Ramesh	Venkatesh	1995-04-30	ramesh.venkatesh@example.com	9845671234
7	Sundar	Babu	1998-09-14	sundar.babu@example.com	9871234567
8	Lakshmi	Iyer	1997-03-08	lakshmi.iyer@example.com	9823456789
9	Bala	Murugan	1999-12-21	bala.murugan@example.com	9815674321
10	Janani	Sridhar	2001-06-11	janani.sridhar@example.com	9834123456

## TEACHERS:

The screenshot shows the MySQL Workbench interface with the 'hexa (sisdb)' database selected. In the Navigator pane, the 'Tables' section is expanded, showing 'courses', 'enrollments', 'payments', 'students', and 'teachers'. The 'teachers' table is selected. In the main query editor, a SQL script is running, selecting data from both the 'students' and 'teachers' tables. The 'Result Grid' pane displays the selected data, which includes columns: teacher\_id, first\_name, last\_name, and email. The data is as follows:

teacher_id	first_name	last_name	email
1	Ganesh	Narayanan	ganesh.narayanan@example.com
2	Uma	Ravi	uma.ravi@example.com
3	Harish	Selvam	harish.selvam@example.com
4	Revathi	Krishnan	revathi.krishnan@example.com

## COURSES:

The screenshot shows the MySQL Workbench interface with the 'sisdb' schema selected. In the SQL editor tab, several queries are run to retrieve data from the 'courses' table. The results are displayed in a grid format.

```
50 (7, 450.00, '2024-04-10'),  
51 (8, 550.00, '2024-04-15'),  
52 (9, 650.00, '2024-04-20'),  
53 (10, 750.00, '2024-04-25');  
54  
55 • SELECT * FROM students;  
56 • SELECT * FROM teachers;  
57 • SELECT * FROM courses;  
58 • SELECT * FROM students;  
59
```

course_id	course_name	credits	teacher_id
3	Database Management Systems	5	3
4	Computer Networks	3	4
5	Software Engineering Principles	4	1
6	Artificial Intelligence	5	2
7	Data Structures & Algorithms	4	3
8	Operating Systems	4	4
9	Cyber Security Basics	3	1
10	Machine Learning	5	2
• NULLS	NULLS	NULLS	NULLS

## PAYMENTS:

The screenshot shows the MySQL Workbench interface with the 'sisdb' schema selected. In the SQL editor tab, queries are run to retrieve data from the 'payments' table. The results are displayed in a grid format.

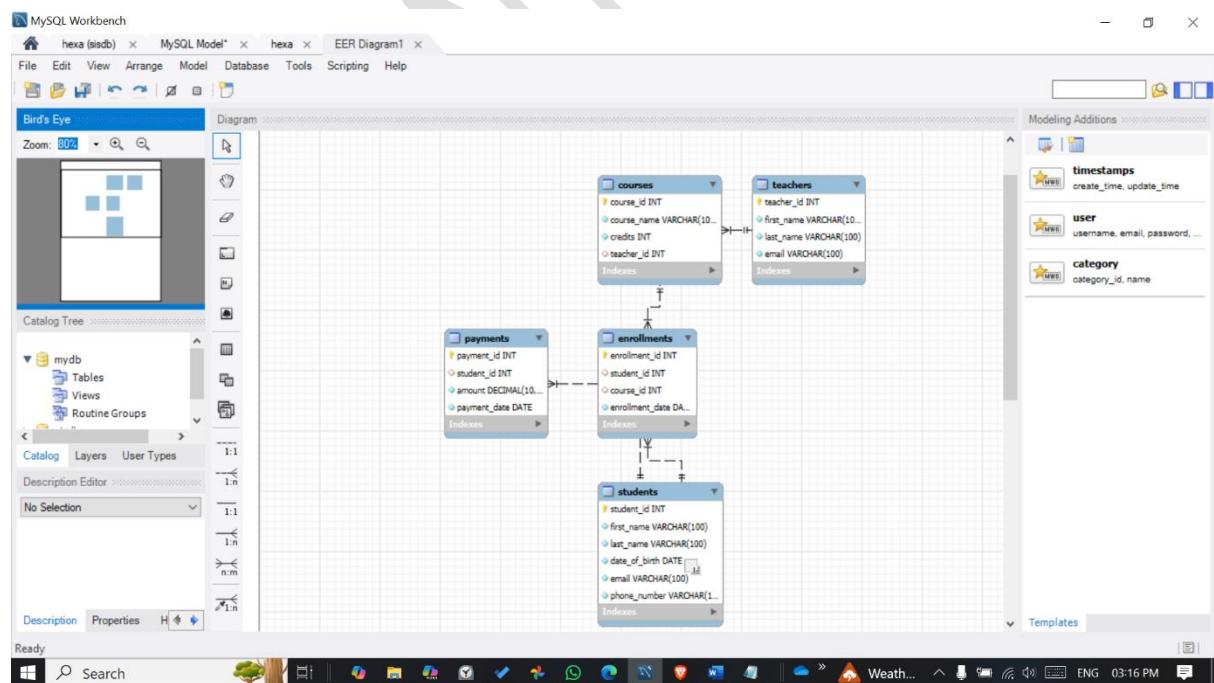
```
51 (8, 550.00, '2024-04-15'),  
52 (9, 650.00, '2024-04-20'),  
53 (10, 750.00, '2024-04-25');  
54  
55 • SELECT * FROM students;  
56 • SELECT * FROM teachers;  
57 • SELECT * FROM courses;  
58 • SELECT * FROM enrollments;  
59 • SELECT * FROM payments;  
60
```

payment_id	student_id	amount	payment_date
1	1	500.00	2024-03-10
2	2	700.00	2024-03-15
3	3	300.00	2024-03-20
4	4	400.00	2024-03-25
5	5	600.00	2024-04-01
6	6	800.00	2024-04-05
7	7	450.00	2024-04-10
8	8	550.00	2024-04-15
9	9	650.00	2024-04-20
10	10	750.00	2024-04-25

## ENROLMENTS:

The screenshot shows the MySQL Workbench interface. In the top navigation bar, tabs include 'hexa (sisdb)', 'MySQL Model\*', 'hexa', and 'EER Diagram1'. The main area has a 'table creation' tab selected. A code editor window displays several SQL SELECT statements. Below it is a 'Result Grid' showing data from an 'enrollments' table. The grid has columns: enrollment\_id, student\_id, course\_id, and enrollment\_date. The data shows 13 rows of enrollment records. On the left, the Navigator pane shows the 'SCHEMAS' section with 'payments' and 'students' selected. The 'students' schema contains columns: student\_id, first\_name, last\_name, date\_of\_birth, email, and phone\_number. The bottom status bar shows the system tray and taskbar.

### 3. Create an ERD (Entity Relationship Diagram) for the database.

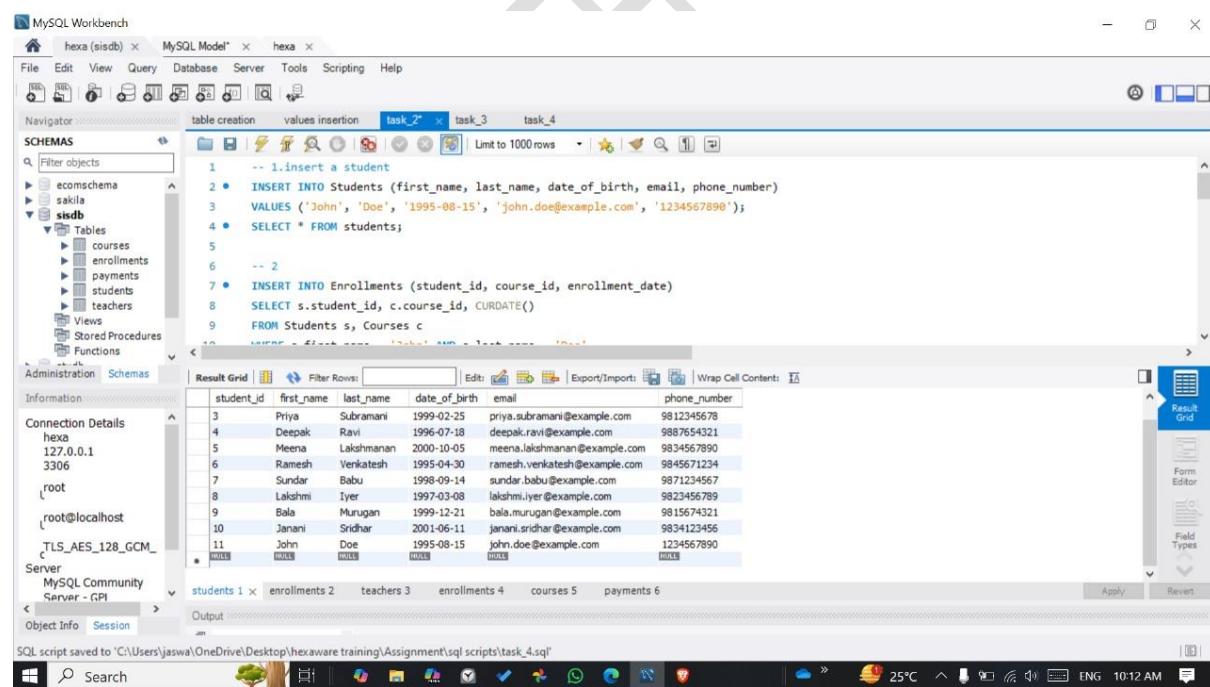


# Tasks 2: Select, Where, Between, AND, LIKE:

## 2.1

1. Write an SQL query to insert a new student into the "Students" table with the following details:

- a. First Name: John
- b. Last Name: Doe
- c. Date of Birth: 1995-08-15
- d. Email: [john.doe@example.com](mailto:john.doe@example.com)
- e. Phone Number: 1234567890



The screenshot shows the MySQL Workbench interface. In the top navigation bar, the database 'hexa (sisdb)' is selected. The main area displays an SQL editor window with the following code:

```
-- 1.insert a student
INSERT INTO Students (first_name, last_name, date_of_birth, email, phone_number)
VALUES ('John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');

-- 2
INSERT INTO Enrollments (student_id, course_id, enrollment_date)
SELECT s.student_id, c.course_id, CURDATE()
FROM Students s, Courses c
```

Below the SQL editor is a Result Grid showing the data for the 'Students' table. The grid has columns: student\_id, first\_name, last\_name, date\_of\_birth, email, and phone\_number. The data includes rows for students like Priya, Deepak, Meena, Ramesh, Sundar, Lakshmi, Bala, Janani, and John.

## 2.2

Write an SQL query to enroll a student in a course. Choose an existing student and course and insert a record into the "Enrollments" table with the enrollment date.

The screenshot shows the MySQL Workbench interface with a query editor window. The code entered is:

```
4 •  SELECT * FROM students;
5
6 -- 2
7 •  INSERT INTO Enrollments (student_id, course_id, enrollment_date)
8   SELECT s.student_id, c.course_id, CURDATE()
9   FROM Students s, Courses c
10 WHERE s.first_name = 'John' AND s.last_name = 'Doe'
11 AND c.course_name = 'Data Science Fundamentals';
12
13 •  SELECT * FROM enrollments;
```

The Result Grid shows the following data:

	enrollment_id	student_id	course_id	enrollment_date
3	3	3	3	2024-03-03
4	4	4	4	2024-03-04
5	5	5	5	2024-03-05
6	6	6	6	2024-03-06
7	7	7	7	2024-03-07
8	8	8	8	2024-03-08
9	9	9	9	2024-03-09
10	10	10	10	2024-03-10
11	11	11	1	2025-03-21
	NULL	NULL	NULL	NULL

## 2.3

Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.

The screenshot shows the MySQL Workbench interface with a query editor window. The code entered is:

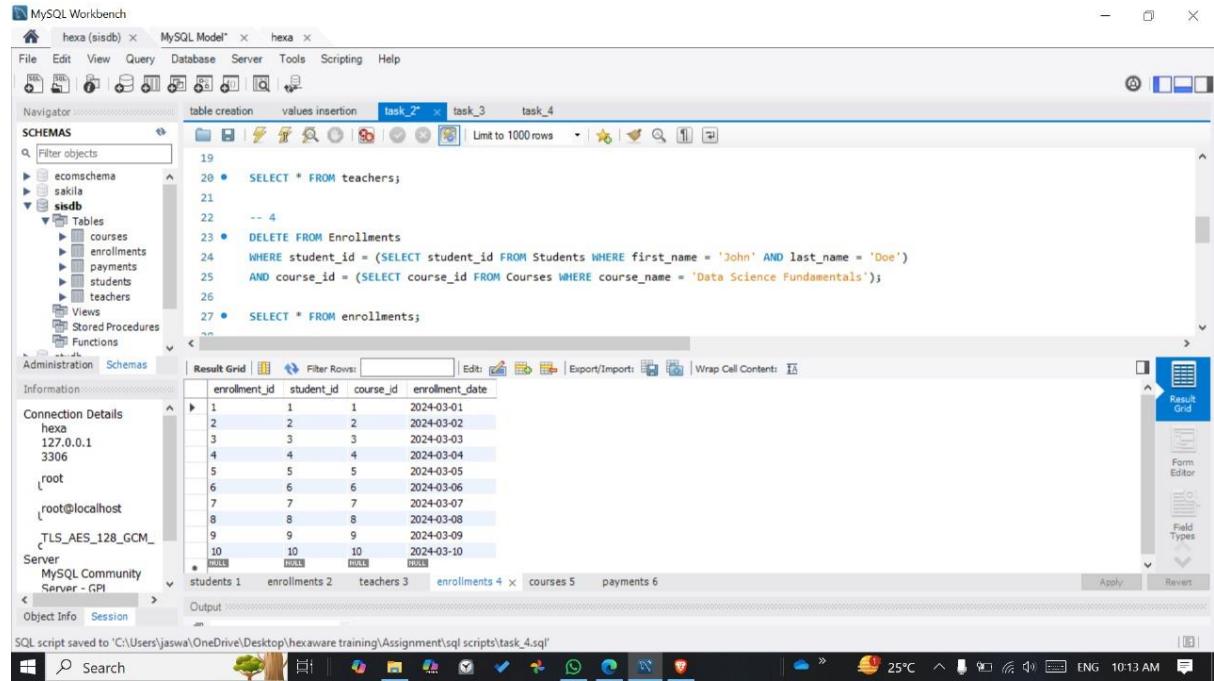
```
13 •  SELECT * FROM enrollments;
14
15 -- 3
16 •  UPDATE Teachers
17   SET email = 'uma.updated@example.com'
18   WHERE teacher_id = 2;
19
20 •  SELECT * FROM teachers;
21
```

The Result Grid shows the following data:

	teacher_id	first_name	last_name	email
1	1	Ganesh	Narayanan	ganesh.narayanan@example.com
2	2	Uma	Ravi	uma.updated@example.com
3	3	Harish	Selvam	harish.selvam@example.com
4	4	Revathi	Krishnan	revathi.krishnan@example.com
	NULL	NULL	NULL	NULL

## 2.4

Write an SQL query to delete a specific enrollment record from the "Enrollments" table. Select an enrollment record based on the student and course.



The screenshot shows the MySQL Workbench interface with the following details:

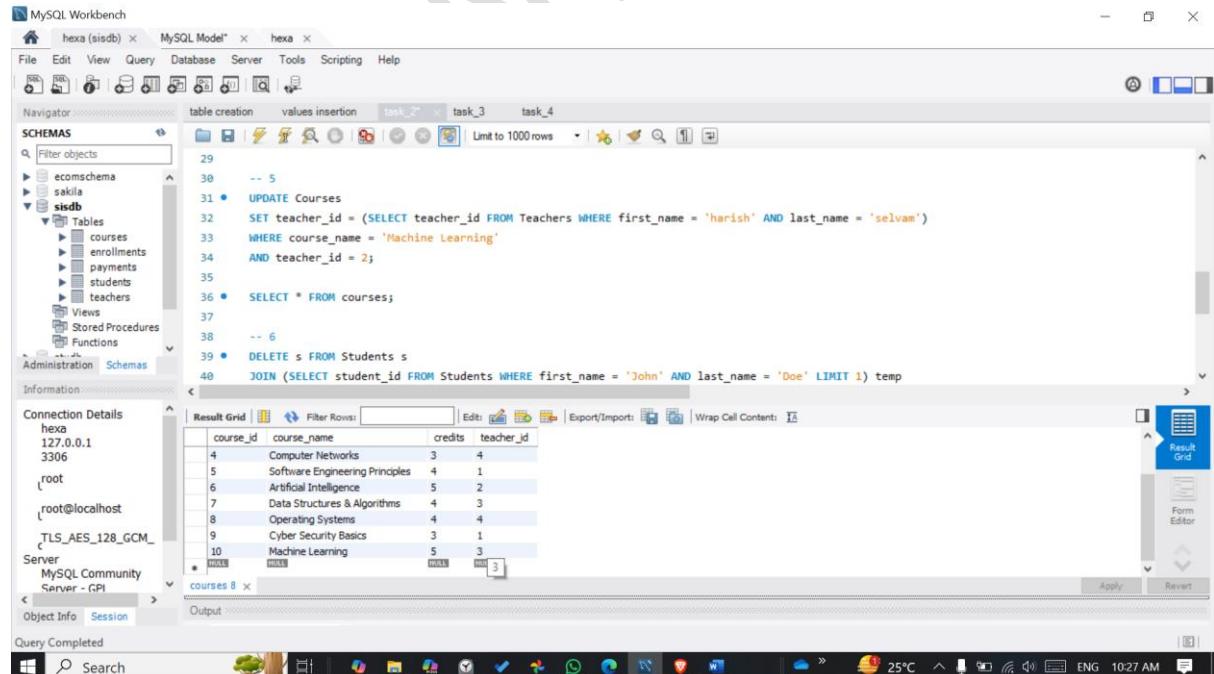
- File:** hexa (sisdb) × MySQL Model\* × hexa ×
- Navigator:** table creation values insertion task\_2\* × task\_3 task\_4
- SCHEMAS:** ecomschema, sakila, sisdb (selected), Views, Stored Procedures, Functions
- Tables:** courses, enrollments, payments, students, teachers
- SQL Editor:** Contains the following SQL code:

```
19
20 • SELECT * FROM teachers;
21
22 -- 4
23 • DELETE FROM Enrollments
24 WHERE student_id = (SELECT student_id FROM Students WHERE first_name = 'John' AND last_name = 'Doe')
25 AND course_id = (SELECT course_id FROM Courses WHERE course_name = 'Data Science Fundamentals');
26
27 • SELECT * FROM enrollments;
```
- Result Grid:** Shows a table with columns: enrollment\_id, student\_id, course\_id, enrollment\_date. The data is as follows:

enrollment_id	student_id	course_id	enrollment_date
1	1	1	2024-03-01
2	2	2	2024-03-02
3	3	3	2024-03-03
4	4	4	2024-03-04
5	5	5	2024-03-05
6	6	6	2024-03-06
7	7	7	2024-03-07
8	8	8	2024-03-08
9	9	9	2024-03-09
10	10	10	2024-03-10
- Information:** Connection Details, Server, MySQL Community Server - GPL
- Object Info:** Session
- Output:** SQL script saved to 'C:\Users\jaswa\Desktop\hexaware training\Assignment\sql scripts\task\_4.sql'
- System Bar:** Search, Task View, Start button, Icons, Date/Time: 25°C ENG 10:13 AM

## TASK 2.5

Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.



The screenshot shows the MySQL Workbench interface with the following details:

- File:** hexa (sisdb) × MySQL Model\* × hexa ×
- Navigator:** table creation values insertion task\_2\* × task\_3 task\_4
- SCHEMAS:** ecomschema, sakila, sisdb (selected), Views, Stored Procedures, Functions
- Tables:** courses, enrollments, payments, students, teachers
- SQL Editor:** Contains the following SQL code:

```
29
30 -- 5
31 • UPDATE Courses
32 SET teacher_id = (SELECT teacher_id FROM Teachers WHERE first_name = 'Harish' AND last_name = 'Selvam')
33 WHERE course_name = 'Machine Learning'
34 AND teacher_id = 2;
35
36 • SELECT * FROM courses;
37
38 -- 6
39 • DELETE s FROM Students s
40 JOIN (SELECT student_id FROM Students WHERE first_name = 'John' AND last_name = 'Doe' LIMIT 1) temp
```
- Result Grid:** Shows a table with columns: course\_id, course\_name, credits, teacher\_id. The data is as follows:

course_id	course_name	credits	teacher_id
4	Computer Networks	3	4
5	Software Engineering Principles	4	1
6	Artificial Intelligence	5	2
7	Data Structures & Algorithms	4	3
8	Operating Systems	4	4
9	Cyber Security Basics	3	1
10	Machine Learning	5	3
- Information:** Connection Details, Server, MySQL Community Server - GPL
- Object Info:** Session
- Output:** Query Completed
- System Bar:** Search, Task View, Start button, Icons, Date/Time: 25°C ENG 10:27 AM

## Task 2.6

Delete a specific student from the "Students" table and remove all their enrollment records from the "Enrollments" table. Be sure to maintain referential integrity.

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
35
36 • SELECT * FROM courses;
37
38 --- 6
39 • DELETE FROM Enrollments
40 WHERE student_id = (SELECT student_id FROM Students WHERE first_name = 'bala' AND last_name = 'murugan' LIMIT 1);
41
42 --- 7
43 • UPDATE Payments
44 SET amount = 950.00
```

The results grid shows the following data for the Enrollments table:

enrollment_id	student_id	course_id	enrollment_date
6	6	6	2024-03-06
7	7	7	2024-03-07
8	8	8	2024-03-08
10	10	10	2024-03-10
NULL	NULL	NULL	NULL

The output pane shows the execution of the queries:

#	Time	Action	Message	Duration / Fetch
128	10:32:13	SELECT * FROM students LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
129	10:33:02	SELECT * FROM enrollments LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec

## Task 2.7

Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
42 --- 7
43 • UPDATE Payments
44 SET amount = 950.00
45 WHERE payment_date BETWEEN '2024-04-01' AND '2024-04-30'
46 AND student_id = (SELECT student_id FROM Students WHERE first_name = 'sundar' AND last_name = 'babu' LIMIT 1);
47
48 • SELECT * FROM payments;
49 • SELECT * FROM enrollments;
```

The results grid shows the following data for the Payments table:

payment_id	student_id	amount	payment_date
5	5	600.00	2024-04-01
6	6	800.00	2024-04-05
7	7	950.00	2024-04-10
8	8	550.00	2024-04-15
9	9	650.00	2024-04-20
10	10	750.00	2024-04-25
NULL	NULL	NULL	NULL

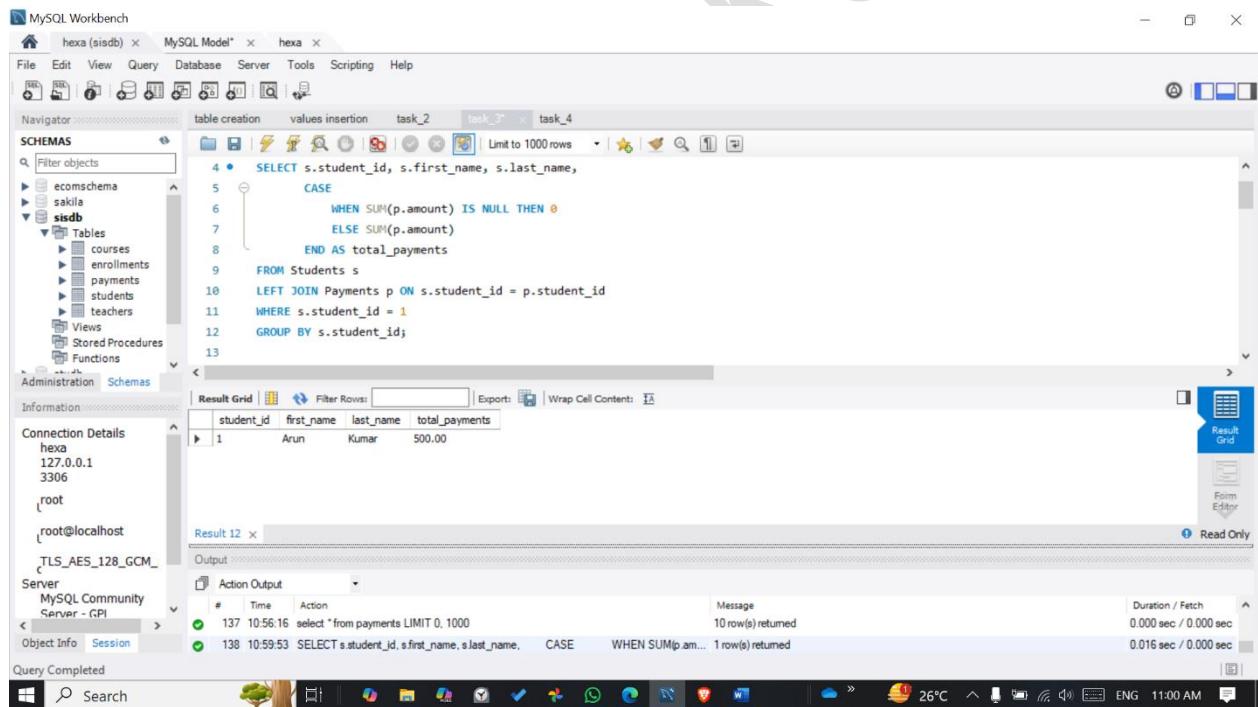
The output pane shows the execution of the queries:

#	Time	Action	Message	Duration / Fetch
134	10:37:27	UPDATE Payments SET amount = 950.00 WHERE payment_date BETWEEN '2024-04-01' AND '2024-04-30' AND student_id = (SELECT student_id FROM Students WHERE first_name = 'sundar' AND last_name = 'babu' LIMIT 1);	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.031 sec
135	10:37:31	SELECT * FROM payments LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

# Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:

## 3.1

Write an SQL query to calculate the total payments made by a specific student. You will need to join the "Payments" table with the "Students" table based on the student's ID.



The screenshot shows the MySQL Workbench interface with a query editor and results pane. The query is:

```
SELECT s.student_id, s.first_name, s.last_name,
       CASE
           WHEN SUM(p.amount) IS NULL THEN 0
           ELSE SUM(p.amount)
       END AS total_payments
  FROM Students s
 LEFT JOIN Payments p ON s.student_id = p.student_id
 WHERE s.student_id = 1
 GROUP BY s.student_id;
```

The results grid shows one row:

student_id	first_name	last_name	total_payments
1	Arun	Kumar	500.00

The status bar at the bottom indicates the query completed successfully.

### 3.2

Write an SQL query to retrieve a list of courses along with the count of students enrolled in each course. Use a JOIN operation between the "Courses" table and the "Enrollments" table.

The screenshot shows the MySQL Workbench interface with a query editor window. The code is:

```
13
14 -- 2
15 • SELECT c.course_name, COUNT(e.student_id) AS student_count
16   FROM Courses c
17   LEFT JOIN Enrollments e ON c.course_id = e.course_id
18   GROUP BY c.course_id, c.course_name
19   ORDER BY student_count DESC;
20
21 -- 3
22 • SELECT s.first_name, s.last_name
```

The result grid shows:

course_name	student_count
Artificial Intelligence	1
Data Structures & Algorithms	1
Operating Systems	1
Machine Learning	1
Cyber Security Basics	0

The output pane shows two log entries:

#	Time	Action	Message	Duration / Fetch
138	10:59:53	SELECT s.student_id, s.first_name, s.last_name.	CASE WHEN SUM(ip.am... 1 row(s) returned	0.016 sec / 0.000 sec
139	11:03:34	SELECT c.course_name, COUNT(e.student_id) AS student_count FROM Courses c L...	10 row(s) returned	0.000 sec / 0.000 sec

### 3.3

Write an SQL query to find the names of students who have not enrolled in any course. Use a LEFT JOIN between the "Students" table and the "Enrollments" table to identify students without enrollments.

The screenshot shows the MySQL Workbench interface with a query editor window. The code is:

```
19 ORDER BY student_count DESC;
20
21 -- 3
22 • SELECT s.student_id, s.first_name, s.last_name
23   FROM Students s
24   LEFT JOIN Enrollments e ON s.student_id = e.student_id
25   WHERE e.enrollment_id IS NULL;
26
27
28 -- 4
```

The result grid shows:

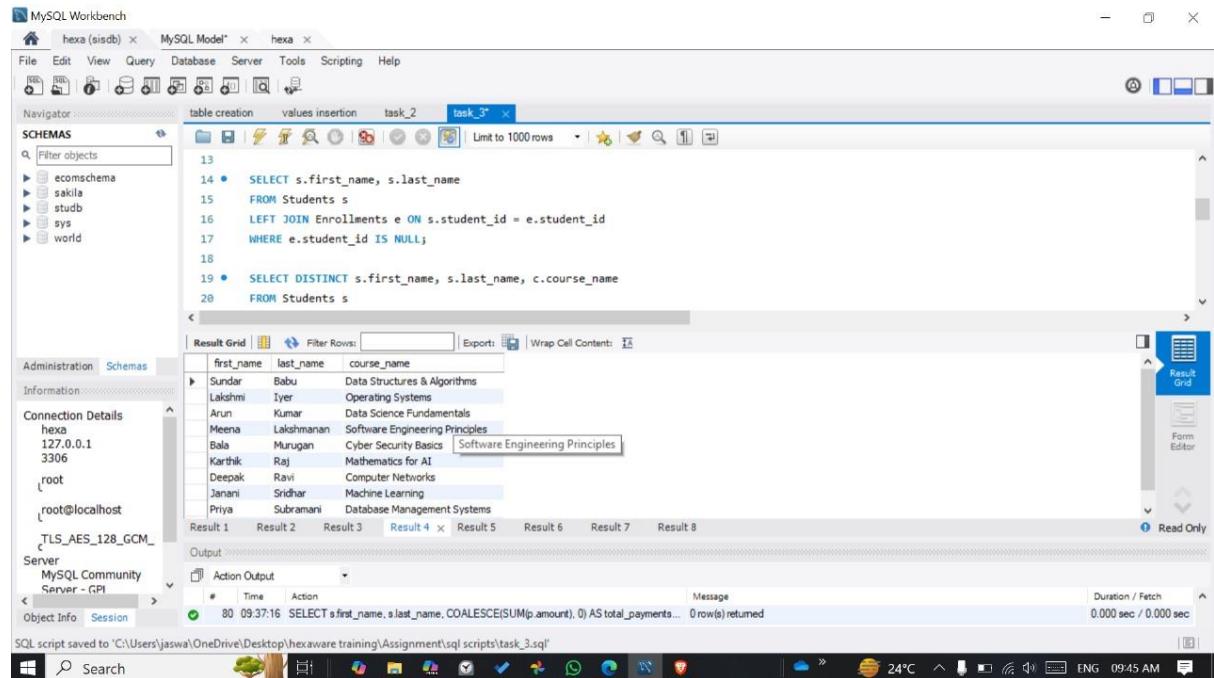
student_id	first_name	last_name
9	Bala	Murugan
		Murugan

The output pane shows two log entries:

#	Time	Action	Message	Duration / Fetch
139	11:03:34	SELECT c.course_name, COUNT(e.student_id) AS student_count FROM Courses c L...	10 row(s) returned	0.000 sec / 0.000 sec
140	11:14:18	SELECT s.student_id, s.first_name, s.last_name FROM Students s LEFT JOIN Enrolm...	1 row(s) returned	0.000 sec / 0.000 sec

### 3.4

Write an SQL query to retrieve the first name, last name of students, and the names of the courses they are enrolled in. Use JOIN operations between the "Students" table and the "Enrollments" and "Courses" tables.



The screenshot shows the MySQL Workbench interface with a query editor window titled 'task\_3'. The code entered is:

```
13
14 • SELECT s.first_name, s.last_name
15   FROM Students s
16   LEFT JOIN Enrollments e ON s.student_id = e.student_id
17   WHERE e.student_id IS NULL;
18
19 • SELECT DISTINCT s.first_name, s.last_name, c.course_name
20   FROM Students s
```

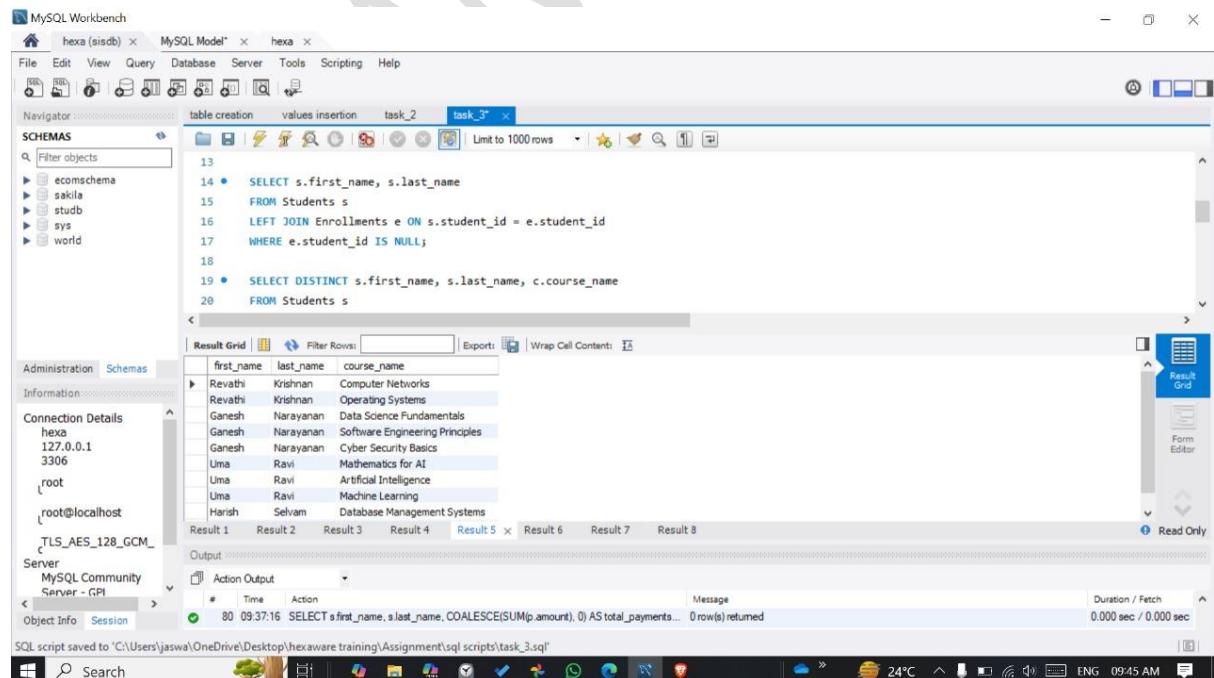
The result grid displays the following data:

first_name	last_name	course_name
Sundar	Babu	Data Structures & Algorithms
Lakshmi	Iyer	Operating Systems
Arun	Kumar	Data Science Fundamentals
Meena	Lakshmanan	Software Engineering Principles
Bala	Murugan	Cyber Security Basics
Karthik	Raj	Mathematics for AI
Deepak	Ravi	Computer Networks
Janani	Sridhar	Machine Learning
Priya	Subramani	Database Management Systems

The status bar at the bottom indicates the message '0 row(s) returned'.

### 3.5

Create a query to list the names of teachers and the courses they are assigned to. Join the "Teacher" table with the "Courses" table.



The screenshot shows the MySQL Workbench interface with a query editor window titled 'task\_3'. The code entered is identical to the one in question 3.4:

```
13
14 • SELECT s.first_name, s.last_name
15   FROM Students s
16   LEFT JOIN Enrollments e ON s.student_id = e.student_id
17   WHERE e.student_id IS NULL;
18
19 • SELECT DISTINCT s.first_name, s.last_name, c.course_name
20   FROM Students s
```

The result grid displays the following data:

first_name	last_name	course_name
Revathi	Krishnan	Computer Networks
Revathi	Krishnan	Operating Systems
Ganesh	Narayanan	Data Science Fundamentals
Ganesh	Narayanan	Software Engineering Principles
Ganesh	Narayanan	Cyber Security Basics
Uma	Ravi	Mathematics for AI
Uma	Ravi	Artificial Intelligence
Uma	Ravi	Machine Learning
Harish	Selvam	Database Management Systems

The status bar at the bottom indicates the message '0 row(s) returned'.

### 3.6

Retrieve a list of students and their enrollment dates for a specific course. You'll need to join the "Students" table with the "Enrollments" and "Courses" tables.

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is `hexa (sisdb)`.
- Tables:** The `sisdb` schema contains tables: `courses`, `enrollments`, `payments`, `students`, and `teachers`.
- Query Editor:** The query is:37 FROM Teachers t  
38 RIGHT JOIN Courses c ON t.teacher\_id = c.teacher\_id  
39 ORDER BY t.last\_name, t.first\_name;  
40  
41 -- 6  
42 • SELECT s.student\_id, s.first\_name, s.last\_name, e.enrollment\_date  
43 FROM Students s  
44 JOIN Enrollments e ON s.student\_id = e.student\_id  
45 JOIN Courses c ON e.course\_id = c.course\_id  
46 WHERE c.course\_name = 'Machine Learning';  
47  
48  
49 -- 7
- Result Grid:** The result grid shows two rows of data:| student\_id | first\_name | last\_name | enrollment\_date |
| --- | --- | --- | --- |
| 10 | Janani | Sridhar | 2024-03-10 |
| 13 | Arjun | Kumar | 2025-03-21 |

### 3.7

Find the names of students who have not made any payments. Use a LEFT JOIN between the "Students" table and the "Payments" table and filter for students with NULL payment records.

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is `hexa (sisdb)`.
- Tables:** The `sisdb` schema contains tables: `courses`, `enrollments`, `payments`, `students`, and `teachers`.
- Query Editor:** The query is:55 FROM Students s, Courses c  
56 WHERE s.first\_name = 'Arjun' AND s.last\_name = 'Kumar'  
57 AND c.course\_name = 'Machine Learning';  
58 /\* added a person enrolled but not paid \*/  
59 • SELECT s.first\_name, s.last\_name  
60 FROM Students s  
61 LEFT JOIN Payments p ON s.student\_id = p.student\_id  
62 WHERE p.payment\_id IS NULL;  
63 • select \* from payments;  
64  
65 -- 8  
66  
67 • SELECT s.student\_id, s.first\_name, s.last\_name, e.enrollment\_date
- Result Grid:** The result grid shows one row of data:| first\_name | last\_name |
| --- | --- |
| Arjun | Kumar |

### 3.8

Write a query to identify courses that have no enrollments. You'll need to use a LEFT JOIN between the "Courses" table and the "Enrollments" table and filter for courses with NULL enrollment records.

The screenshot shows the MySQL Workbench interface with a query editor window. The code is as follows:

```
56 WHERE s.first_name = 'Arjun' AND s.last_name = 'Kumar'
57 AND c.course_name = 'Machine Learning';
58 /* added a person enrolled but not paid */
59 • SELECT s.first_name, s.last_name
60 FROM Students s
61 LEFT JOIN Payments p ON s.student_id = p.student_id
62 WHERE p.payment_id IS NULL;
63 • select * from payments;
64
65 -- 8
66
67 • SELECT c.course_id, c.course_name
68 FROM Courses c
```

The result grid shows one row:

course_id	course_name
9	Cyber Security Basics

### 3.9

Identify students who are enrolled in more than one course. Use a self-join on the "Enrollments" table to find students with multiple enrollment records.

The screenshot shows the MySQL Workbench interface with a query editor window. The code is as follows:

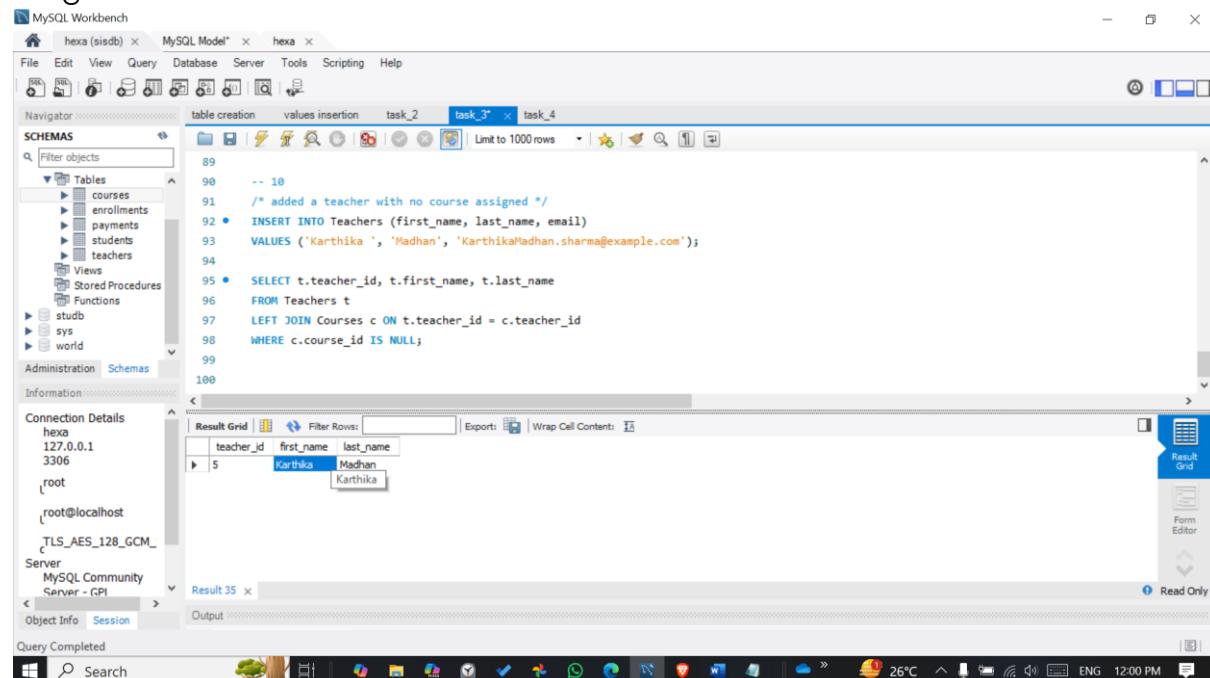
```
79 WHERE s.first_name = 'Arjun' AND s.last_name = 'Kumar'
80 AND c.course_name = 'Artificial Intelligence';
81
82 /* added a person course enrollment for example */
83
84 • SELECT s.student_id, s.first_name, s.last_name, COUNT(e.course_id) AS total_courses
85 FROM Students s
86 JOIN Enrollments e ON s.student_id = e.student_id
87 GROUP BY s.student_id, s.first_name, s.last_name
88 HAVING COUNT(e.course_id) > 1;
89
90
```

The result grid shows one row:

student_id	first_name	last_name	total_courses
13	Arjun	Kumar	2

### 3.10

Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher" table and the "Courses" table and filter for teachers with NULL course assignments.



The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar displays the Navigator with Schemas (hexa (sisdb)), Tables (courses, enrollments, payments, students, teachers), Views, Stored Procedures, Functions, stubs, sys, and world. The main area shows a query editor with the following code:

```
89
90 -- 10
91 /* added a teacher with no course assigned */
92 • INSERT INTO Teachers (first_name, last_name, email)
93 VALUES ('Karthika ', 'Madhan', 'KarthikaMadhan.sharma@example.com');
94
95 • SELECT t.teacher_id, t.first_name, t.last_name
96 FROM Teachers t
97 LEFT JOIN Courses c ON t.teacher_id = c.teacher_id
98 WHERE c.course_id IS NULL;
99
100
```

The Result Grid shows the output of the query:

teacher_id	first_name	last_name
5	Karthika	Madhan

The status bar at the bottom indicates a connection to 127.0.0.1, port 3306, user root@localhost, and session TLS\_AES\_128\_GCM\_. The system tray shows the date and time as 26°C, ENG, 12:00 PM.

# Task 4. Subquery and its type:

## 4.1.

Write an SQL query to calculate the average number of students enrolled in each course. Use aggregate functions and subqueries to achieve this.

The screenshot shows the MySQL Workbench interface with a query editor window titled "task\_4\*". The code is as follows:

```
1 • SELECT AVG(enrollment_count) AS avg_students_per_course
2   FROM (
3     SELECT course_id, COUNT(student_id) AS enrollment_count
4       FROM Enrollments
5      GROUP BY course_id
6   ) AS course_enrollment_counts;
```

The result grid shows one row with the value 1.2222.

## 4.2

Identify the student(s) who made the highest payment. Use a subquery to find the maximum payment amount and then retrieve the student(s) associated with that amount.

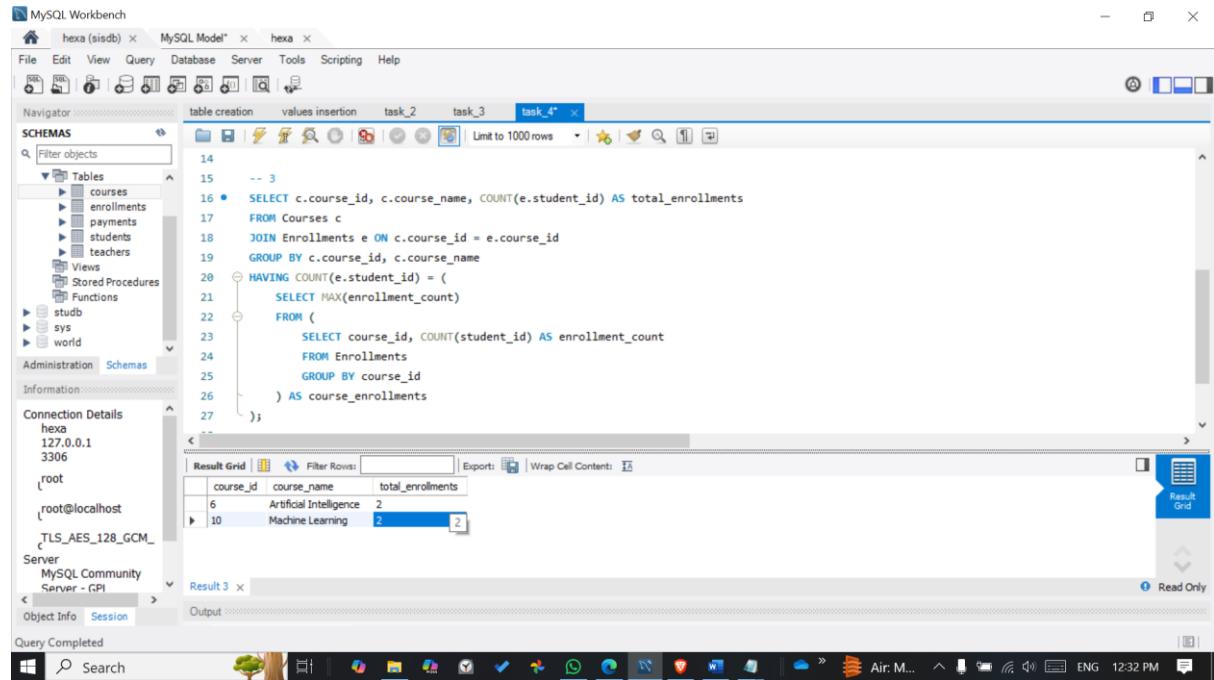
The screenshot shows the MySQL Workbench interface with a query editor window titled "task\_4\*". The code is as follows:

```
4
5   SELECT course_id, COUNT(student_id) AS enrollment_count
6     FROM Enrollments
7    GROUP BY course_id
8  ) AS course_enrollment_counts;
9
10 •  SELECT s.student_id, s.first_name, s.last_name, p.amount
11    FROM Students s
12   JOIN Payments p ON s.student_id = p.student_id
13  WHERE p.amount = (SELECT MAX(amount) FROM Payments);
14
15
```

The result grid shows one row with student\_id 7, first\_name Sundar, last\_name Babu, and amount 950.00.

#### 4.3.

Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the course(s) with the maximum enrollment count.



The screenshot shows the MySQL Workbench interface with a query editor window titled "task\_4". The code is a subquery used to find courses with the highest enrollment count:

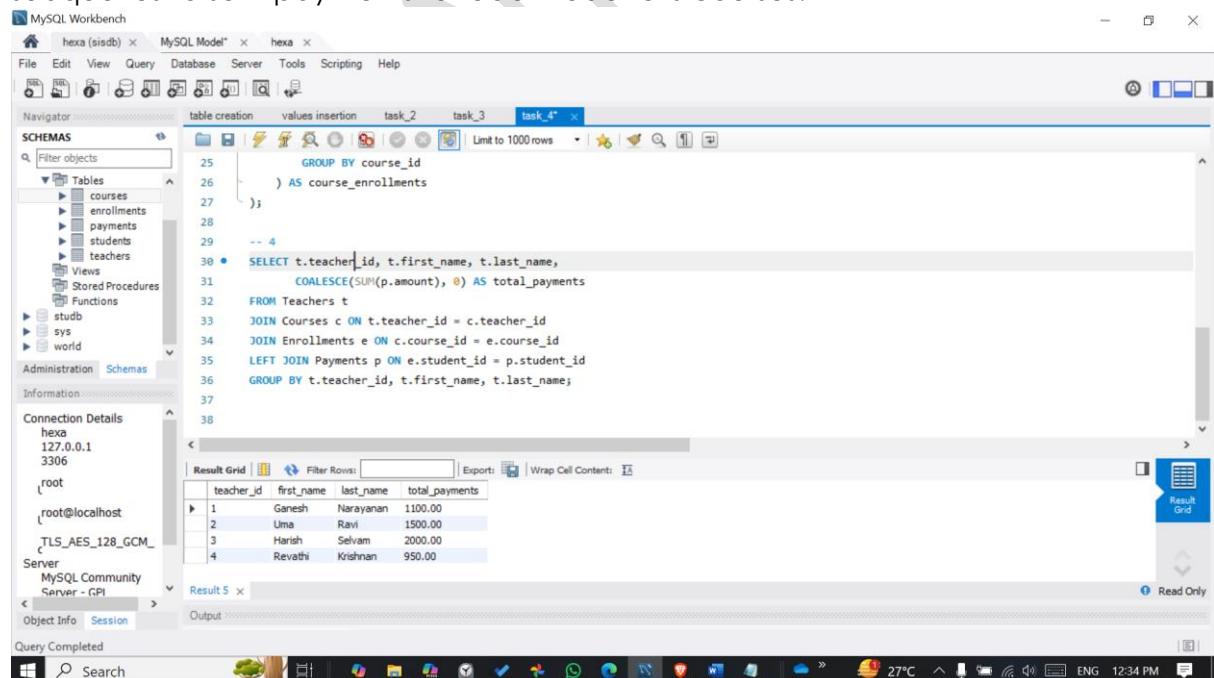
```
14
15 -- 3
16 • SELECT c.course_id, c.course_name, COUNT(e.student_id) AS total_enrollments
17 FROM Courses c
18 JOIN Enrollments e ON c.course_id = e.course_id
19 GROUP BY c.course_id, c.course_name
20 HAVING COUNT(e.student_id) =
21     (SELECT MAX(enrollment_count)
22      FROM (
23          SELECT course_id, COUNT(student_id) AS enrollment_count
24          FROM Enrollments
25          GROUP BY course_id
26      ) AS course_enrollments
27 );
```

The result grid shows two rows of data:

course_id	course_name	total_enrollments
6	Artificial Intelligence	2
10	Machine Learning	2

#### 4.4

Calculate the total payments made to courses taught by each teacher. Use subqueries to sum payments for each teacher's courses.



The screenshot shows the MySQL Workbench interface with a query editor window titled "task\_4". The code uses a subquery to calculate total payments for each teacher:

```
25
26     GROUP BY course_id
27 );
28
29 -- 4
30 • SELECT t.teacher_id, t.first_name, t.last_name,
31     COALESCE(SUM(p.amount), 0) AS total_payments
32 FROM Teachers t
33 JOIN Courses c ON t.teacher_id = c.teacher_id
34 JOIN Enrollments e ON c.course_id = e.course_id
35 LEFT JOIN Payments p ON e.student_id = p.student_id
36 GROUP BY t.teacher_id, t.first_name, t.last_name;
37
38
```

The result grid shows four teachers with their names and total payments:

teacher_id	first_name	last_name	total_payments
1	Ganesh	Narayanan	1100.00
2	Uma	Ravi	1500.00
3	Harish	Selvam	2000.00
4	Revathi	Krishnan	950.00

## 4.5

Identify students who are enrolled in all available courses. Use subqueries to compare a student's enrollments with the total number of courses.

The screenshot shows the MySQL Workbench interface with a query editor window titled "task\_4". The code is as follows:

```
34 JOIN Enrollments e ON c.course_id = e.course_id
35 LEFT JOIN Payments p ON e.student_id = p.student_id
36 GROUP BY t.teacher_id, t.first_name, t.last_name;
37
38 -- 5
39 -- no students have enrolled in all courses so the result is null
40 • SELECT s.student_id, s.first_name, s.last_name
  FROM Students s
41
42 WHERE (
43   SELECT COUNT(DISTINCT e.course_id)
44   FROM Enrollments e
45   WHERE e.student_id = s.student_id
46 ) = (SELECT COUNT(course_id) FROM Courses);
```

The result grid shows one row with student\_id, first\_name, and last\_name all set to null.

## 4.6

Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to find teachers with no course assignments.

The screenshot shows the MySQL Workbench interface with a query editor window titled "task\_4". The code is as follows:

```
44
45   FROM Enrollments e
46   WHERE e.student_id = s.student_id
47 ) = (SELECT COUNT(course_id) FROM Courses);

48 /* 6. Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to
49   find teachers with no course assignments. */
50
51 • SELECT t.teacher_id, t.first_name, t.last_name
  FROM Teachers t
52
53 WHERE t.teacher_id NOT IN (SELECT DISTINCT teacher_id FROM Courses WHERE teacher_id IS NOT NULL);
54
55
56
57
```

The result grid shows one row with teacher\_id, first\_name, and last\_name all set to null.

## 4.7

Calculate the average age of all students. Use subqueries to calculate the age of each student based on their date of birth.

The screenshot shows the MySQL Workbench interface with a query editor window titled "task\_4". The code entered is:

```
50
51 • SELECT t.teacher_id, t.first_name, t.last_name
  FROM Teachers t
  WHERE t.teacher_id NOT IN (SELECT DISTINCT teacher_id FROM Courses WHERE teacher_id IS NOT NULL);
52
53
54
55 /*7. Calculate the average age of all students. Use subqueries to calculate the age of each student
  based on their date of birth. */
56
57
58 • SELECT AVG(student_age) AS average_age
  FROM (
59   SELECT student_id, DATE_FORMAT(NOW(), '%Y') - YEAR(date_of_birth) AS student_age
    FROM students
  ) subquery;
```

The result grid shows one row with the value "26.0000". The status bar at the bottom right indicates "Duration / Fetch: 0.000 sec" and "0.015 sec / 0.000 sec".

## 4.8

Identify courses with no enrollments. Use subqueries to find courses without enrollment records.

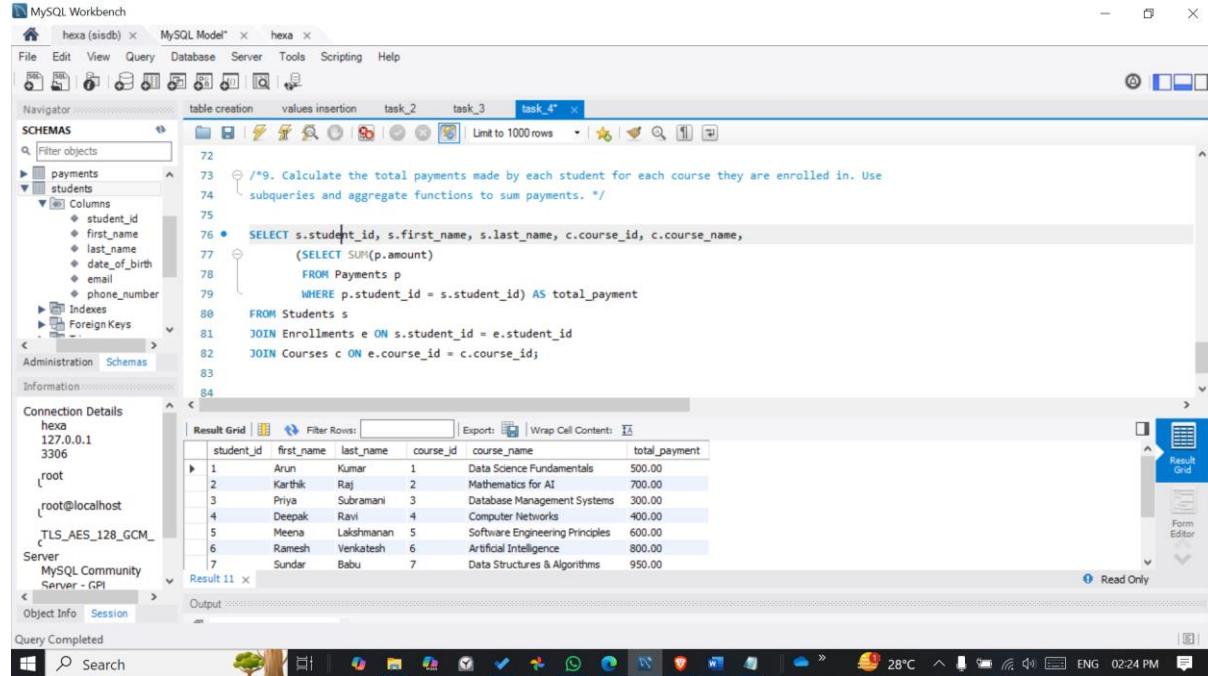
The screenshot shows the MySQL Workbench interface with a query editor window titled "task\_4". The code entered is:

```
63
64 /*8. Identify courses with no enrollments. Use subqueries to find courses without enrollment
  records. */
65
66
67 • SELECT course_id, course_name
  FROM Courses
  WHERE course_id NOT IN (
68   SELECT DISTINCT course_id FROM Enrollments
69 );
70
71
72
73
74
```

The result grid shows one row with the course ID "9" and name "Cyber Security Basics". The status bar at the bottom right indicates "Duration / Fetch: 0.000 sec" and "0.015 sec / 0.000 sec".

## 4.9

Calculate the total payments made by each student for each course they are enrolled in. Use subqueries and aggregate functions to sum payments.



The screenshot shows the MySQL Workbench interface with the following details:

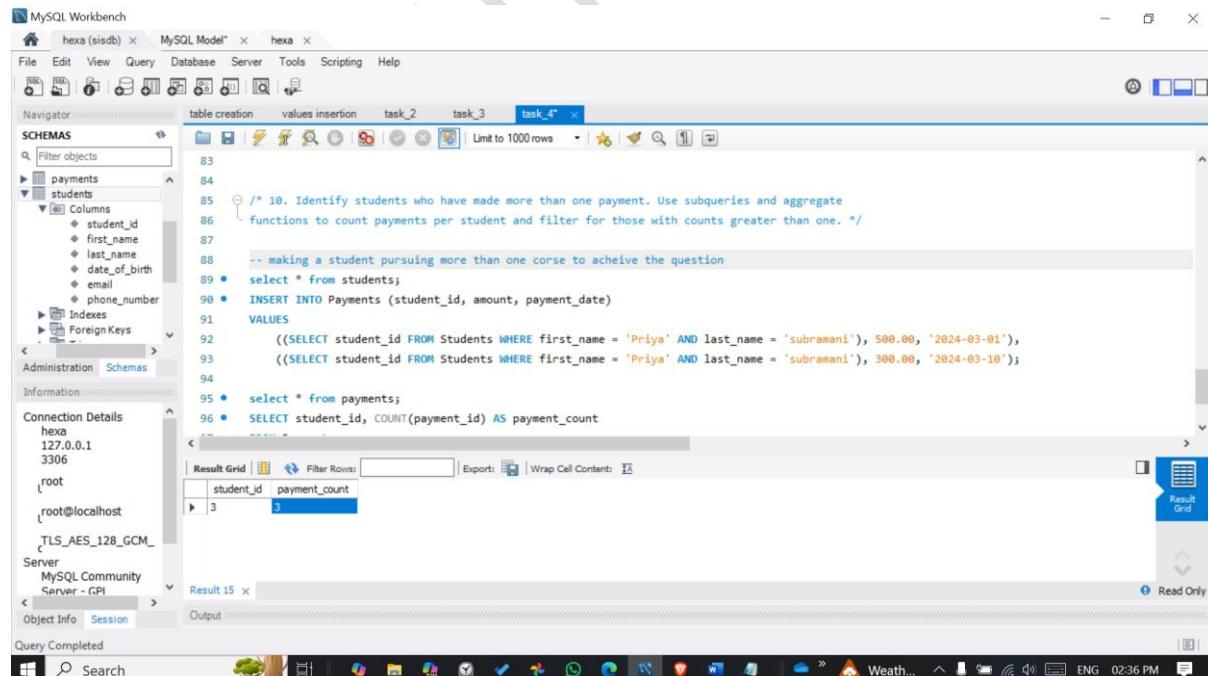
- Navigator:** Shows the `payments` and `students` schemas.
- Query Editor:** Contains the following SQL query:

```
/*9. Calculate the total payments made by each student for each course they are enrolled in. Use subqueries and aggregate functions to sum payments. */

SELECT s.student_id, s.first_name, s.last_name, c.course_id, c.course_name,
       (SELECT SUM(p.amount)
        FROM Payments p
        WHERE p.student_id = s.student_id) AS total_payment
  FROM Students s
 JOIN Enrollments e ON s.student_id = e.student_id
 JOIN Courses c ON e.course_id = c.course_id;
```
- Result Grid:** Displays the results of the query, showing student information and their total payment for each course.

student_id	first_name	last_name	course_id	course_name	total_payment
1	Arun	Kumar	1	Data Science Fundamentals	500.00
2	Karthik	Raj	2	Mathematics for AI	700.00
3	Priya	Subramani	3	Database Management Systems	300.00
4	Deepak	Ravi	4	Computer Networks	400.00
5	Meena	Lakshmanan	5	Software Engineering Principles	600.00
6	Ramesh	Venkatesh	6	Artificial Intelligence	800.00
7	Sundar	Babu	7	Data Structures & Algorithms	950.00

## 4.10



The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the `payments` and `students` schemas.
- Query Editor:** Contains the following SQL query:

```
/* 10. Identify students who have made more than one payment. Use subqueries and aggregate functions to count payments per student and filter for those with counts greater than one. */

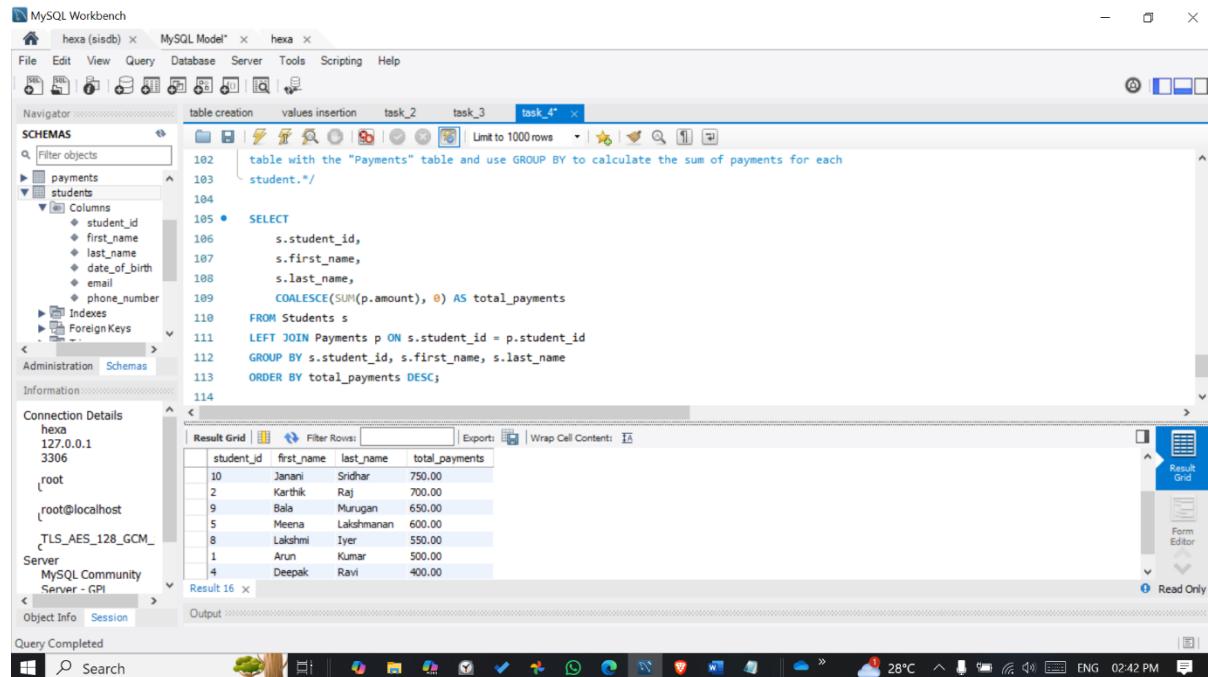
-- making a student pursuing more than one course to achieve the question
SELECT * FROM students;
INSERT INTO Payments (student_id, amount, payment_date)
VALUES
    ((SELECT student_id FROM Students WHERE first_name = 'Priya' AND last_name = 'Subramani'), 500.00, '2024-03-01'),
    ((SELECT student_id FROM Students WHERE first_name = 'Priya' AND last_name = 'Subramani'), 300.00, '2024-03-10');

SELECT * FROM payments;
SELECT student_id, COUNT(payment_id) AS payment_count
```
- Result Grid:** Displays the results of the query, showing student IDs and their payment counts.

student_id	payment_count
3	2

## 4.11

Write an SQL query to calculate the total payments made by each student. Join the "Students" table with the "Payments" table and use GROUP BY to calculate the sum of payments for each student.



The screenshot shows the MySQL Workbench interface with the following details:

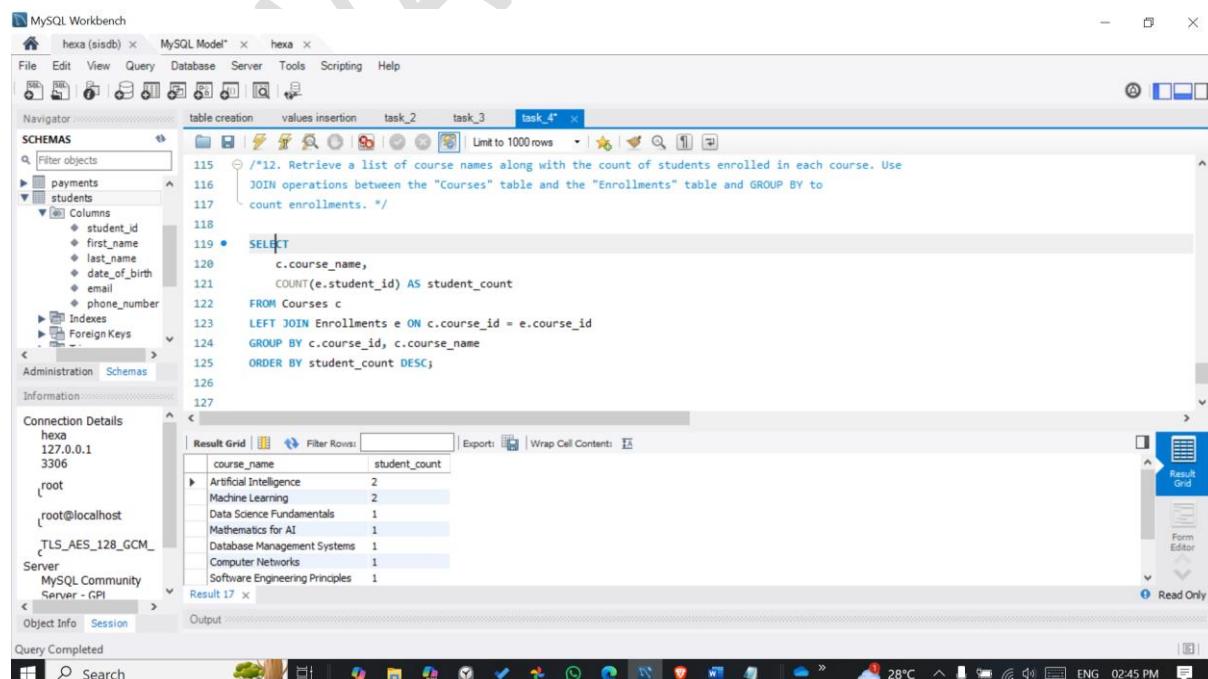
- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (payments, students), Columns (student\_id, first\_name, last\_name, date\_of\_birth, email, phone\_number), Indexes, Foreign Keys.
- Script Editor:** Task 4\* contains the following SQL code:

```
102    table with the "Payments" table and use GROUP BY to calculate the sum of payments for each
103    student.*/
104
105 • SELECT
106     s.student_id,
107     s.first_name,
108     s.last_name,
109     COALESCE(SUM(p.amount), 0) AS total_payments
110    FROM Students s
111   LEFT JOIN Payments p ON s.student_id = p.student_id
112  GROUP BY s.student_id, s.first_name, s.last_name
113 ORDER BY total_payments DESC;
114
```
- Result Grid:** Shows the results of the query, listing student\_id, first\_name, last\_name, and total\_payments. The data is:

student_id	first_name	last_name	total_payments
10	Janani	Sridhar	750.00
2	Karthik	Raj	700.00
9	Bala	Murugan	650.00
5	Meena	Lakshmanan	600.00
8	Lakshmi	Iyer	550.00
1	Arun	Kumar	500.00
4	Deepak	Ravi	400.00
- Session Bar:** Result 16, Read Only.
- System Bar:** Connection Details (hexa, 127.0.0.1, 3306, root, root@localhost, TLS\_AES\_128\_GCM\_), MySQL Community Server - GPL.

## 4.12

Retrieve a list of course names along with the count of students enrolled in each course. Use JOIN operations between the "Courses" table and the "Enrollments" table and GROUP BY to count enrollments.



The screenshot shows the MySQL Workbench interface with the following details:

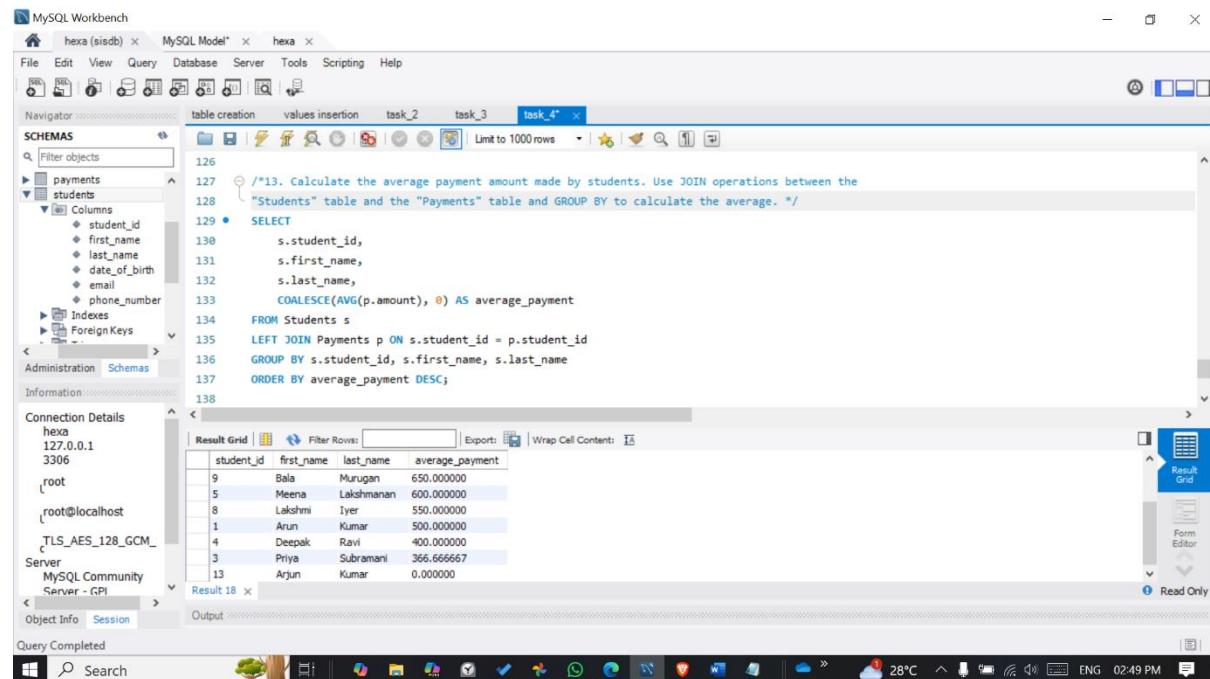
- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (payments, students), Columns (student\_id, first\_name, last\_name, date\_of\_birth, email, phone\_number), Indexes, Foreign Keys.
- Script Editor:** Task 4\* contains the following SQL code:

```
115 /*12. Retrieve a list of course names along with the count of students enrolled in each course. Use
116 JOIN operations between the "Courses" table and the "Enrollments" table and GROUP BY to
117 count enrollments. */
118
119 • SELECT
120     c.course_name,
121     COUNT(e.student_id) AS student_count
122    FROM Courses c
123   LEFT JOIN Enrollments e ON c.course_id = e.course_id
124  GROUP BY c.course_id, c.course_name
125 ORDER BY student_count DESC;
126
127
```
- Result Grid:** Shows the results of the query, listing course\_name and student\_count. The data is:

course_name	student_count
Artificial Intelligence	2
Machine Learning	2
Data Science Fundamentals	1
Mathematics for AI	1
Database Management Systems	1
Computer Networks	1
Software Engineering Principles	1
- Session Bar:** Result 17, Read Only.
- System Bar:** Connection Details (hexa, 127.0.0.1, 3306, root, root@localhost, TLS\_AES\_128\_GCM\_), MySQL Community Server - GPL.

#### 4.13

Calculate the average payment amount made by students. Use JOIN operations between the "Students" table and the "Payments" table and GROUP BY to calculate the average.



The screenshot shows the MySQL Workbench interface with a query editor window titled "task\_4". The code is a SQL query to calculate the average payment amount for students:

```
/*
 * 13. Calculate the average payment amount made by students. Use JOIN operations between the
 * "Students" table and the "Payments" table and GROUP BY to calculate the average. */
SELECT
    s.student_id,
    s.first_name,
    s.last_name,
    COALESCE(AVG(p.amount), 0) AS average_payment
FROM Students s
LEFT JOIN Payments p ON s.student_id = p.student_id
GROUP BY s.student_id, s.first_name, s.last_name
ORDER BY average_payment DESC;
```

The result grid displays the following data:

student_id	first_name	last_name	average_payment
9	Bala	Murugan	650.000000
5	Meena	Lakshmanan	600.000000
8	Lakshmi	Iyer	550.000000
1	Arun	Kumar	500.000000
4	Deepak	Ravi	400.000000
3	Priya	Subramani	366.666667
13	Arjun	Kumar	0.000000

**END OF THE DOCUMENT**

JASWANTH KUMAR

JASWANTH KUMAR

JASWANTH KUMAR

JASWANTH KUMAR

JASWANTH KUMAR