ASSIGNMENT

STUDENT INFORMATION SYSTEM (SQL IMPLIMENTATION)

BY

JASWANTH KUMAR S

BATCH -4
(PYTHON FOUNDATION)

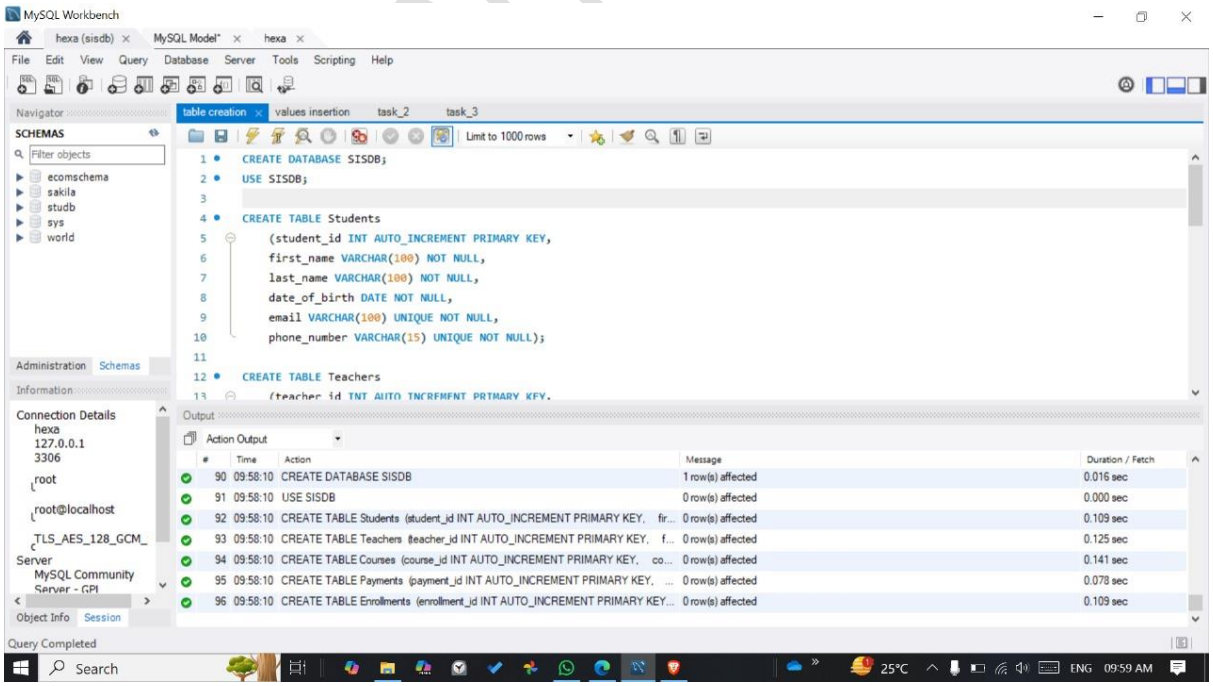
Task 1. Database Design:

***** I HAVE TAKEN A FEW SCREENSHOT TO SHOWCASAE ALTOGETHER ONLY FOR THIS TASK [TASK_1] AND TO LOOK UPON THE INSERTED VALUE , I'VE INCLUDED THE SQL SCRIPTS IN GTIHUB REPOSITORY*****

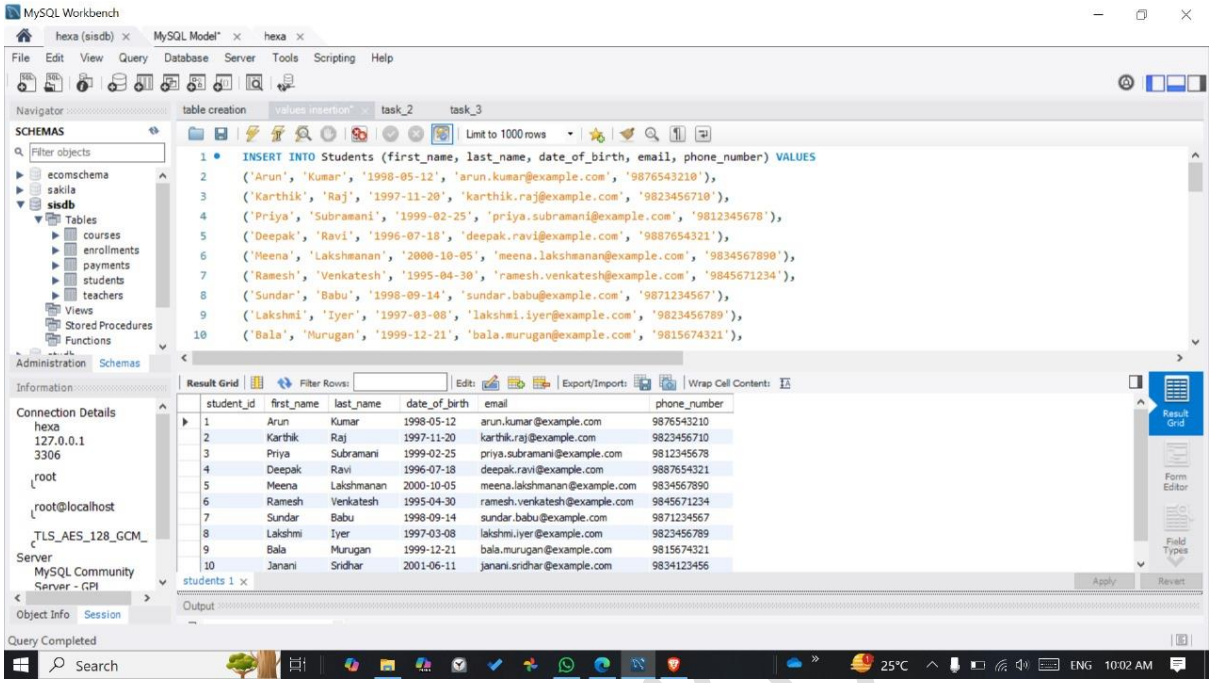
1. Create the database named "SISDB"

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

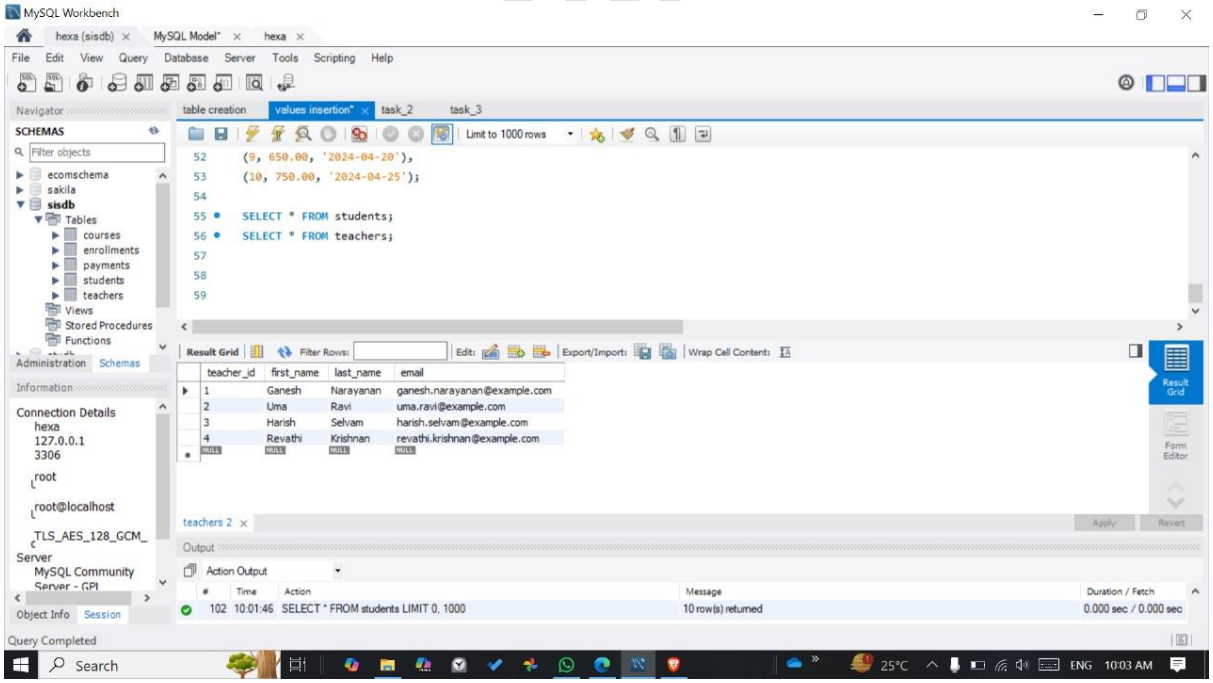
4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.



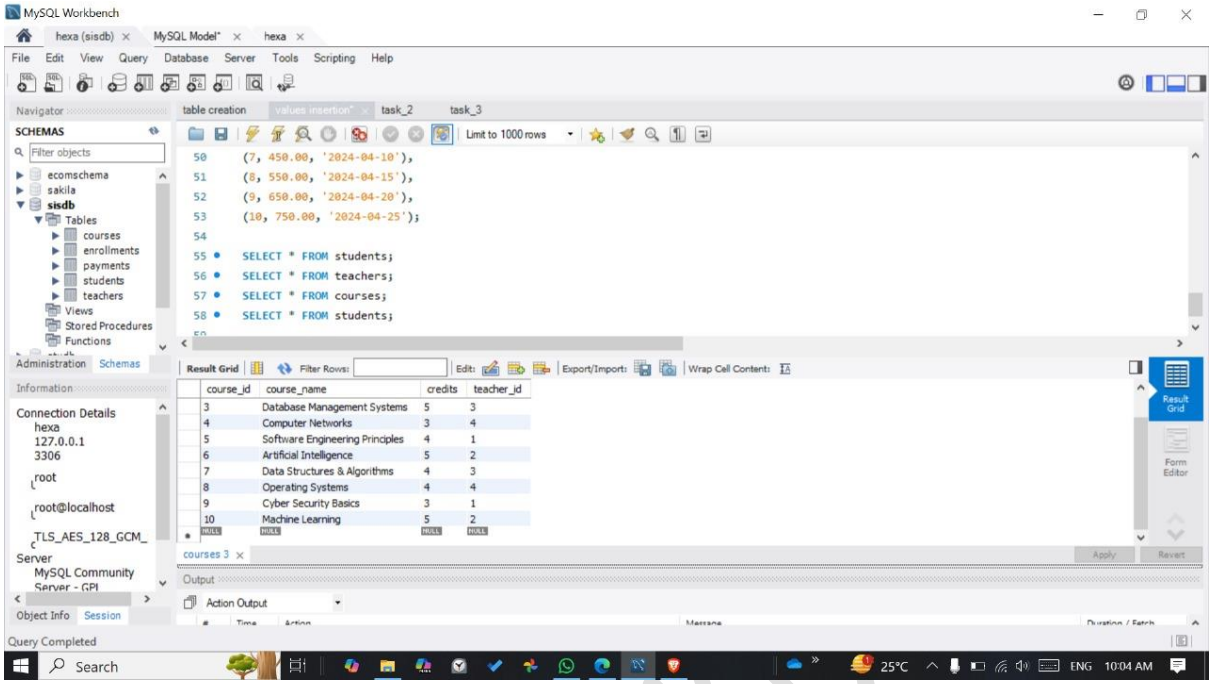
STUDENTS:



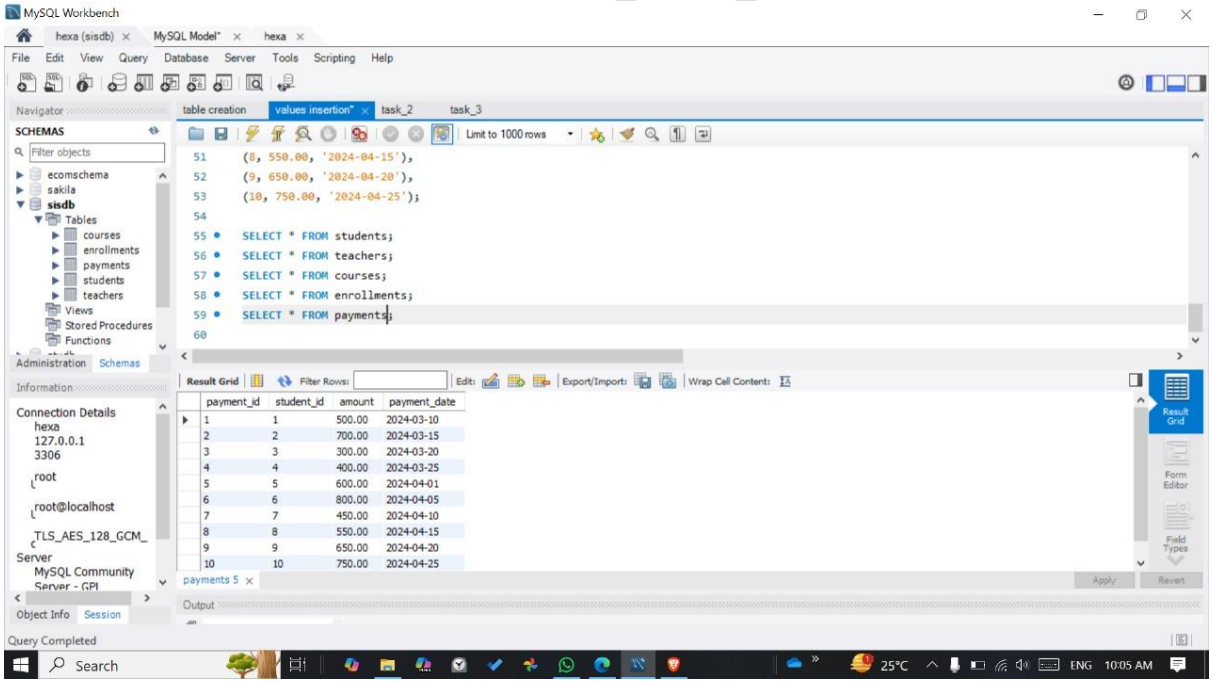
TEACHERS:



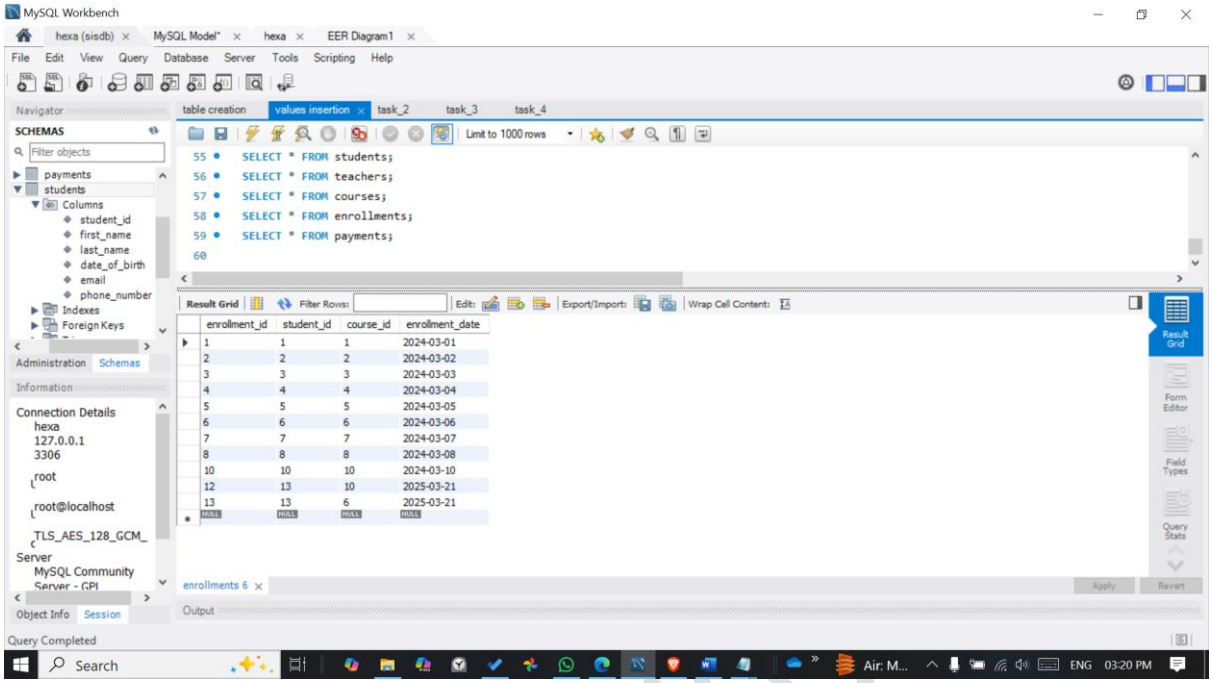
COURSES:



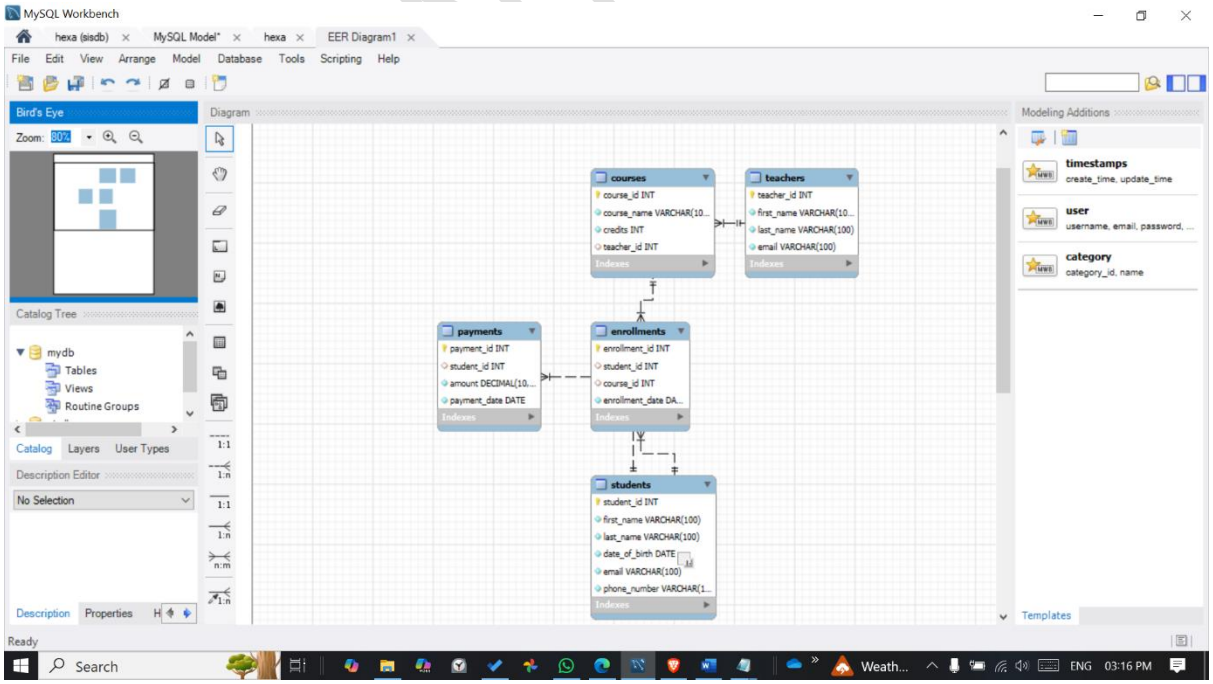
PAYMENTS:



ENROLMENTS:



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:

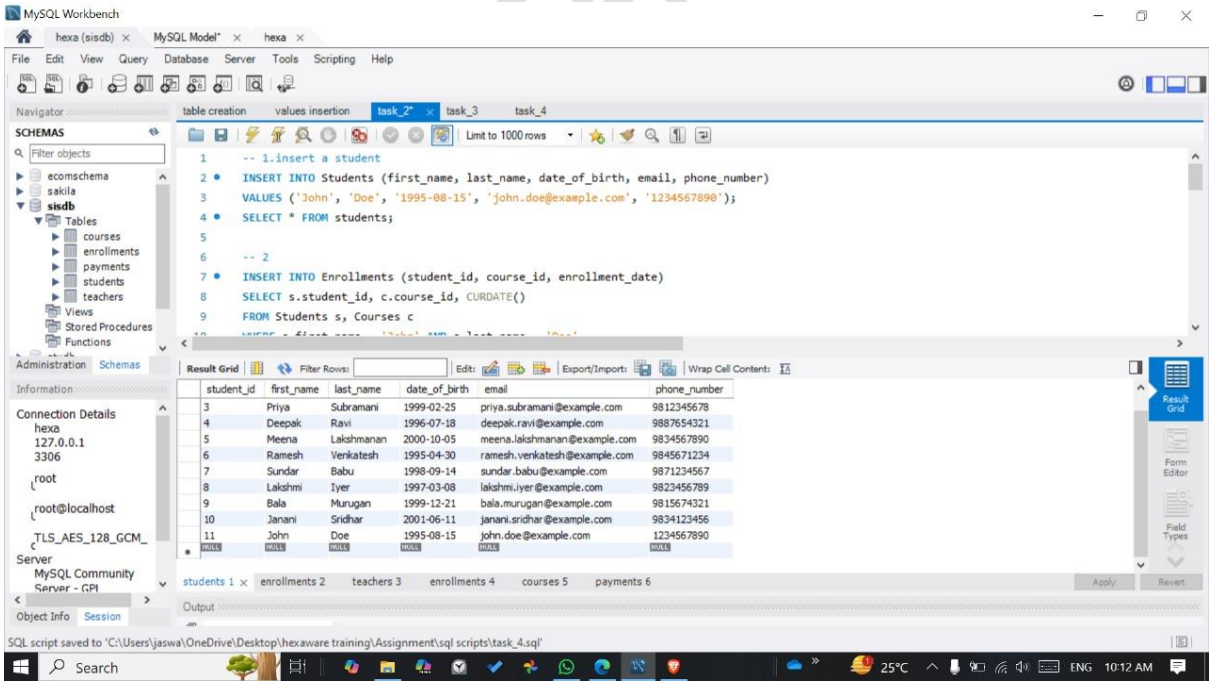
a. First Name: John

b. Last Name: Doe

c. Date of Birth: 1995-08-15

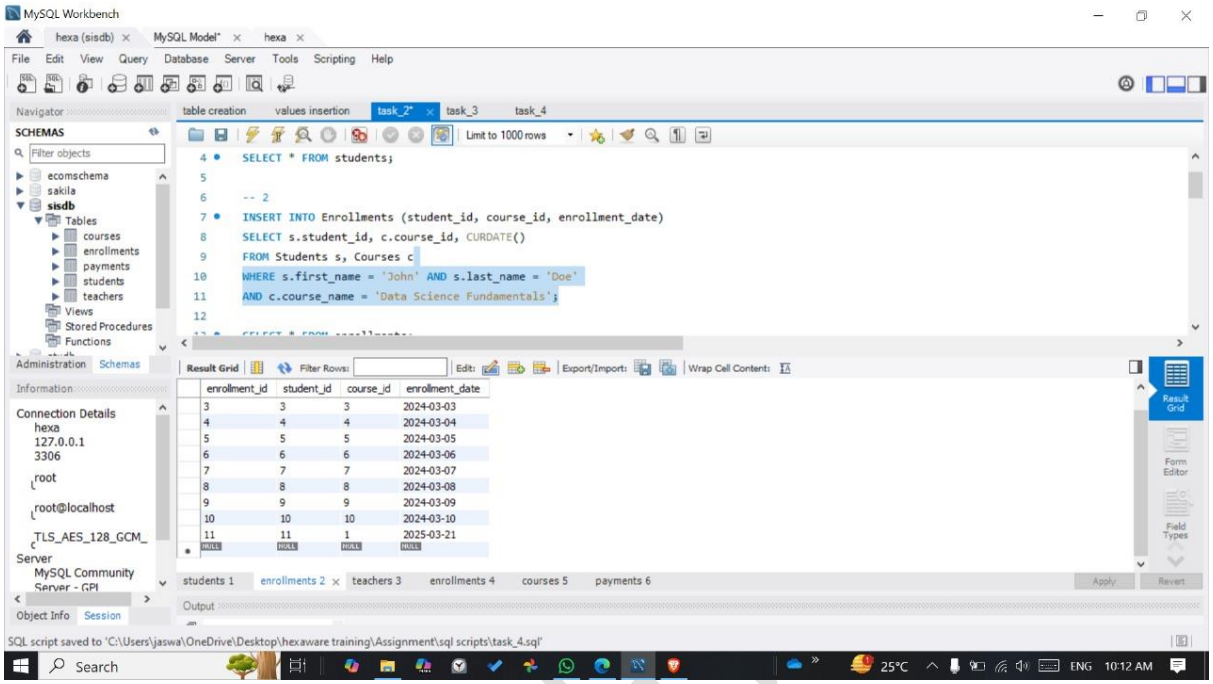
d. Email: john.doe@example.com

e. Phone Number: 1234567890



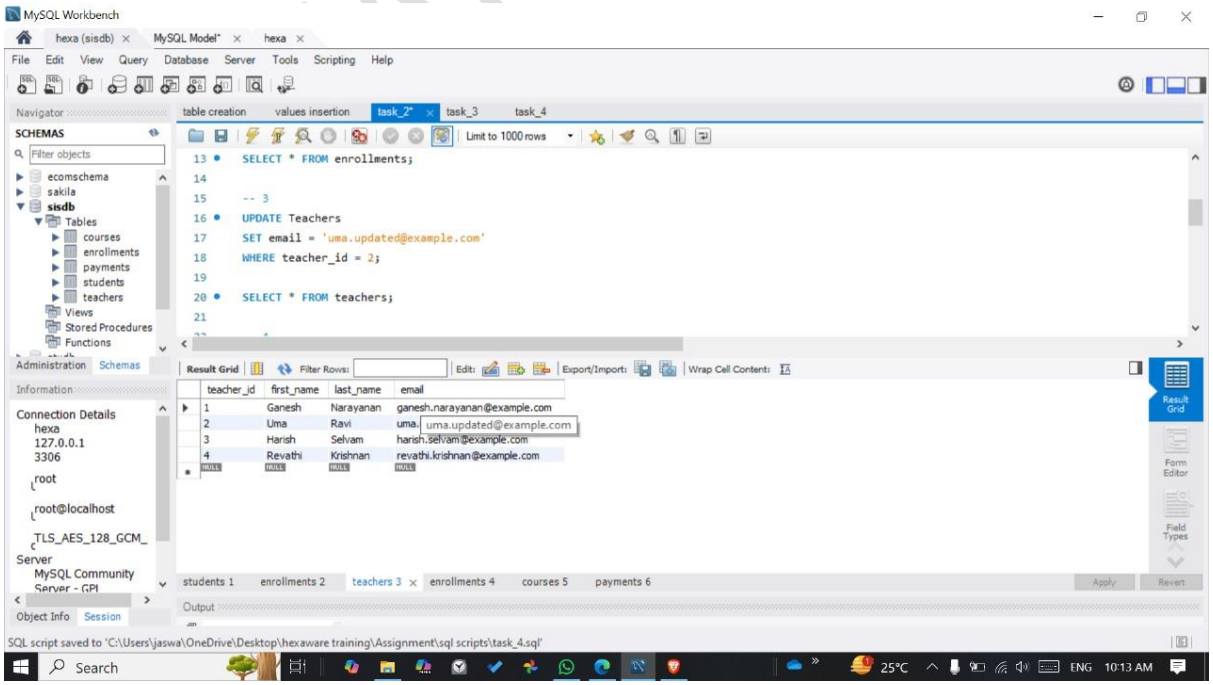
2.2

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.



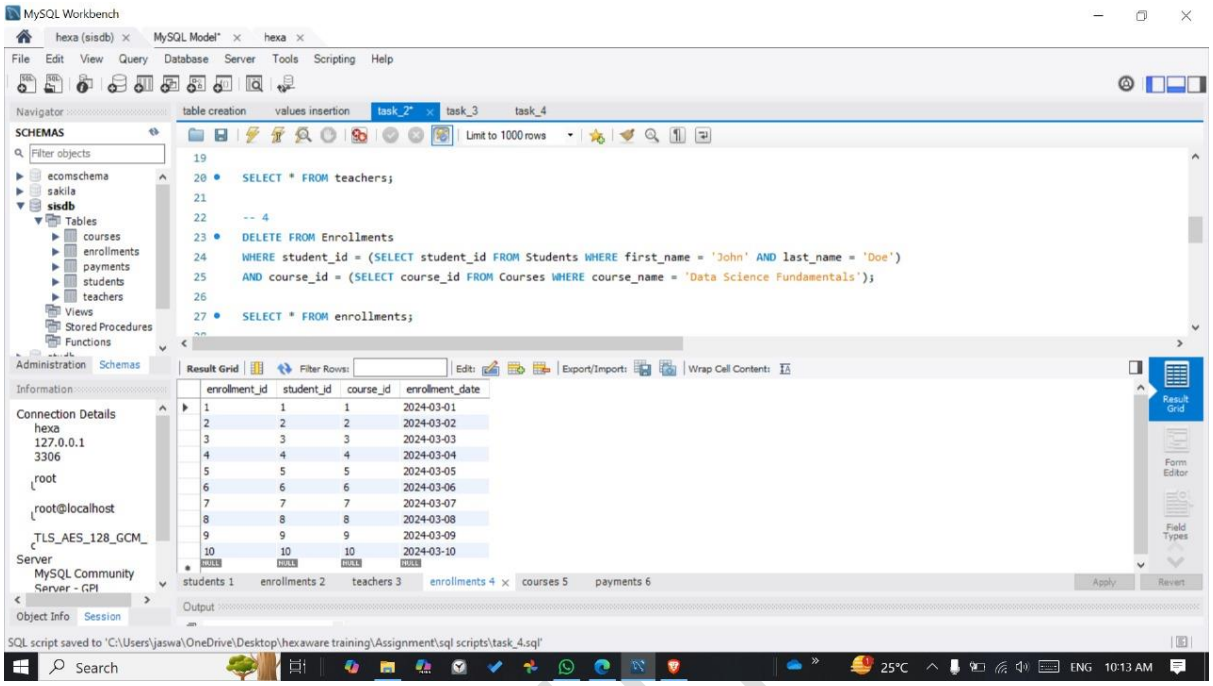
2.3

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



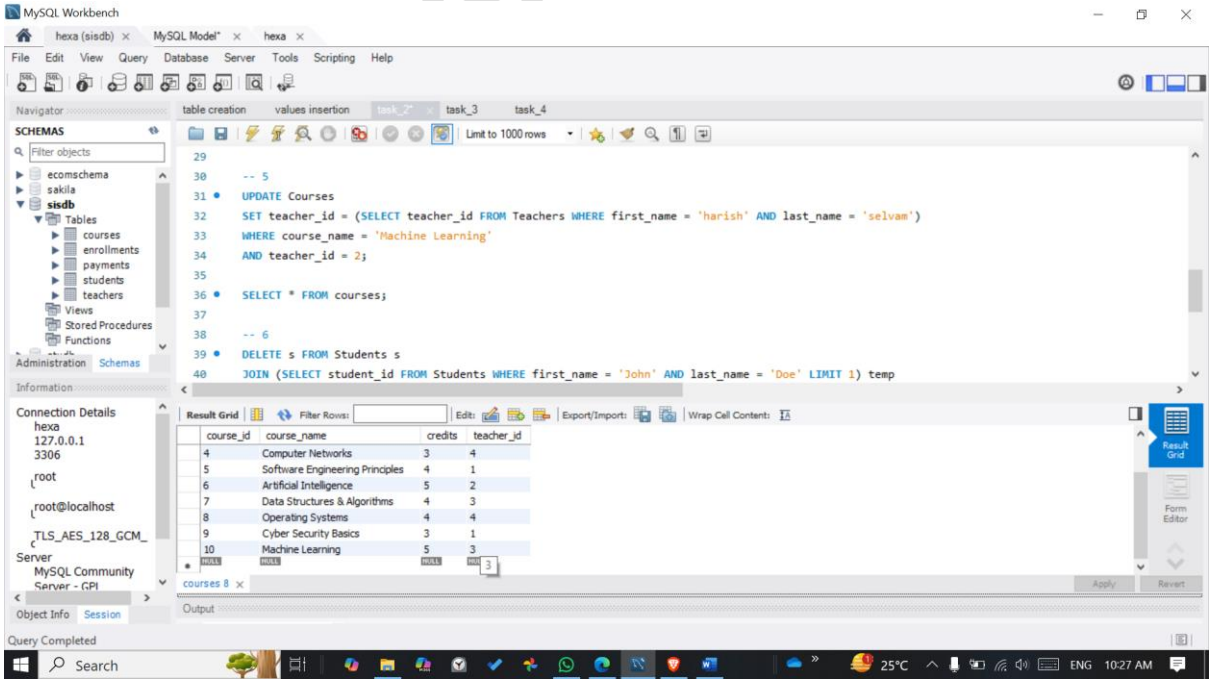
2.4

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



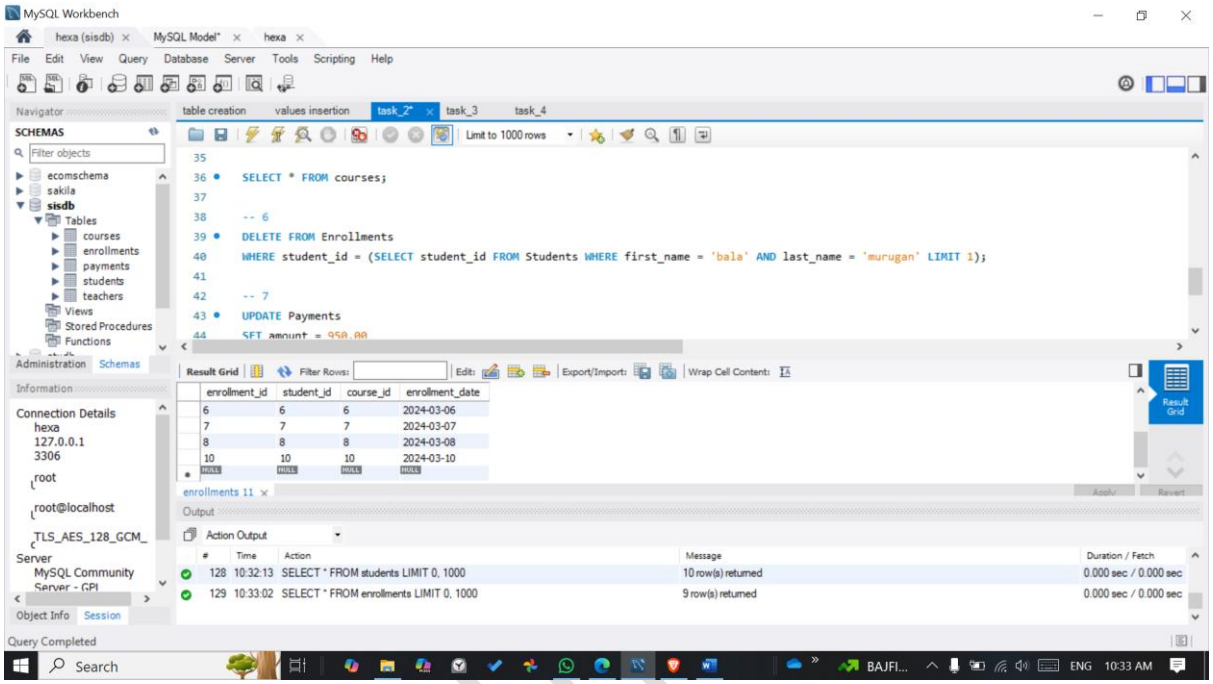
TASK 2.5

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



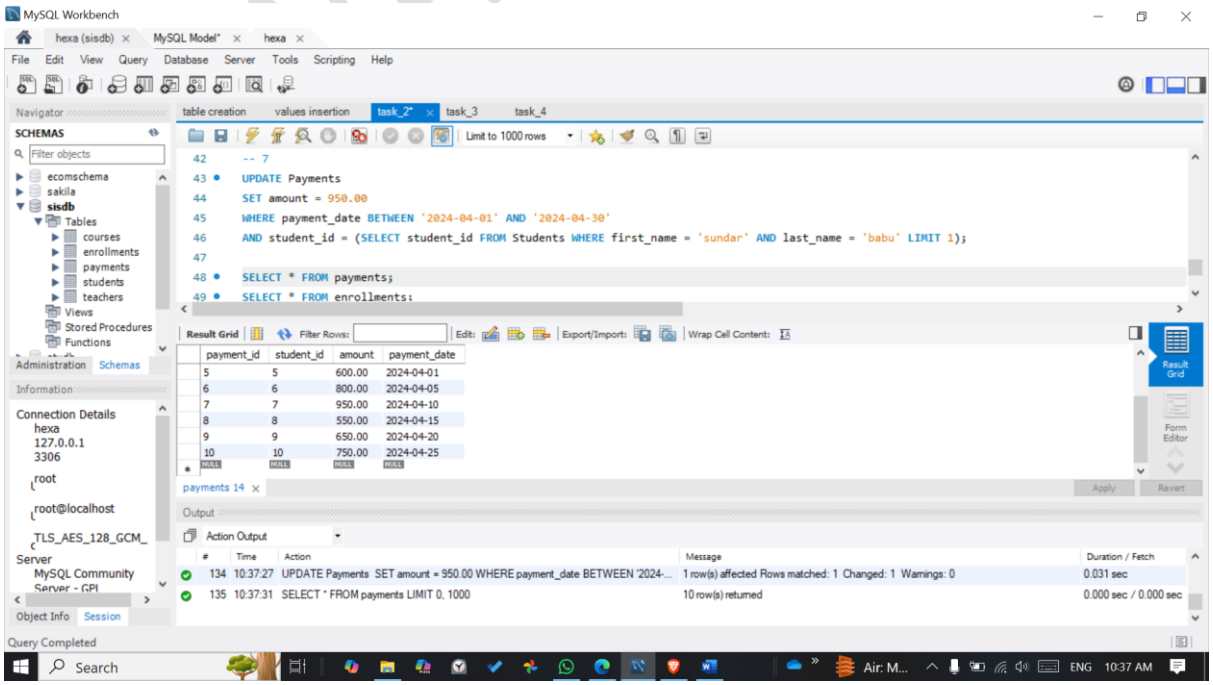
Task 2.6

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



Task 2.7

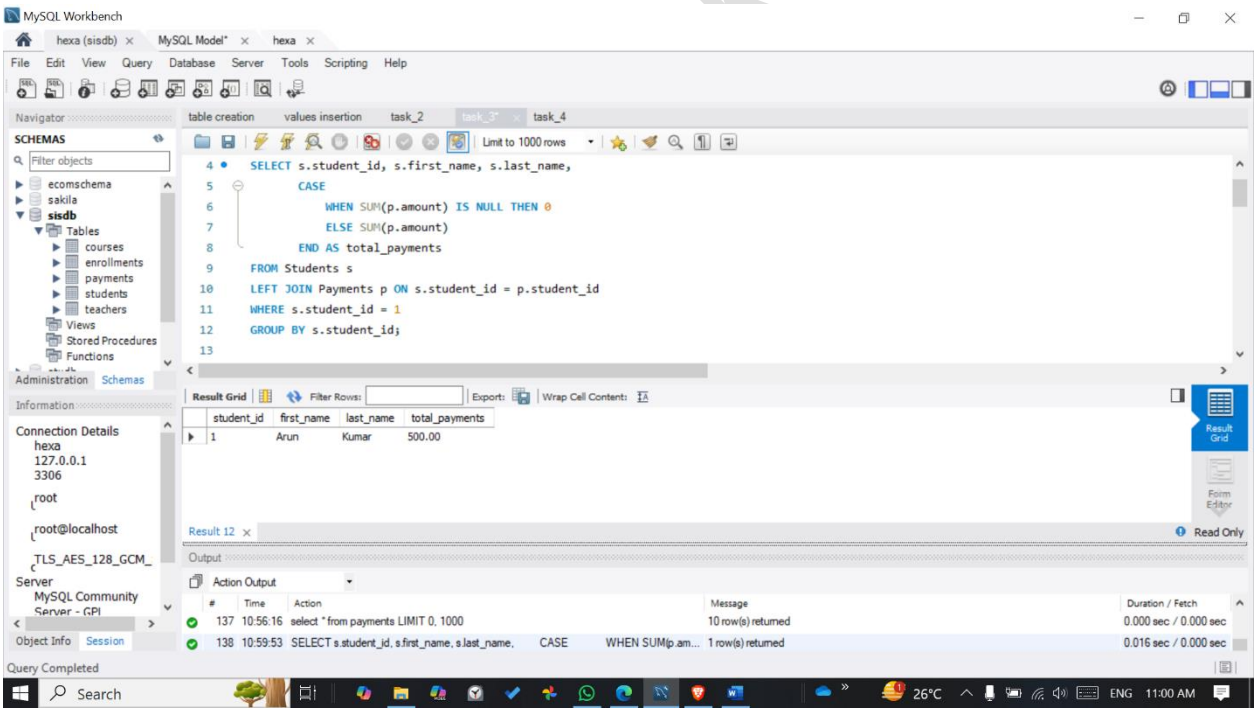
Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.



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

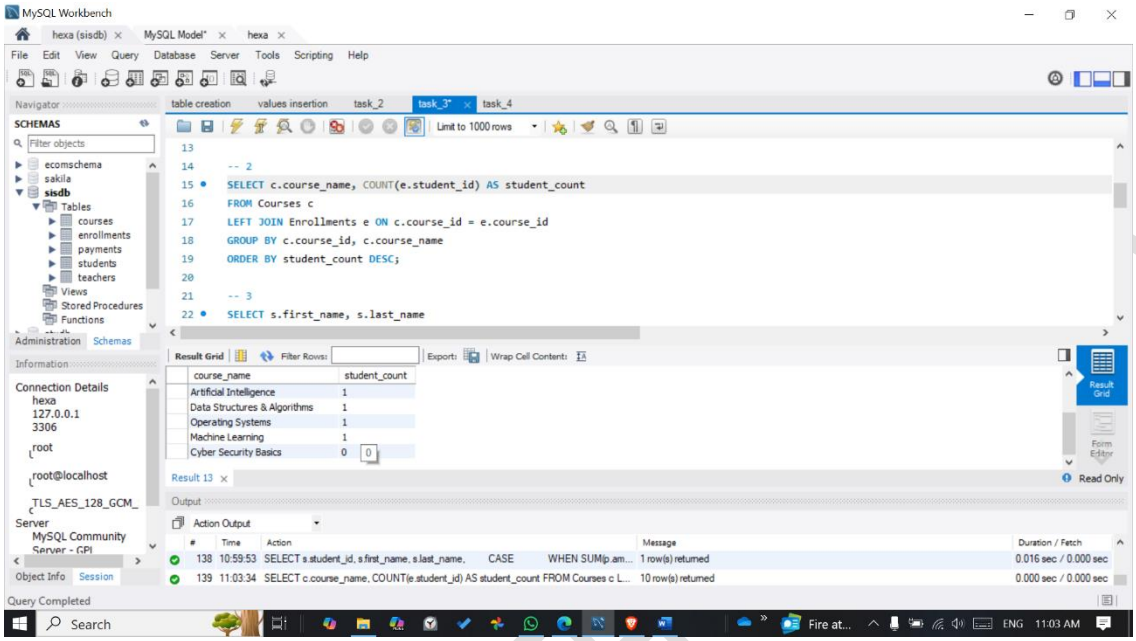
3.1

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.



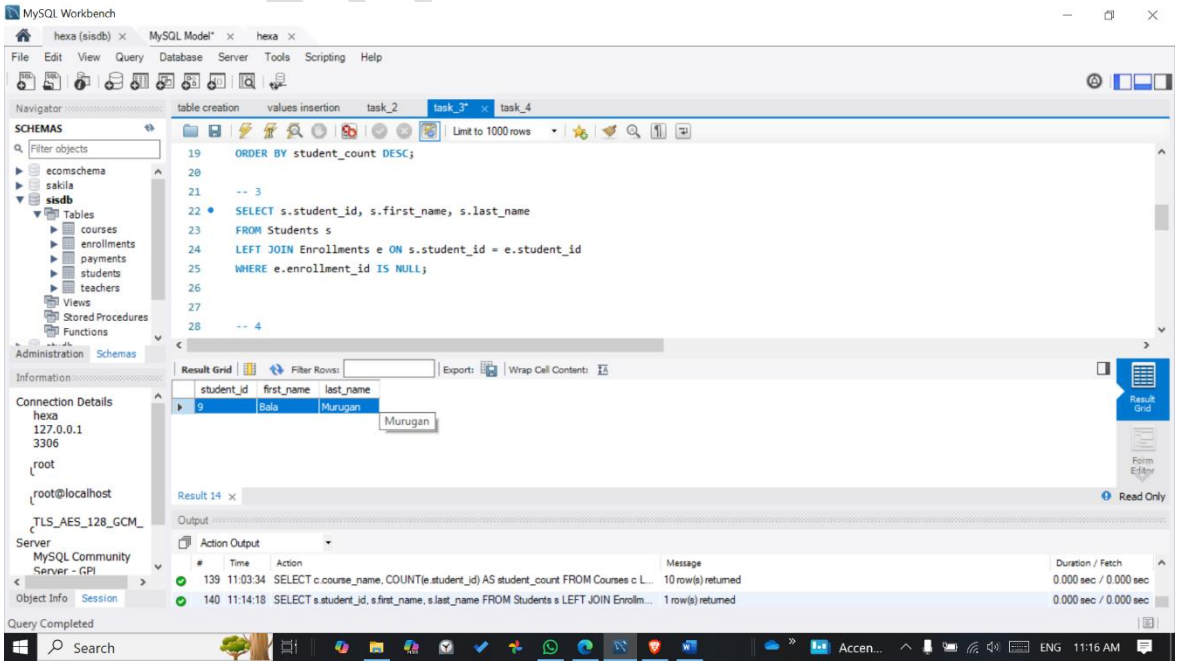
3.2

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.



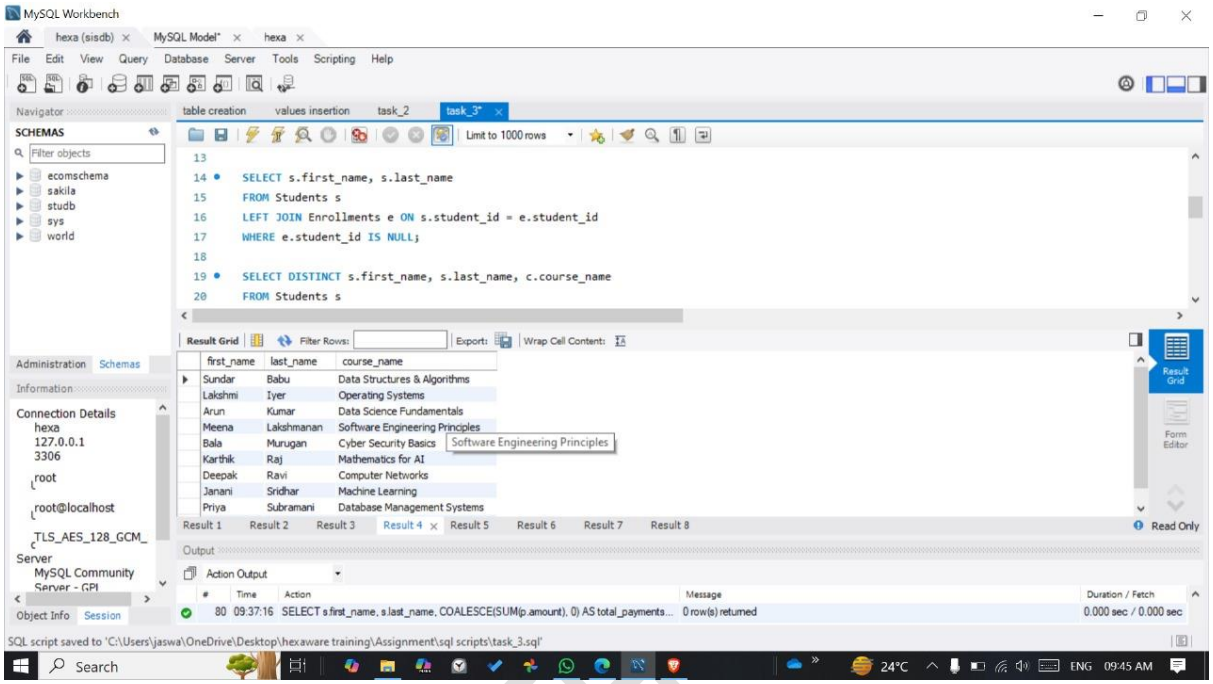
3.3

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.



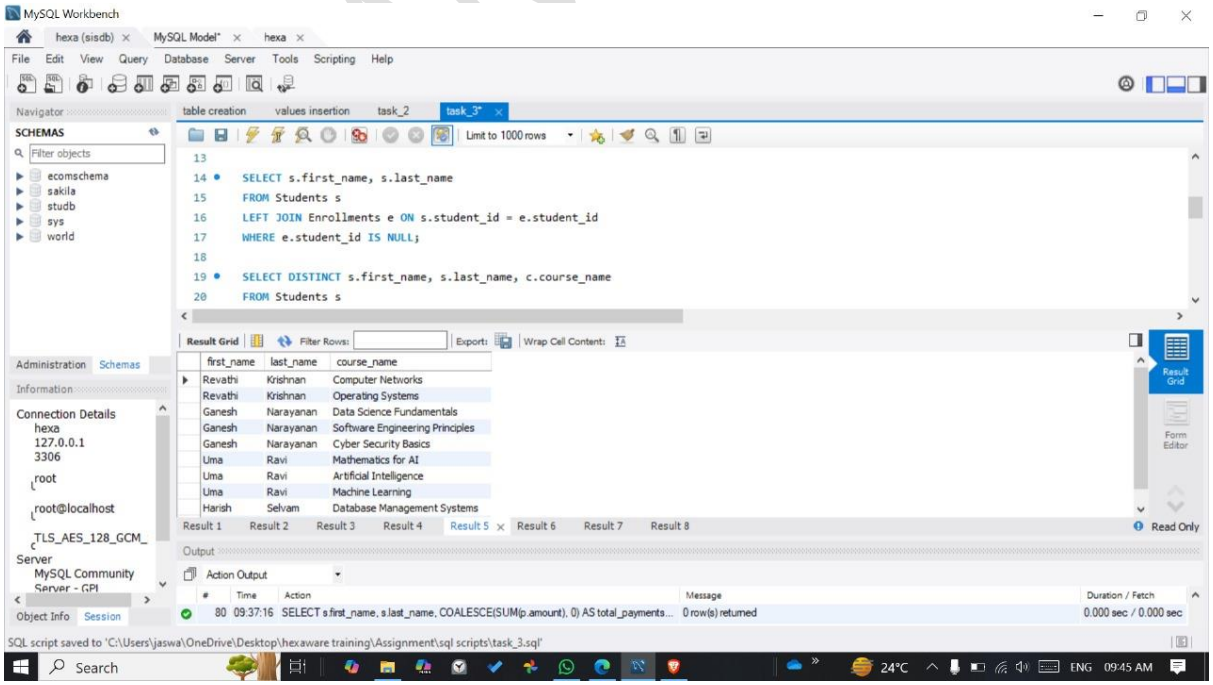
3.4

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.



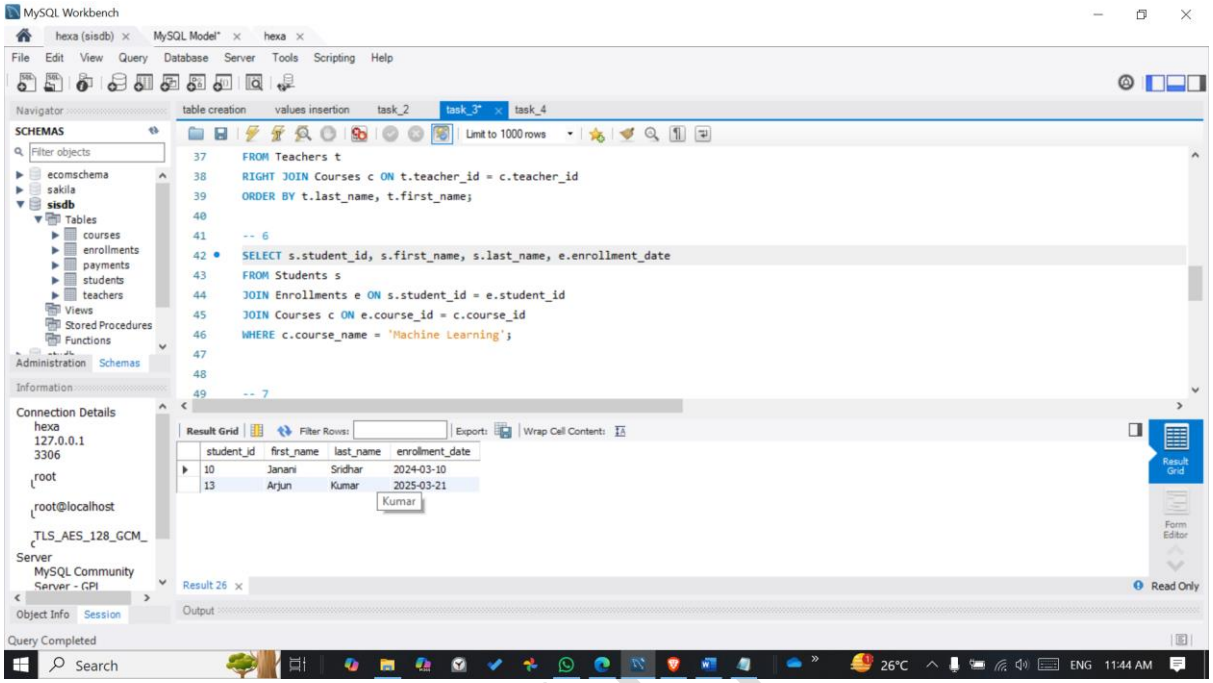
3.5

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



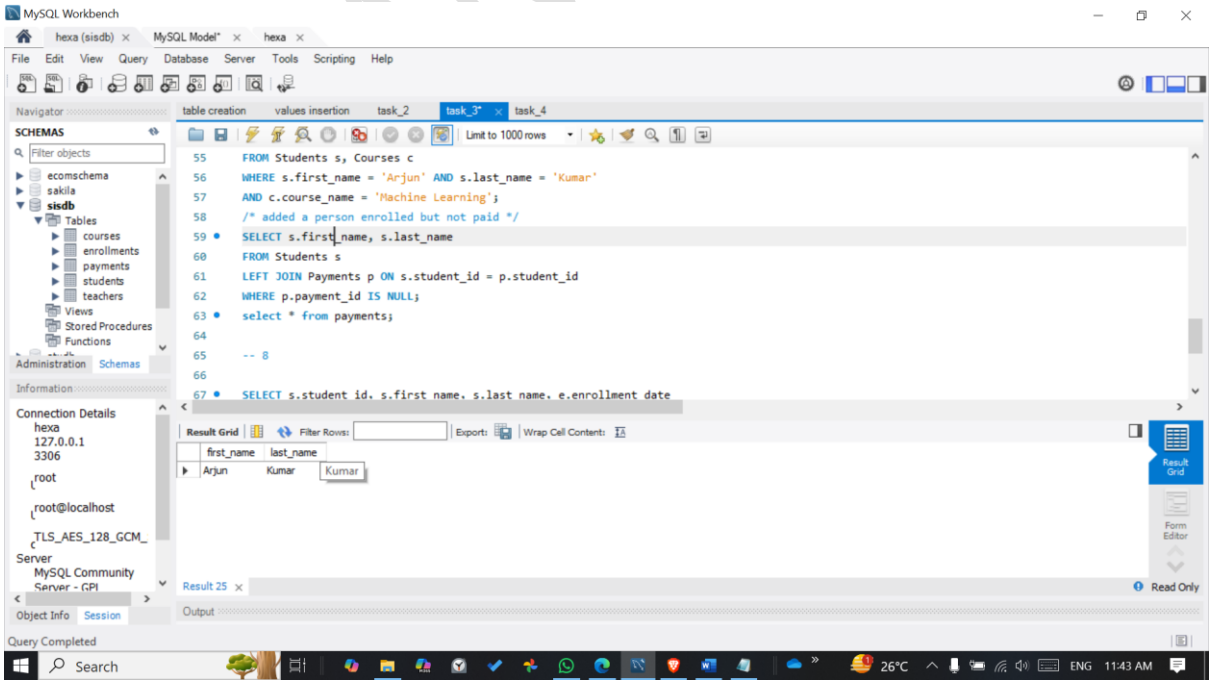
3.6

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.



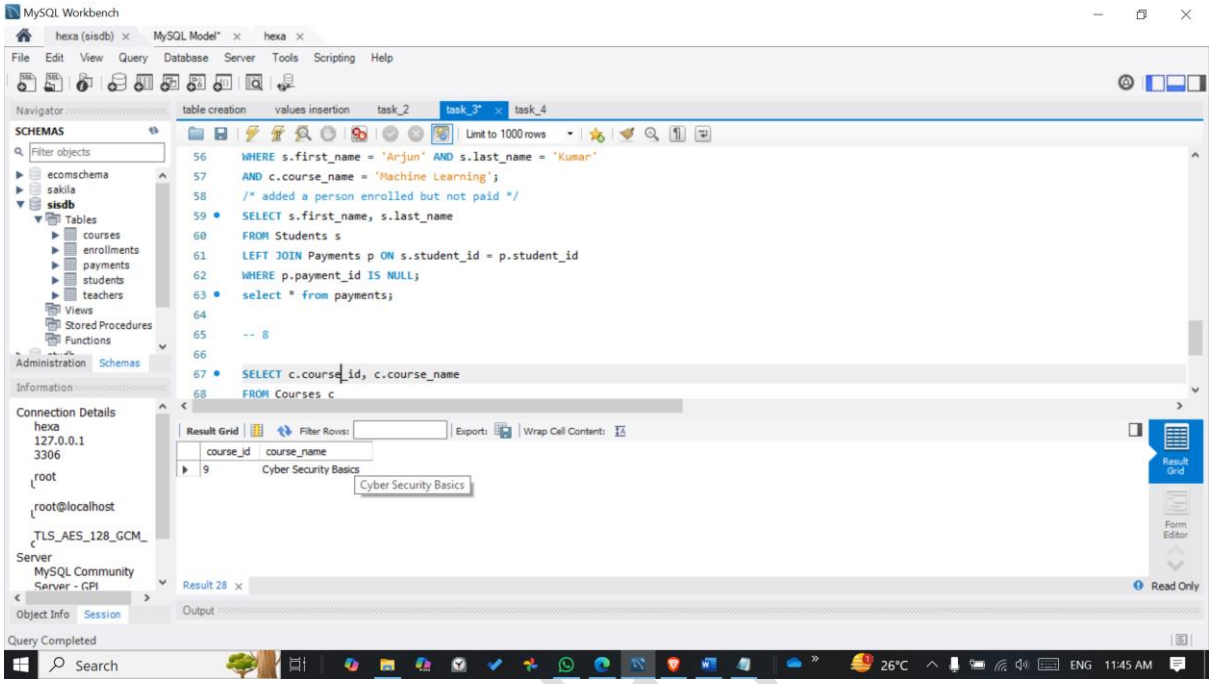
3.7

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.



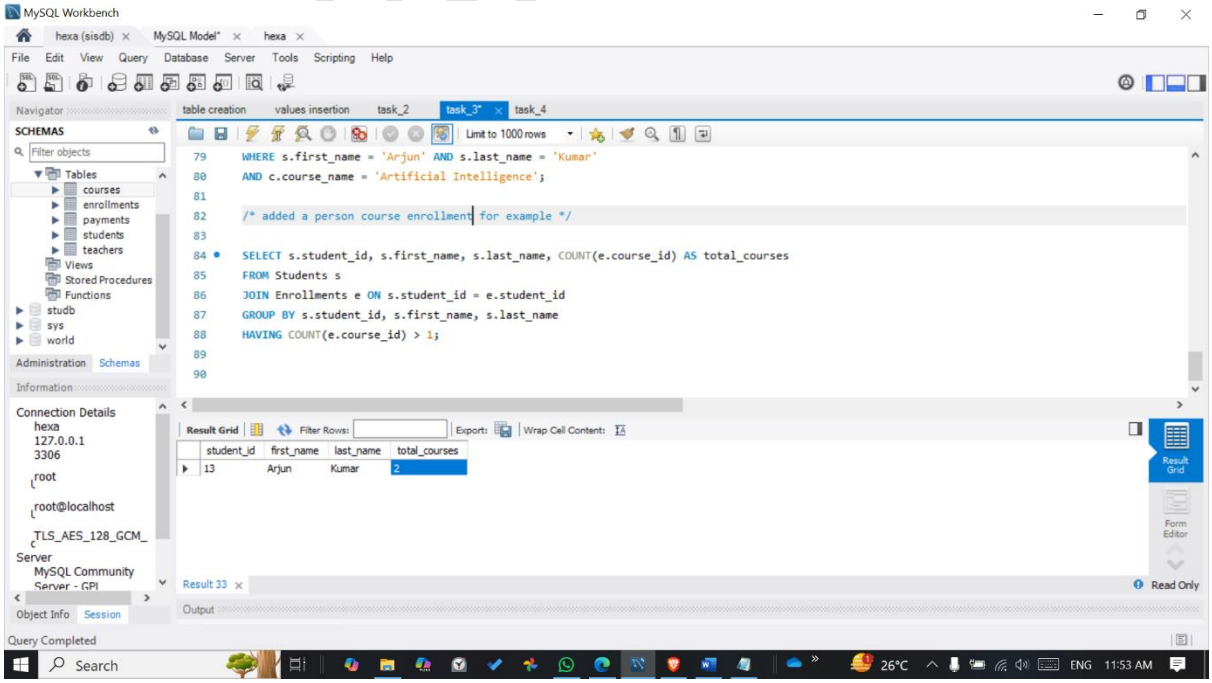
3.8

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.



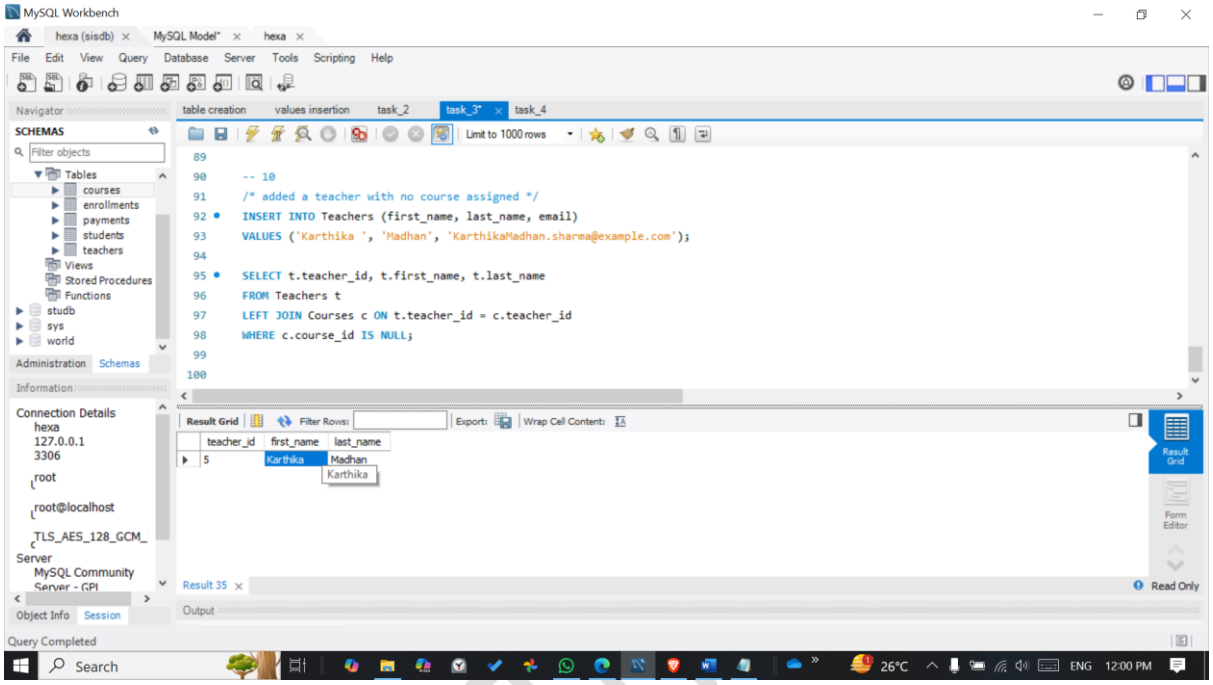
3.9

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.



3.10

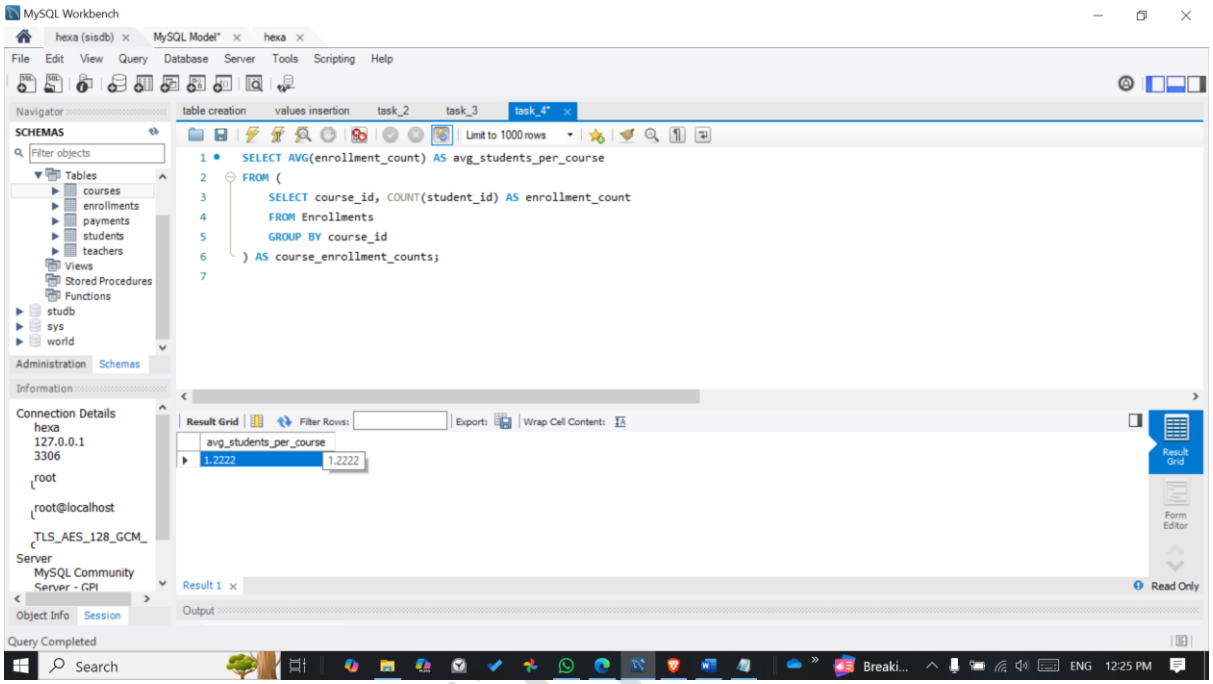
Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher" table and the "Courses" table and filter for teachers with NULL course assignments.



Task 4. Subquery and its type:

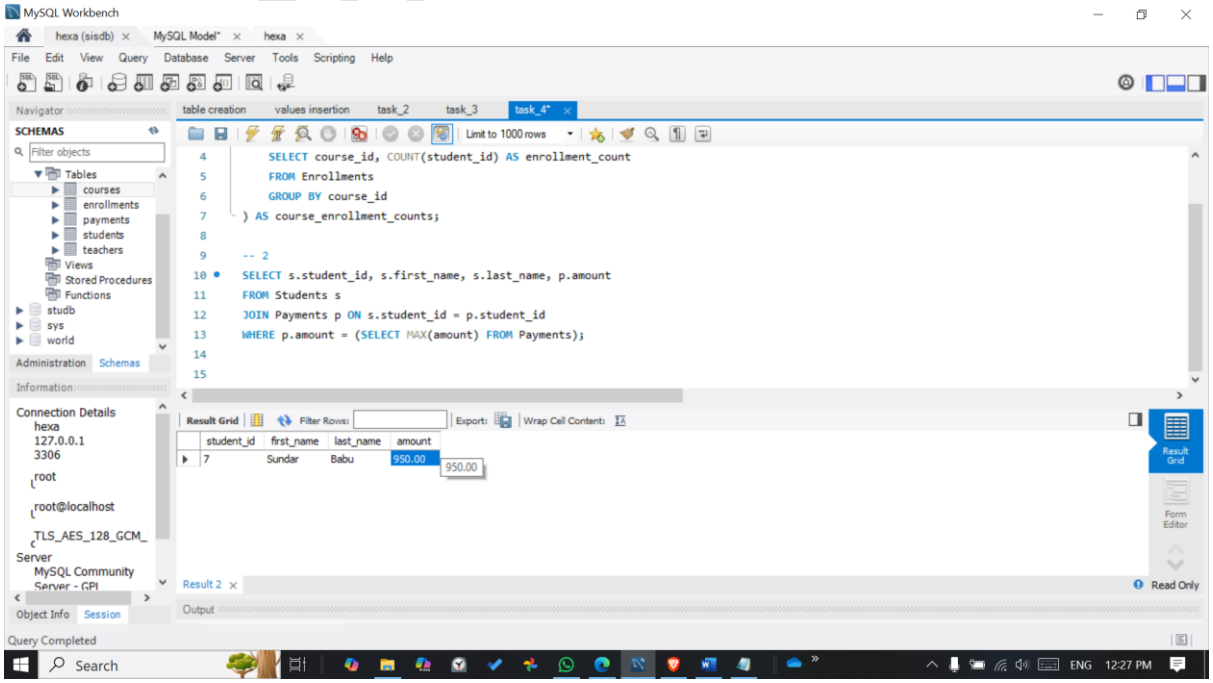
4.1.

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



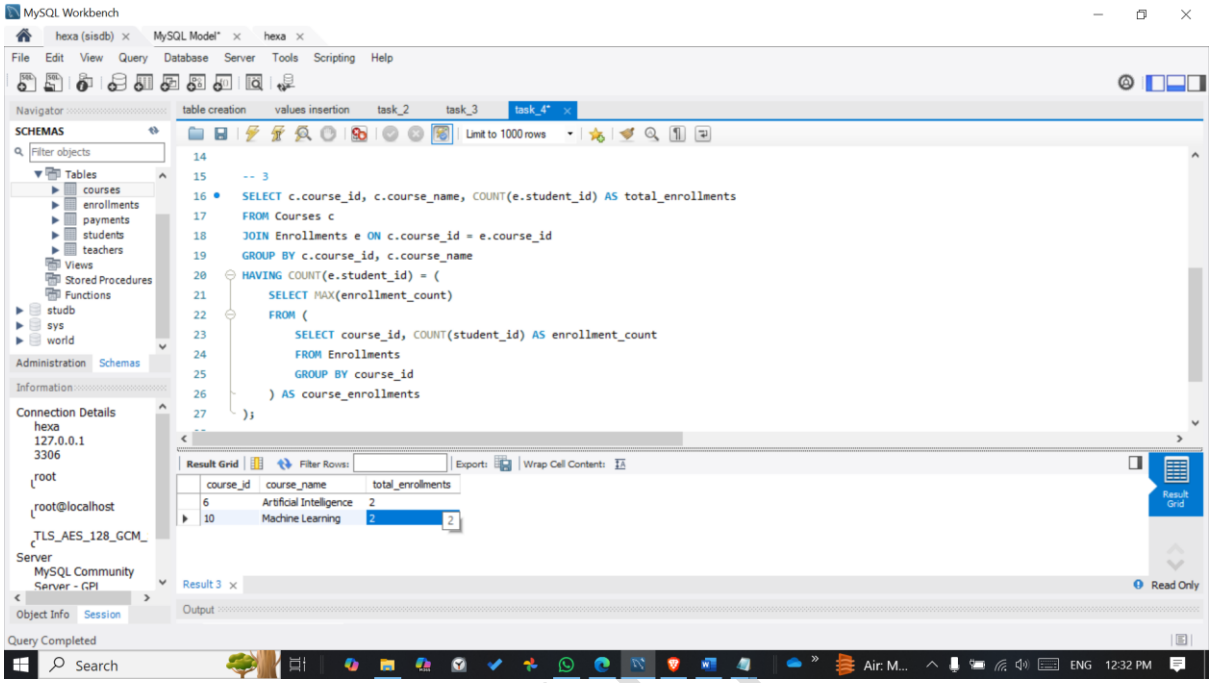
4.2

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.



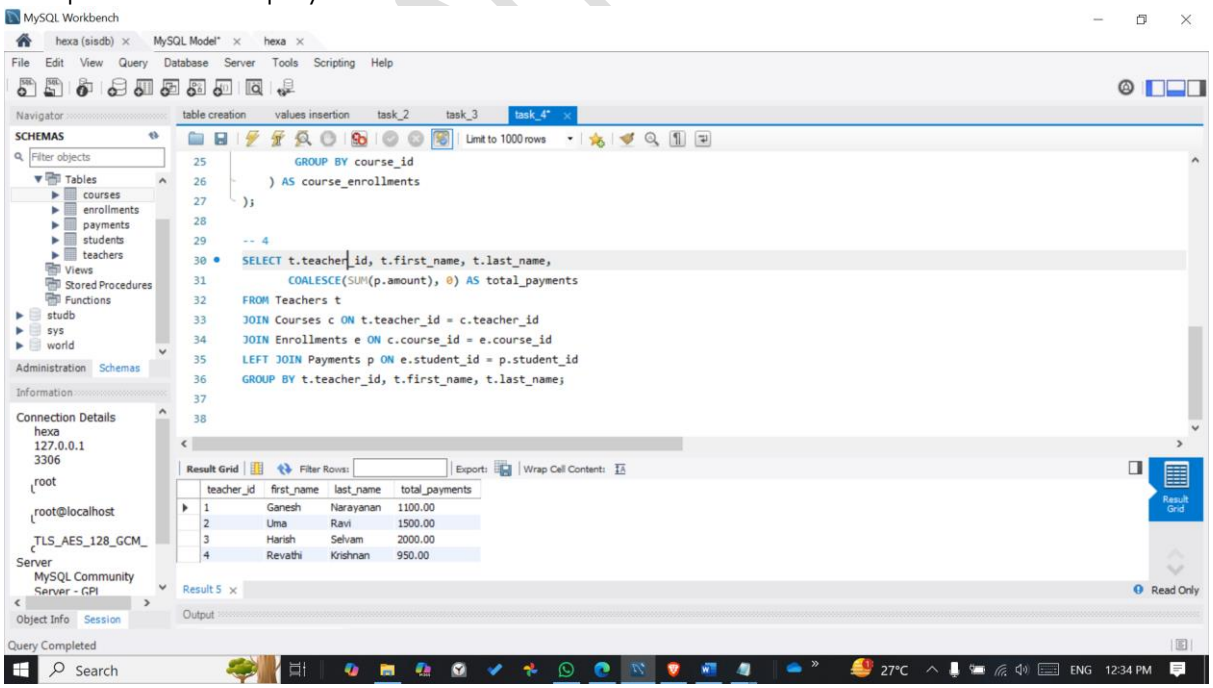
4.3.

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



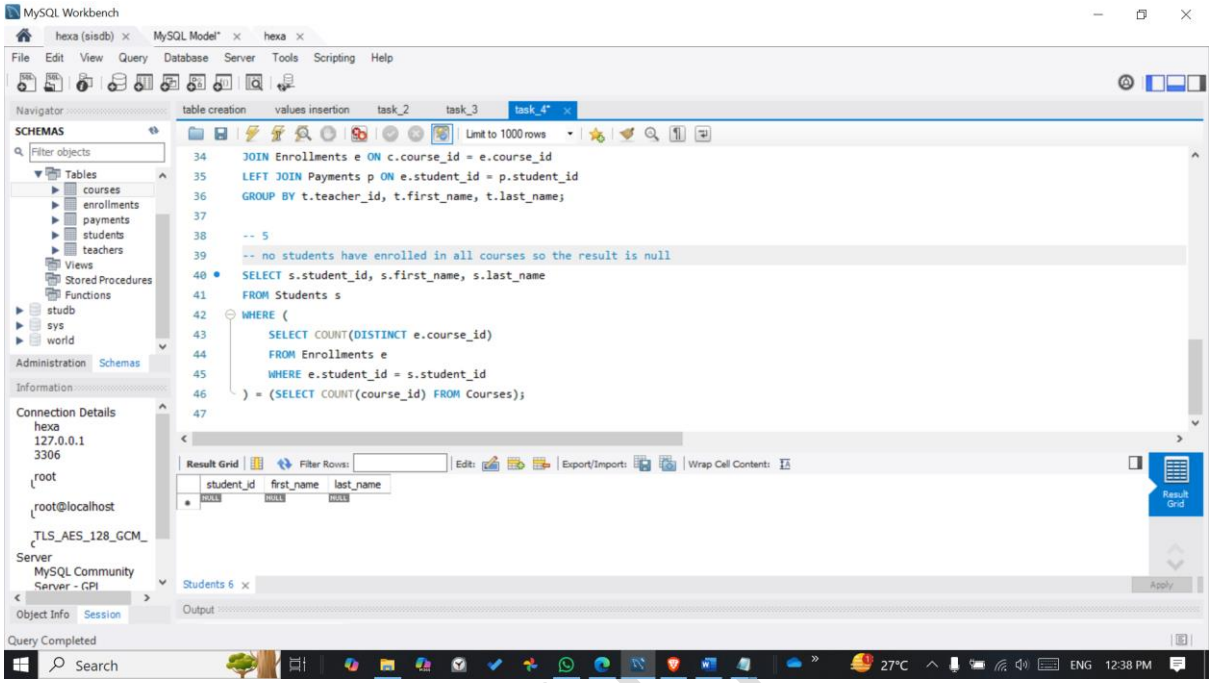
4.4

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



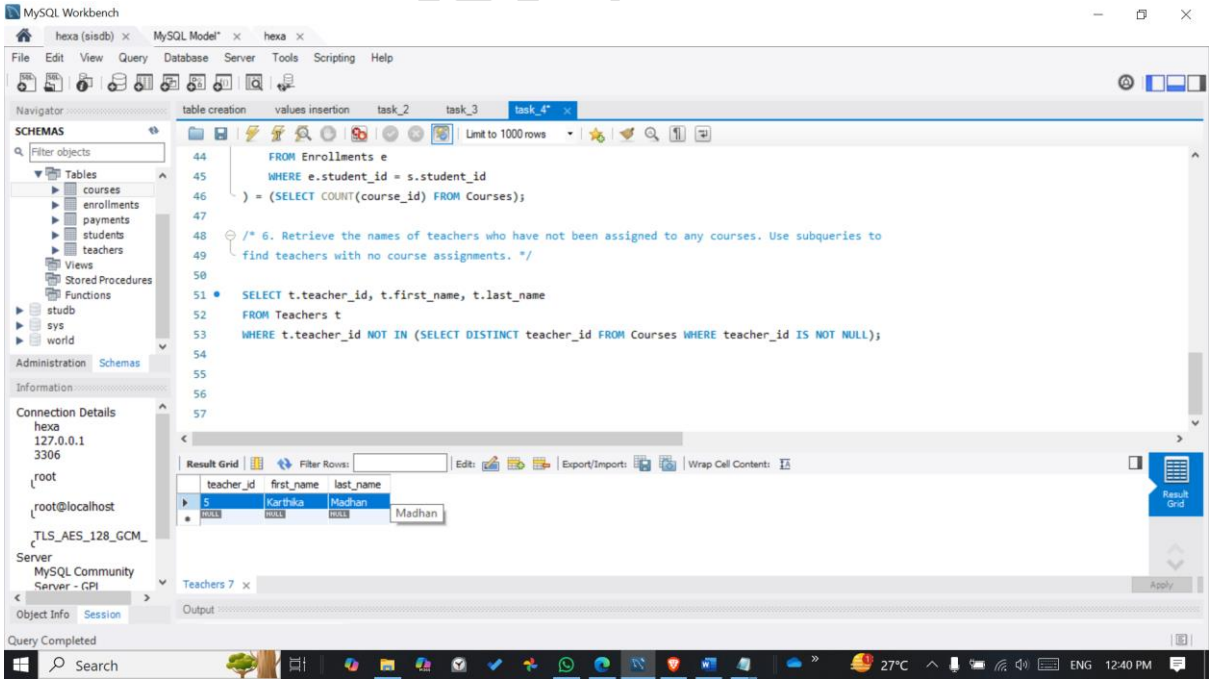
4.5

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



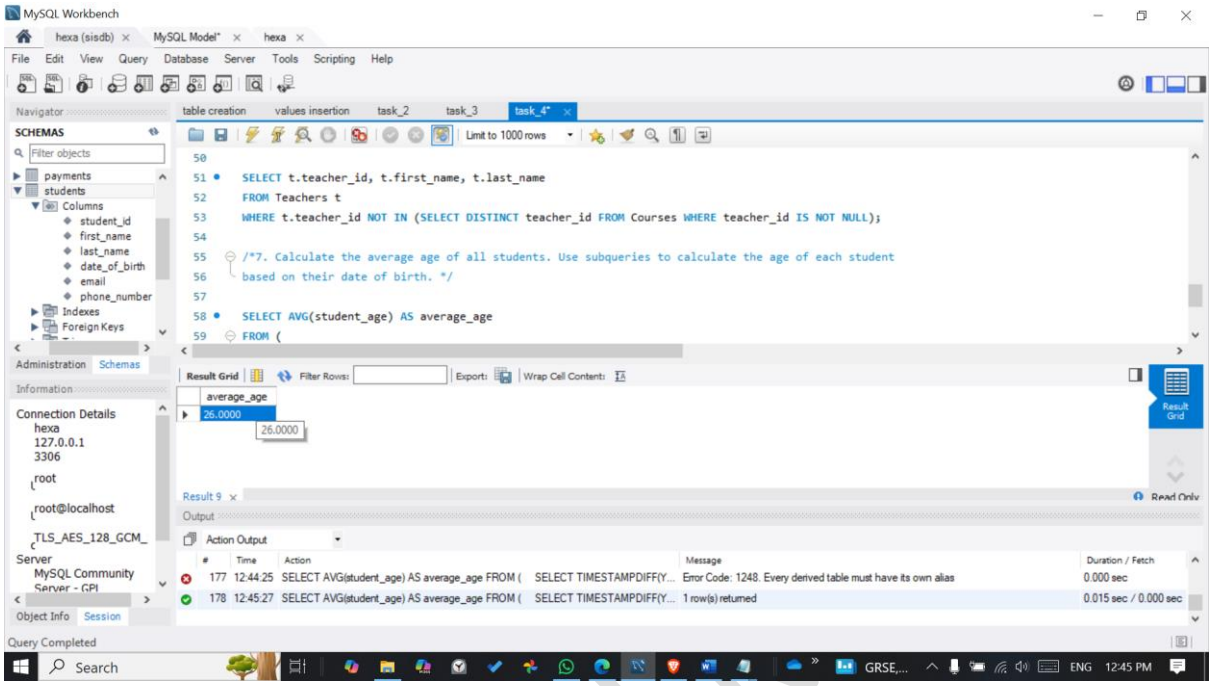
4.6

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



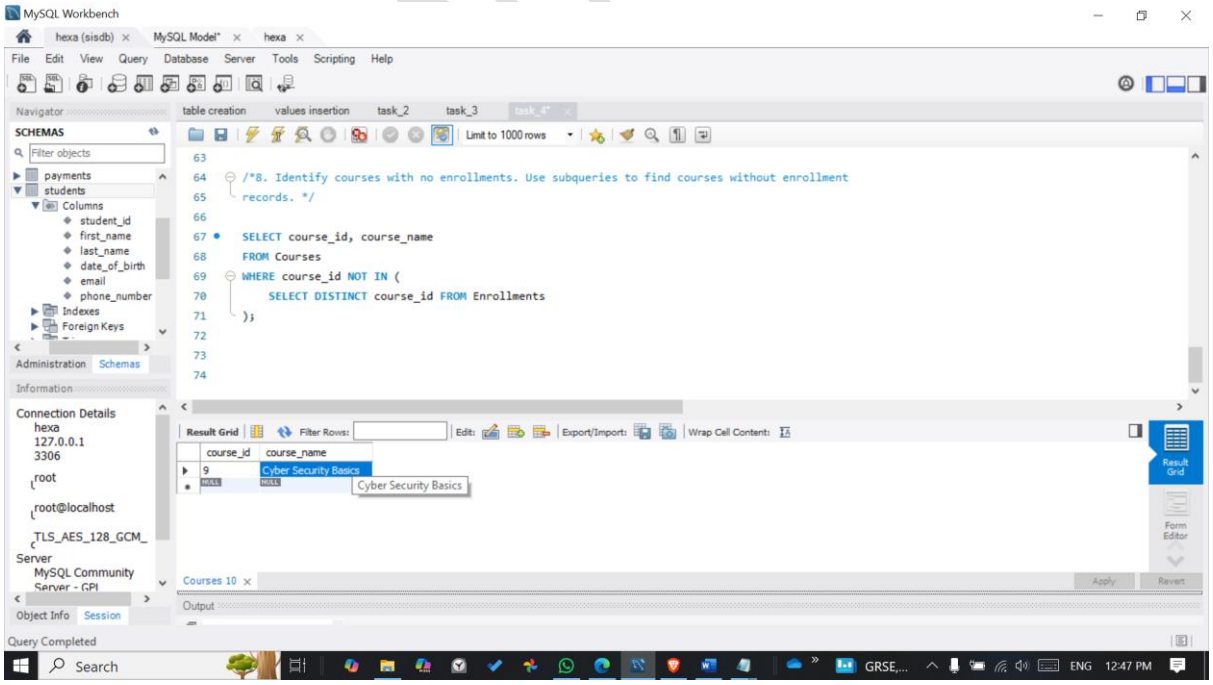
4.7

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



4.8

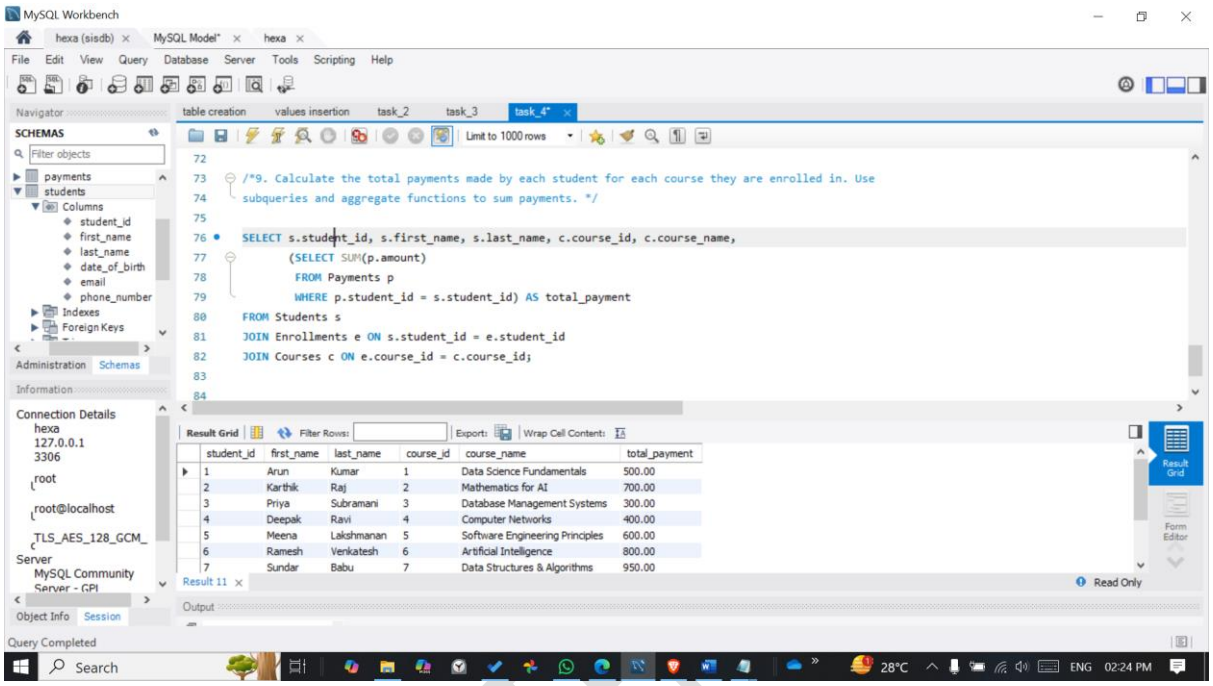
Identify courses with no enrollments. Use subqueries to find courses without enrollment records.



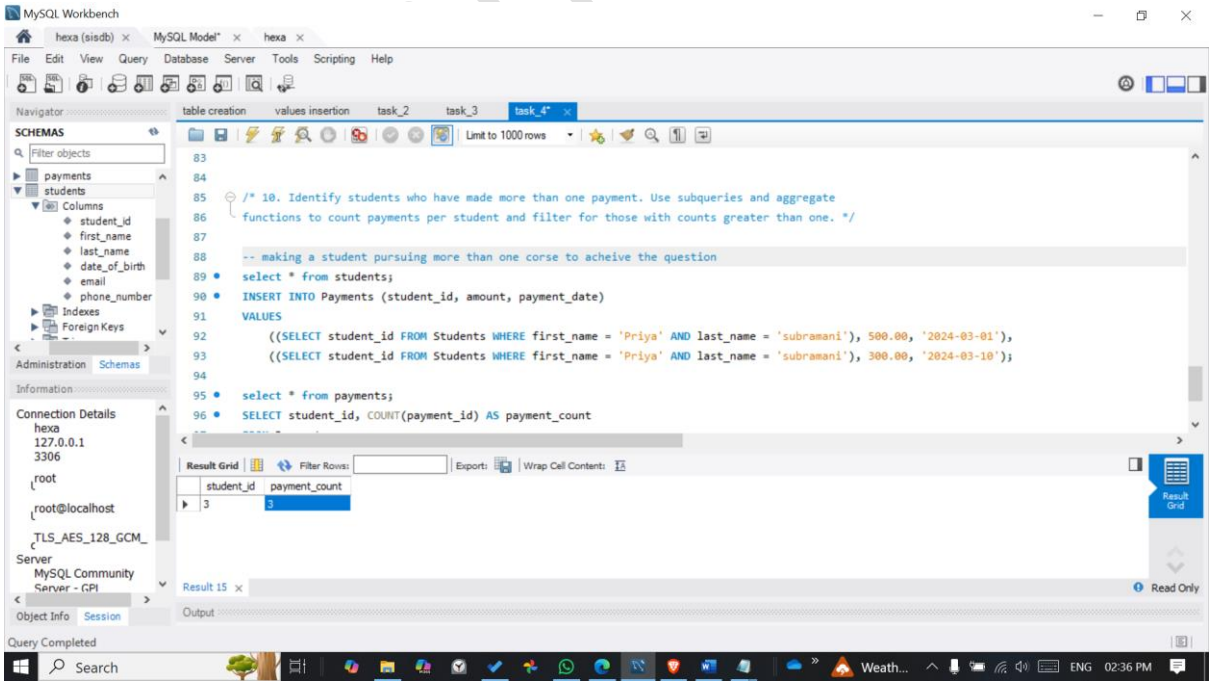
4.9

Calculate the total payments made by each student for each course they are enrolled in. Use

subqueries and aggregate functions to sum payments.

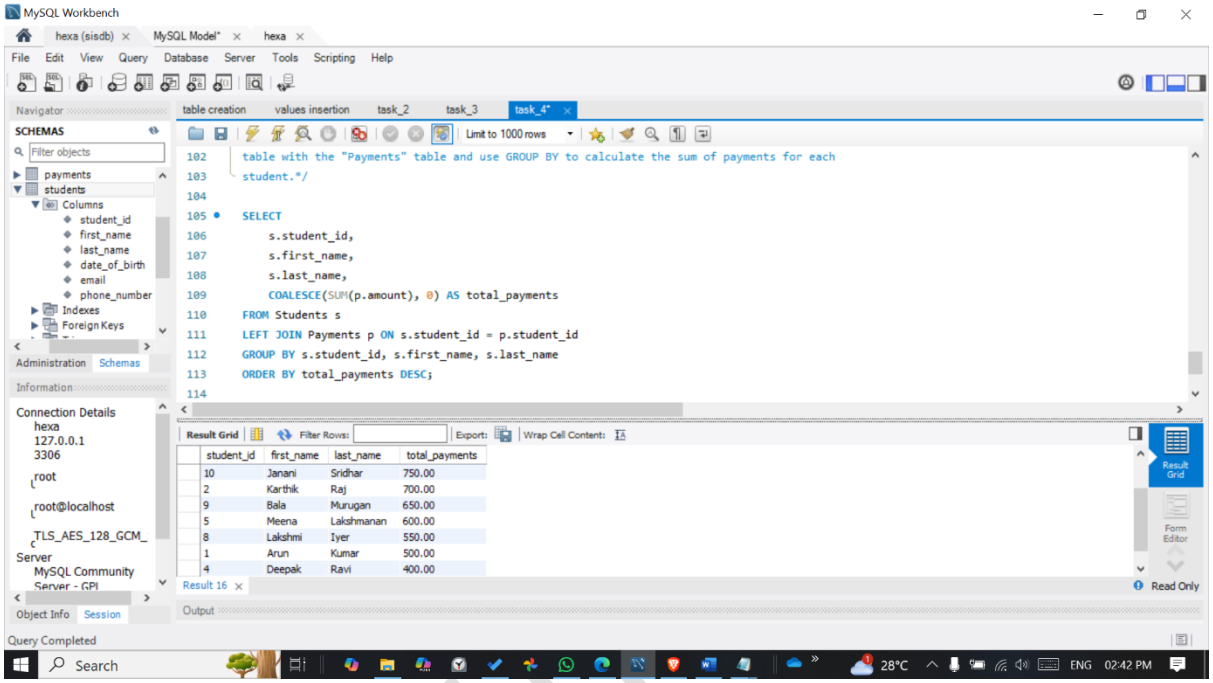


4.10



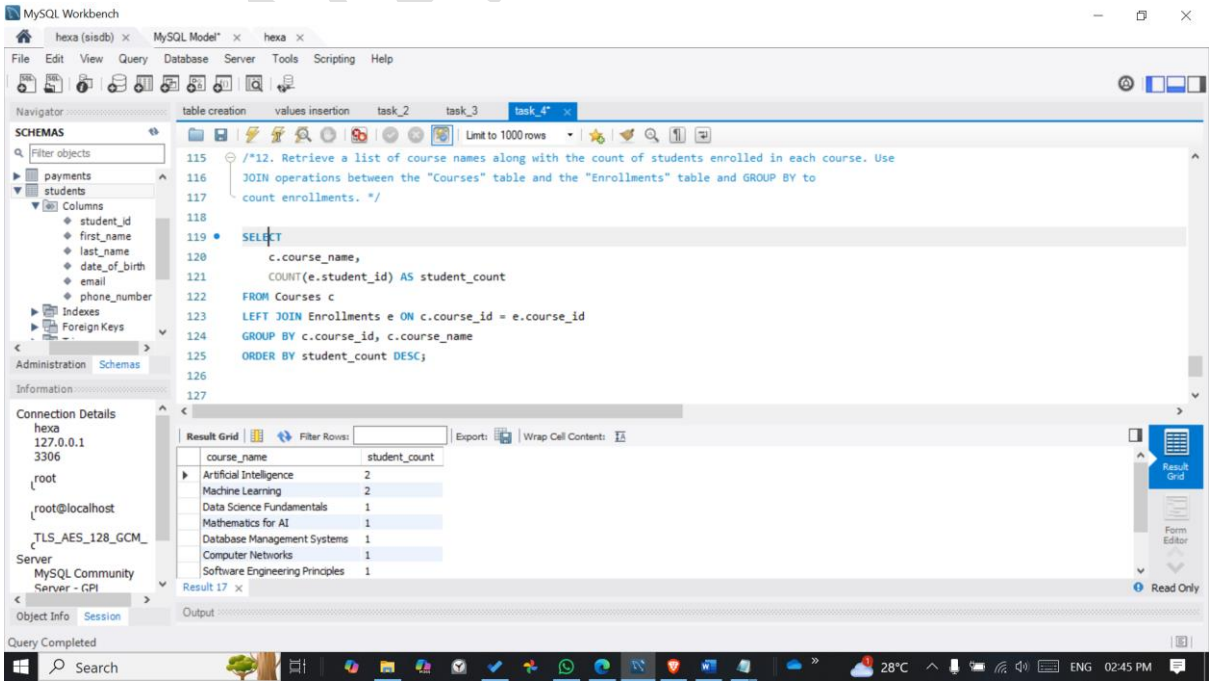
4.11

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.



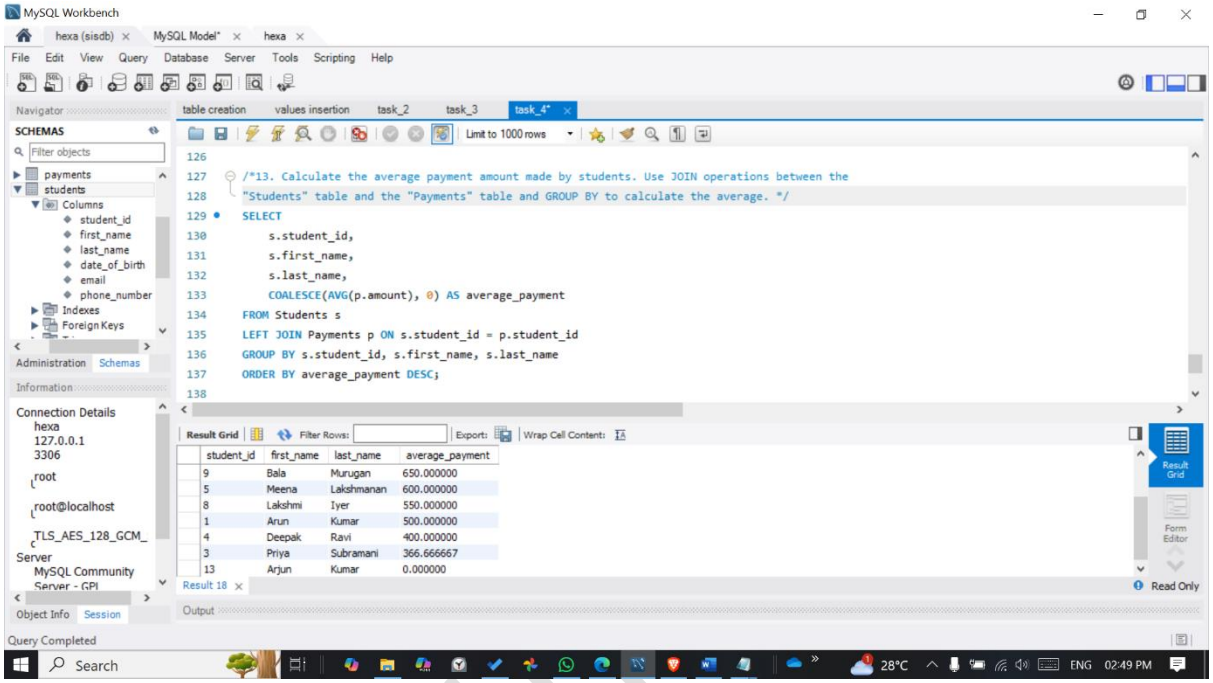
4.12

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.



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.



END OF THE DOCUMENT