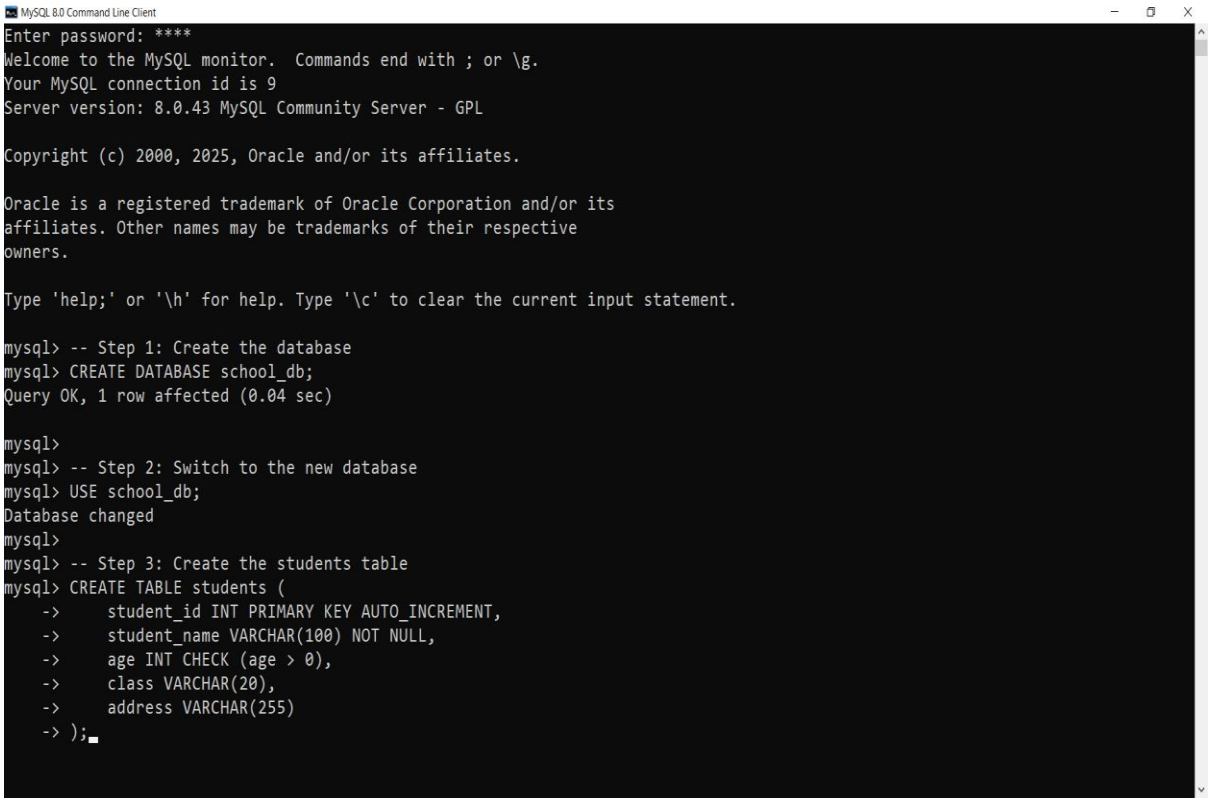
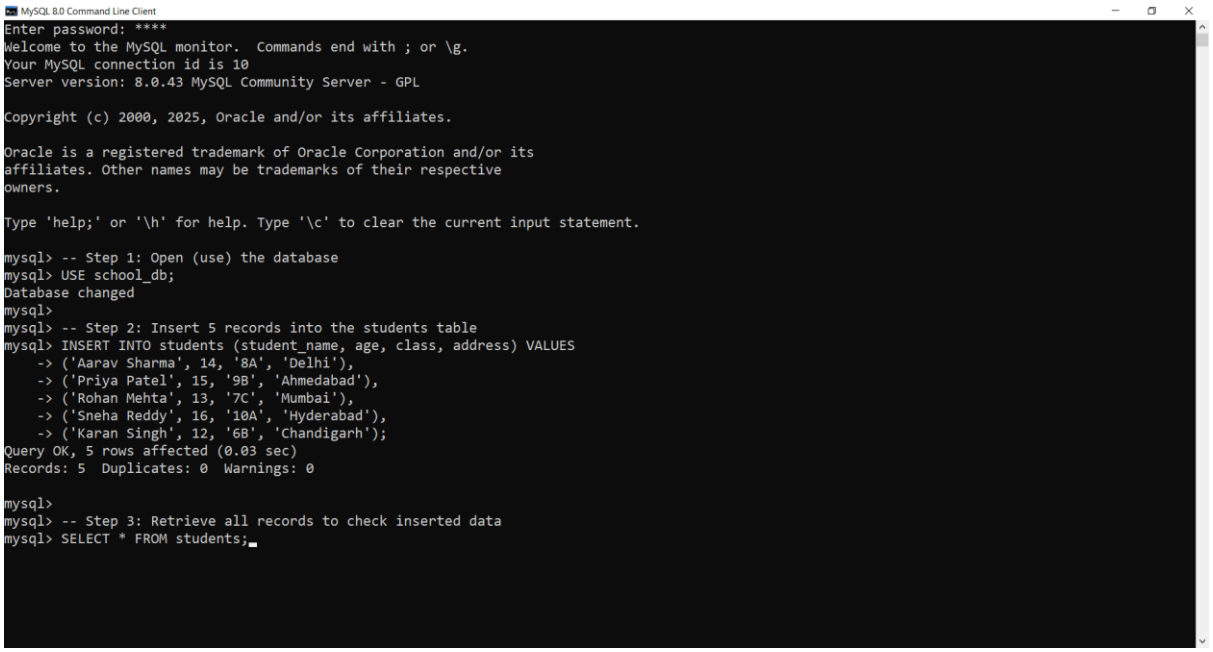
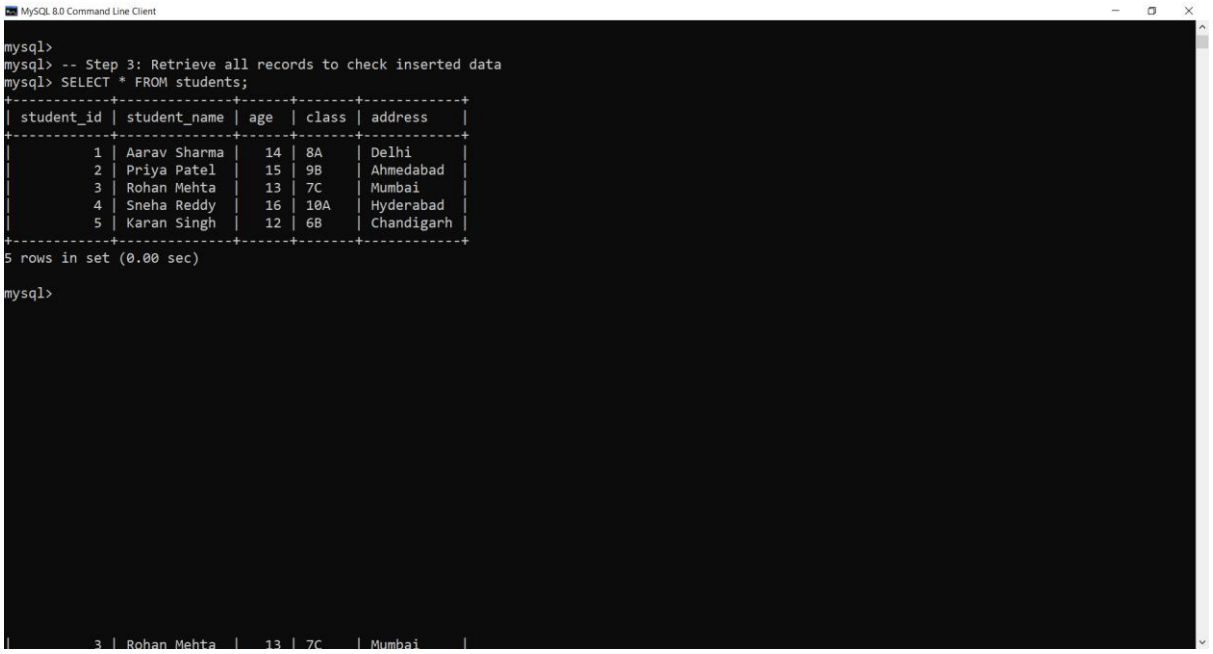
MODULE-4 DBMS

1.Create a new database named school_db and a table called students with the following columns: student_id, student_name, age, class, and address.



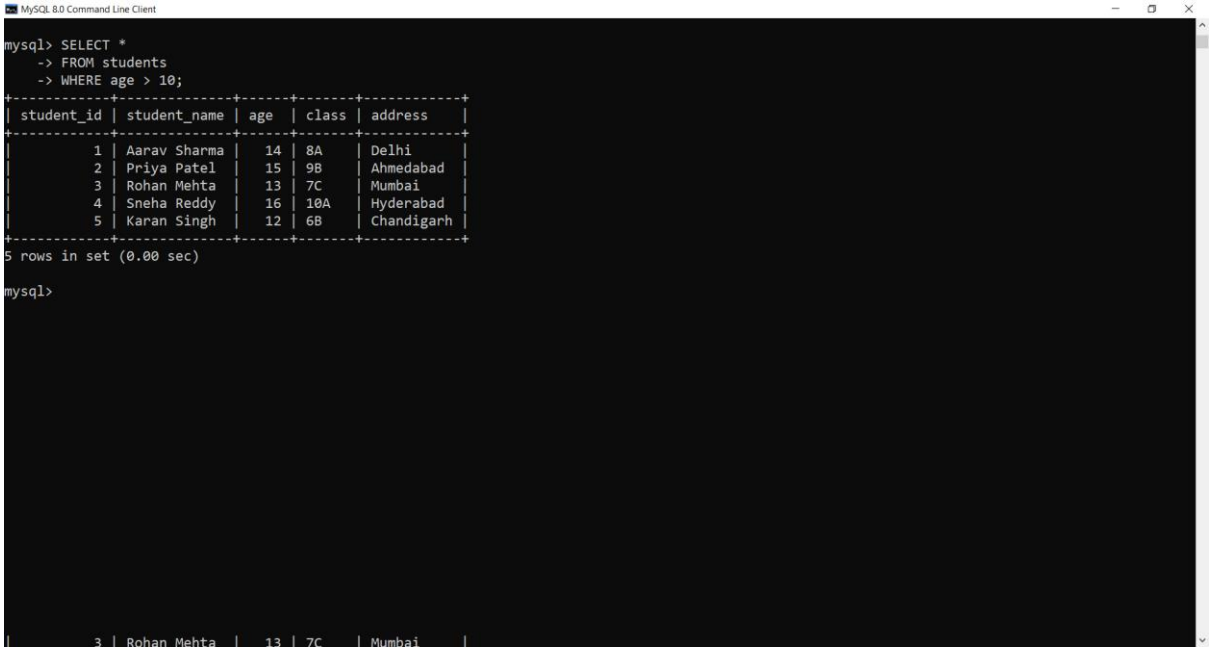
2. Insert five records into the students table and retrieve all records using the SELECT statement.



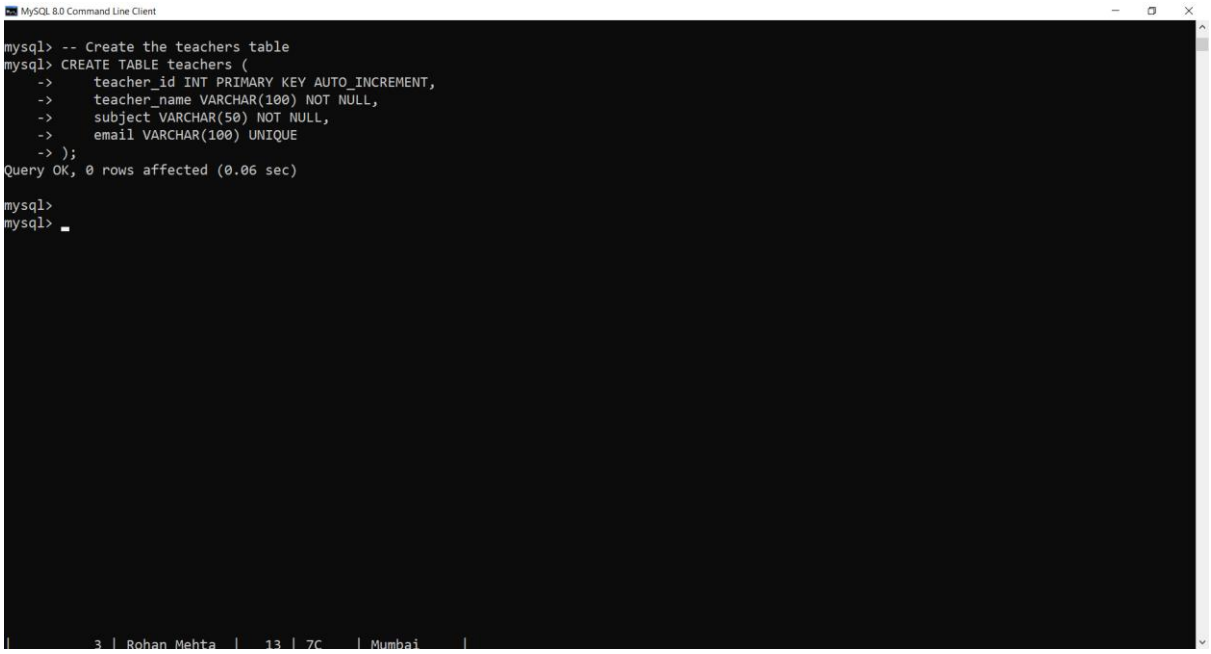


3.Write SQL queries to retrieve specific columns (student_name and age) from the students table.

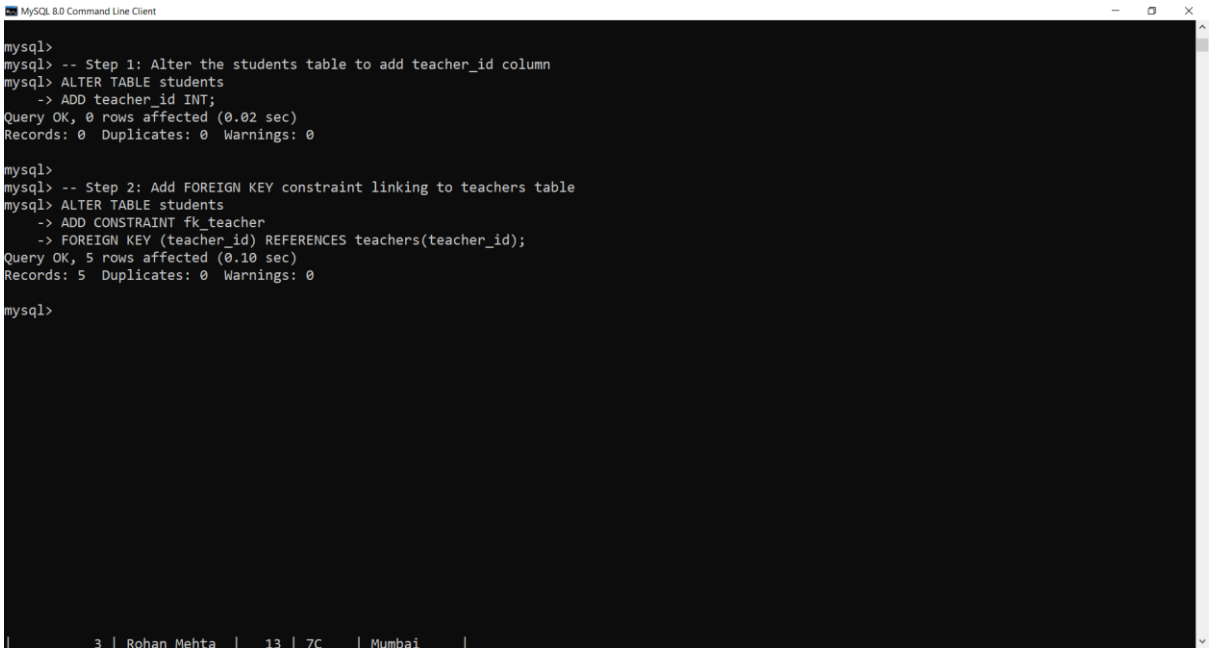
4. Write SQL queries to retrieve all students whose age is greater than 10.



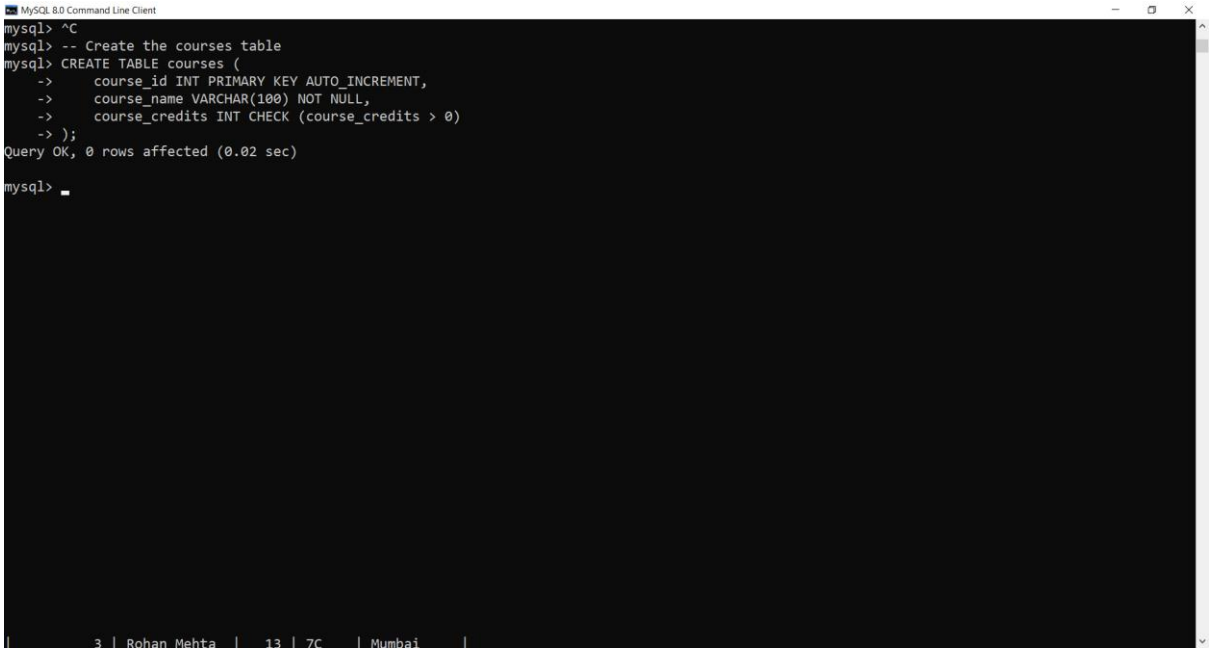
5. Create a table teachers with the following columns: teacher_id (Primary Key),teacher_name (NOT NULL), subject (NOT NULL), and email (UNIQUE).



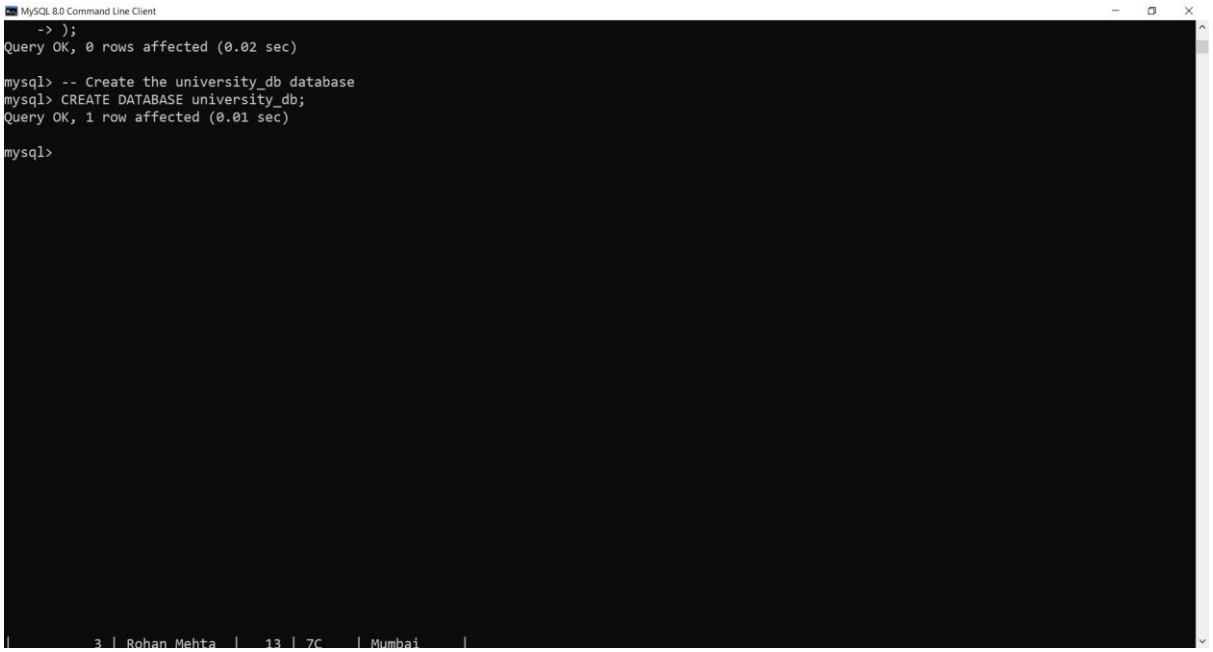
6. : Implement a FOREIGN KEY constraint to relate the teacher_id from the teachers table with the students table.

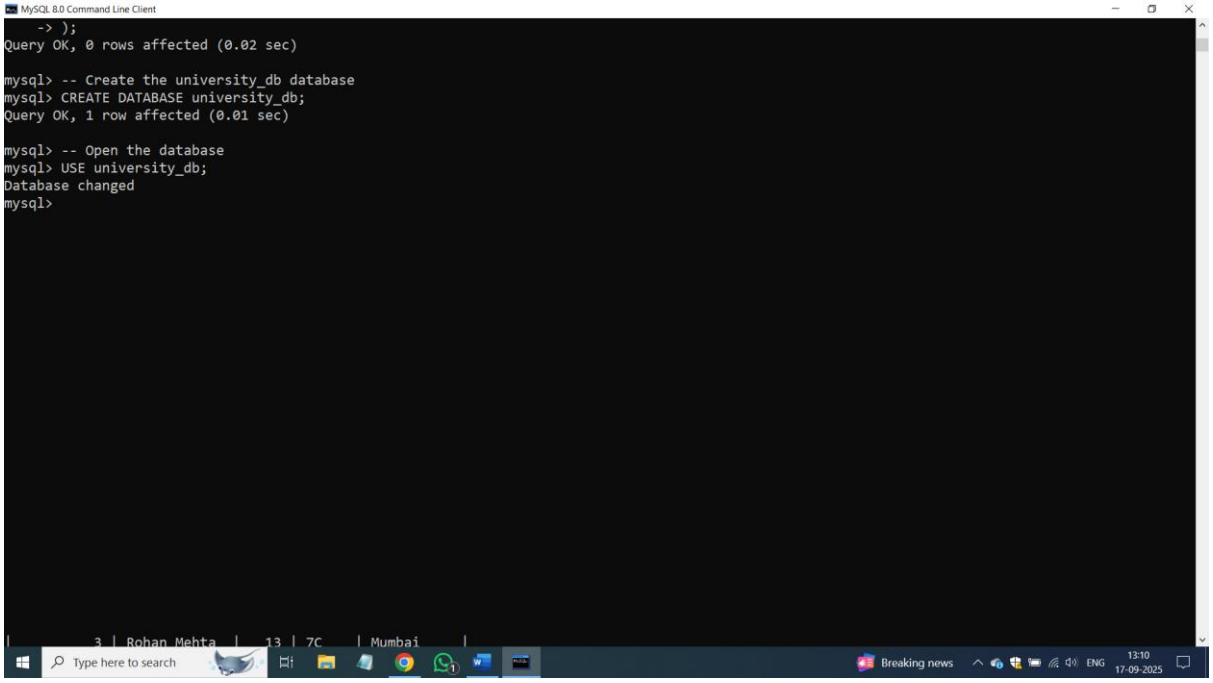


7. : Create a table courses with columns: course_id, course_name, and course_credits. Set the course_id as the primary key.

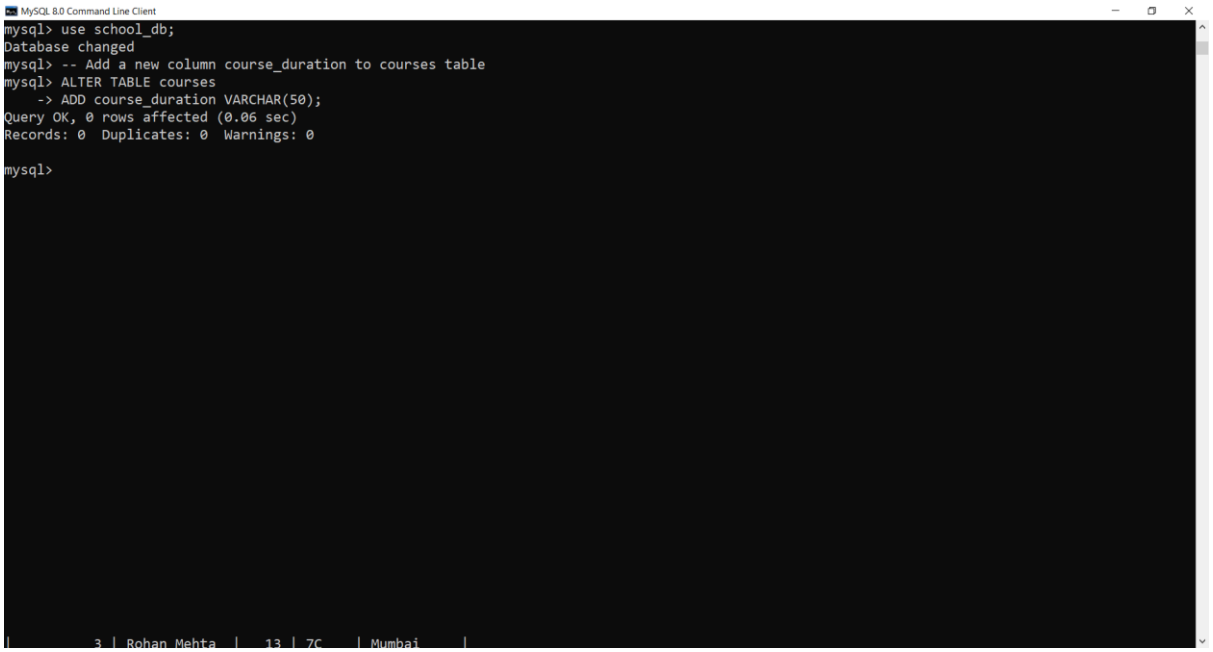


8. Use the CREATE command to create a database university_db.

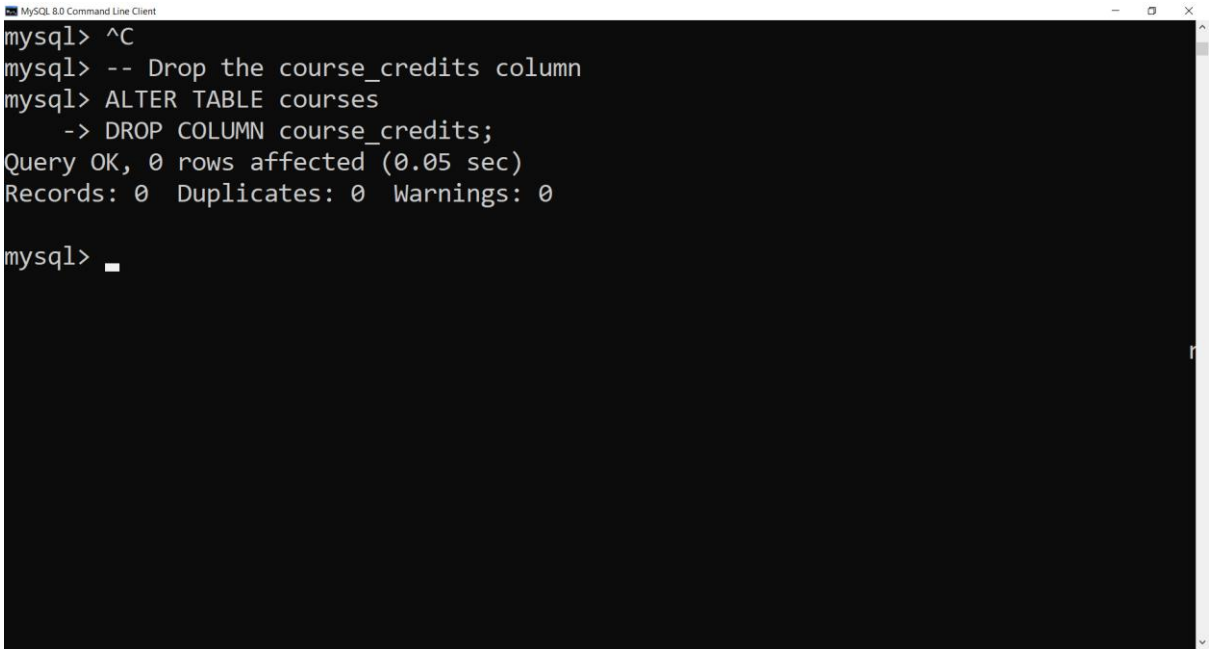




9. : Modify the courses table by adding a column course_duration using the ALTER command.

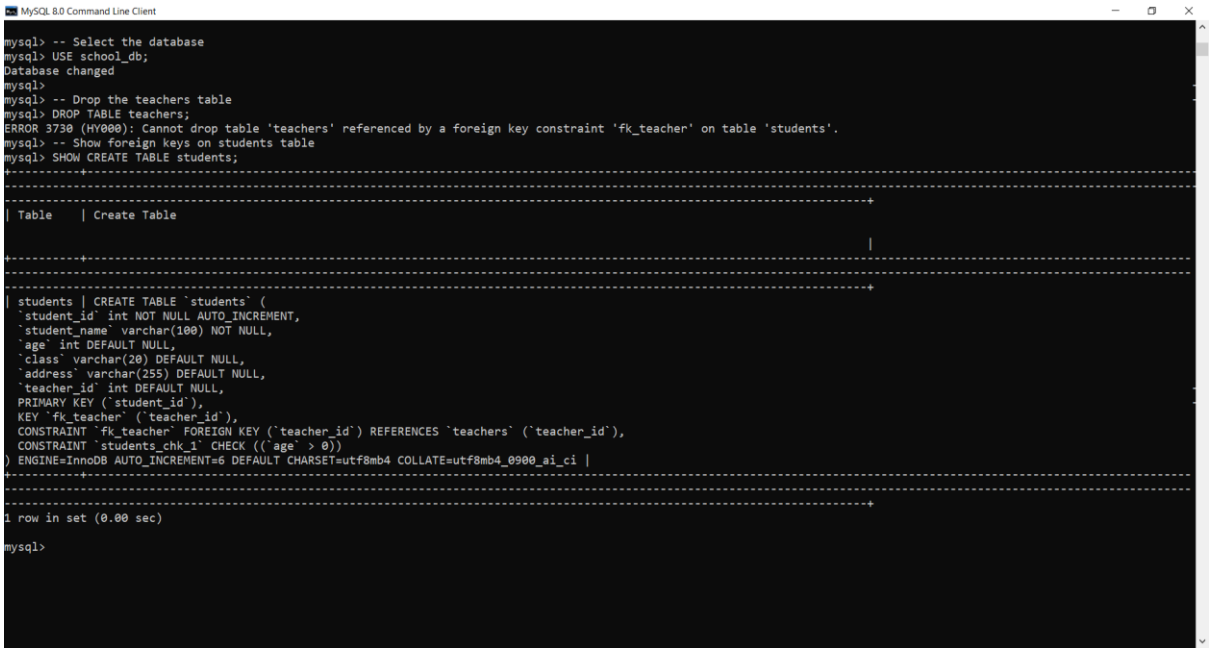


10. Drop the course_credits column from the courses table.

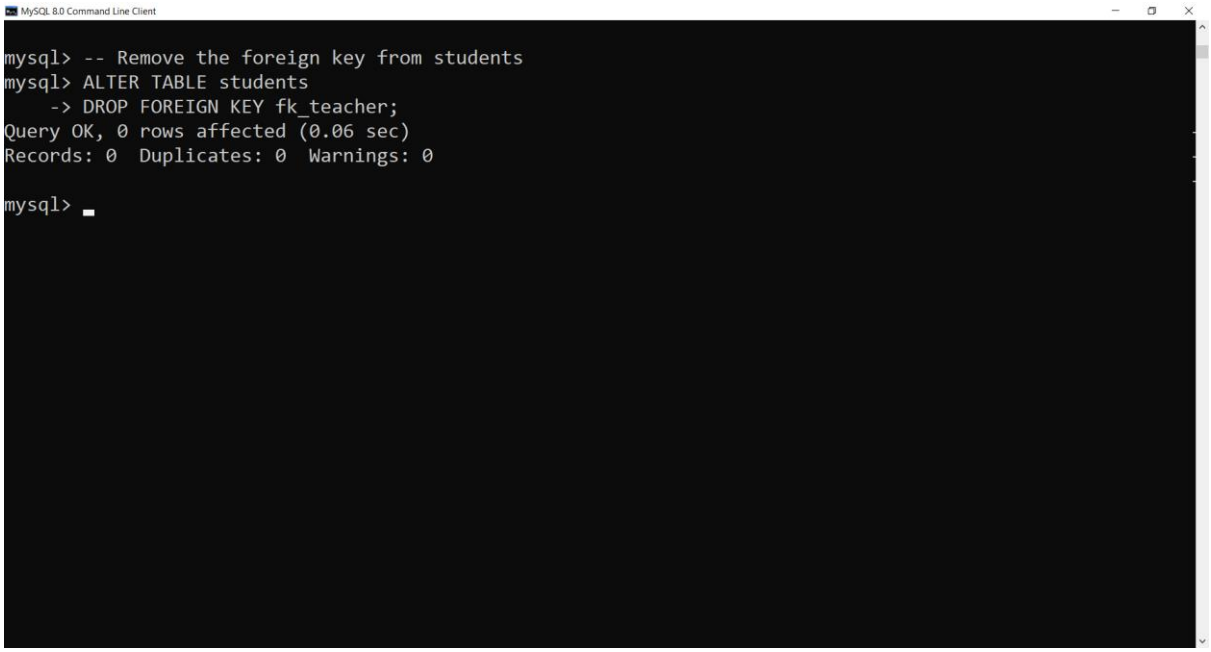


11. Drop the teachers table from the school_db database.

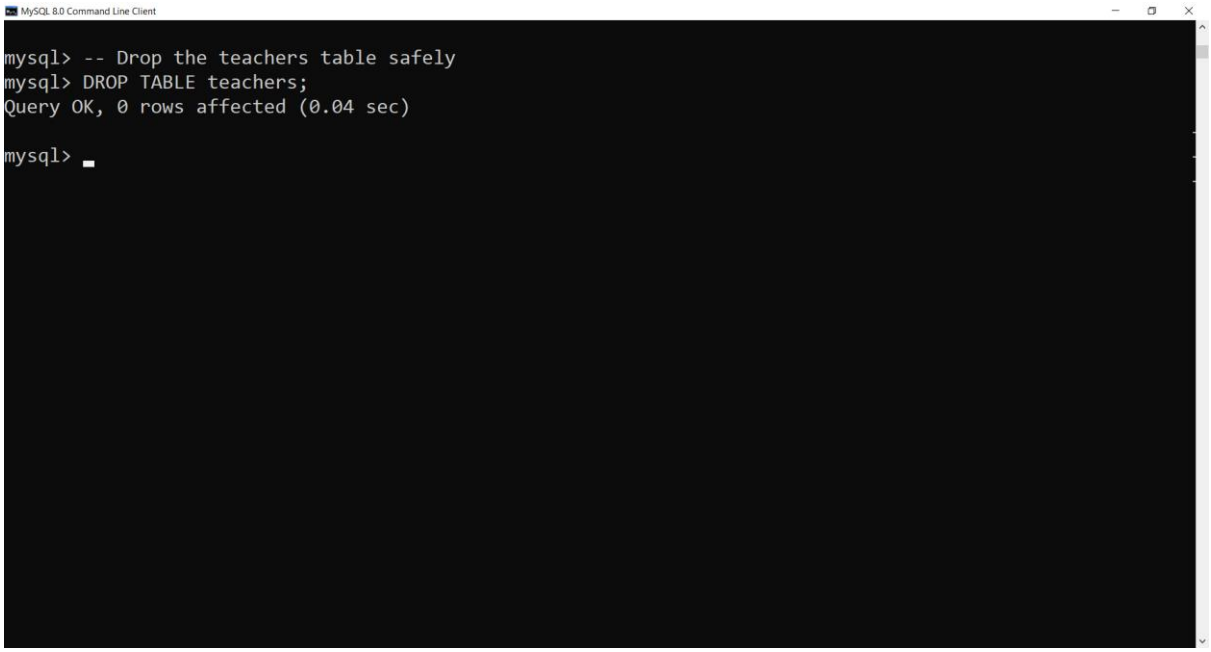
Step 1: Find the foreign key name:



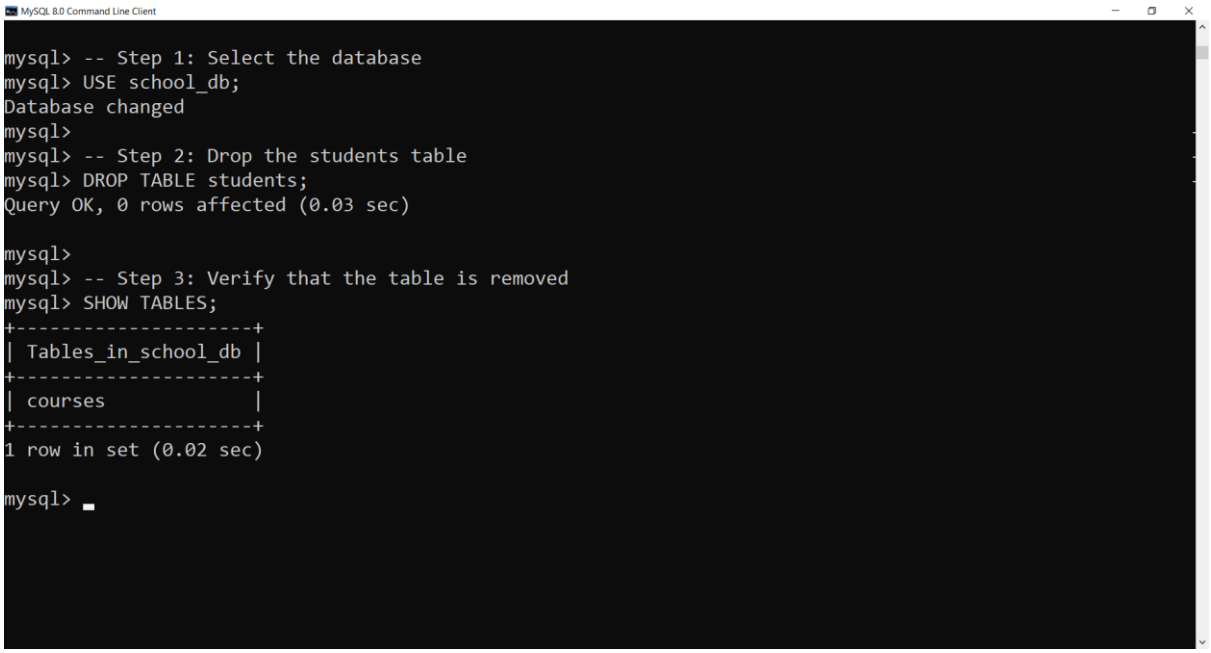
Step 2: Drop the foreign key constraint:



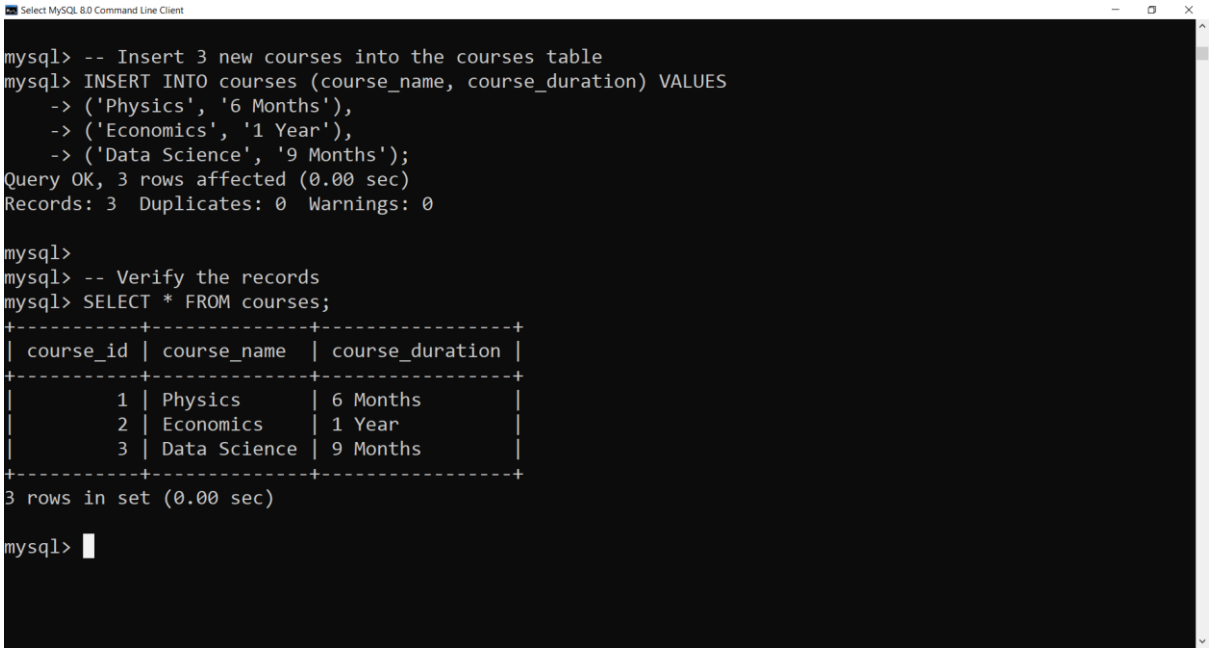
Step 3: Drop the teachers table:



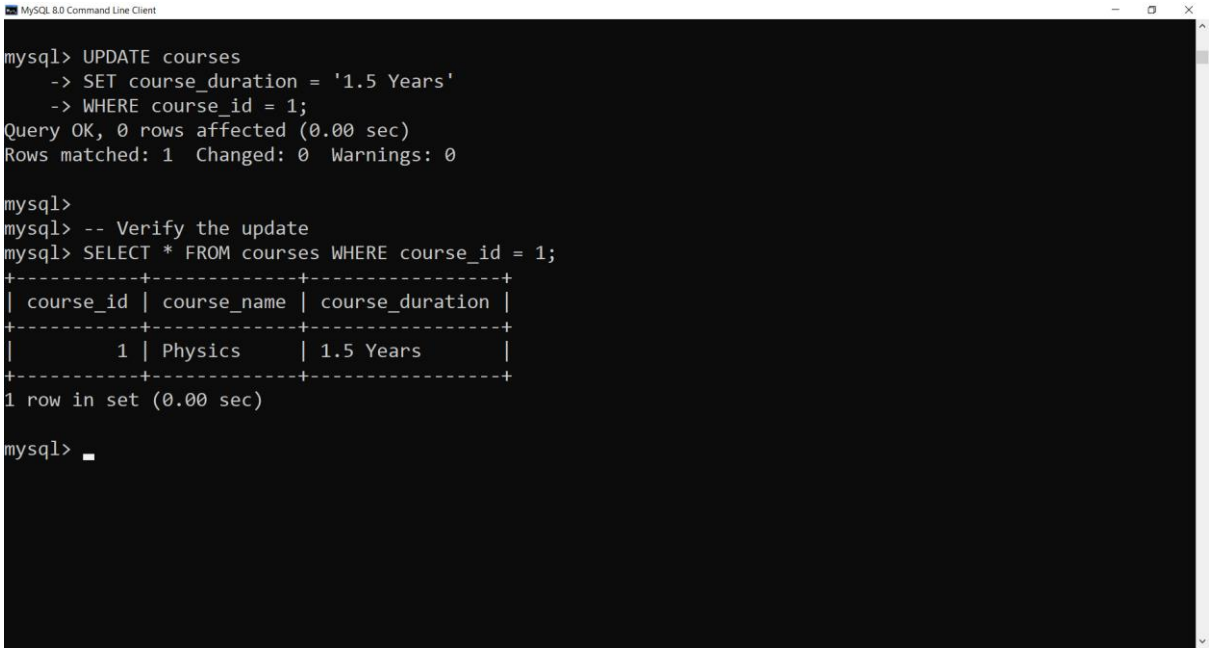
12. Drop the students table from the school_db database and verify that the table has been removed



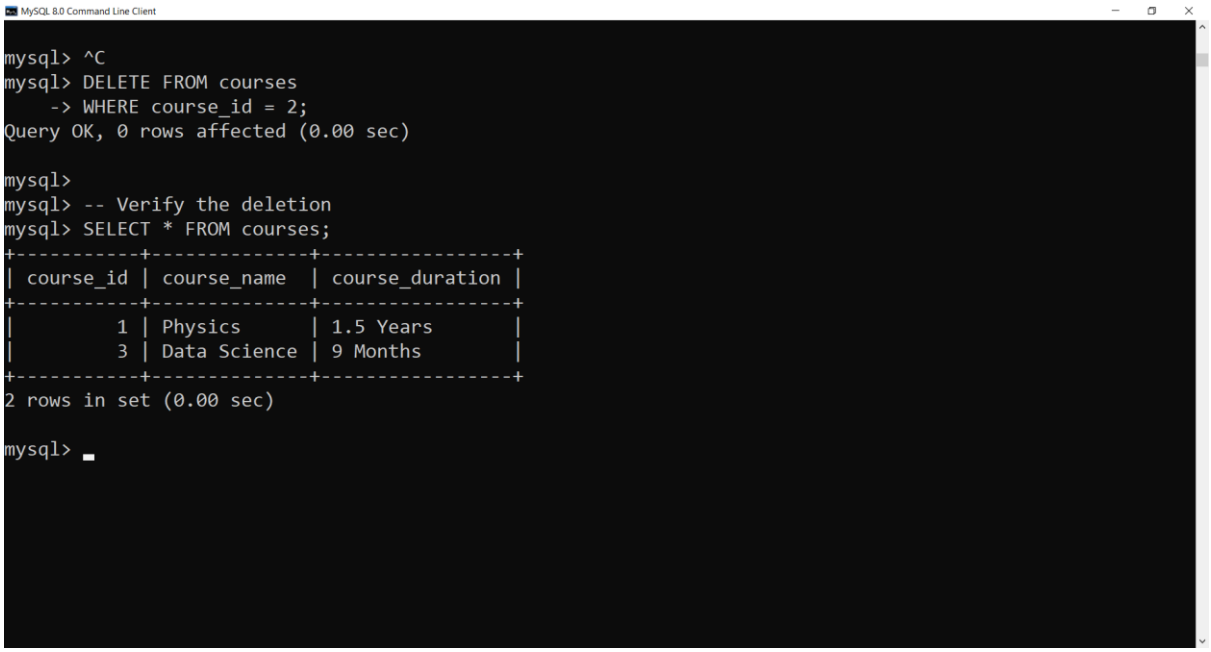
13. : Insert three records into the courses table using the INSERT command



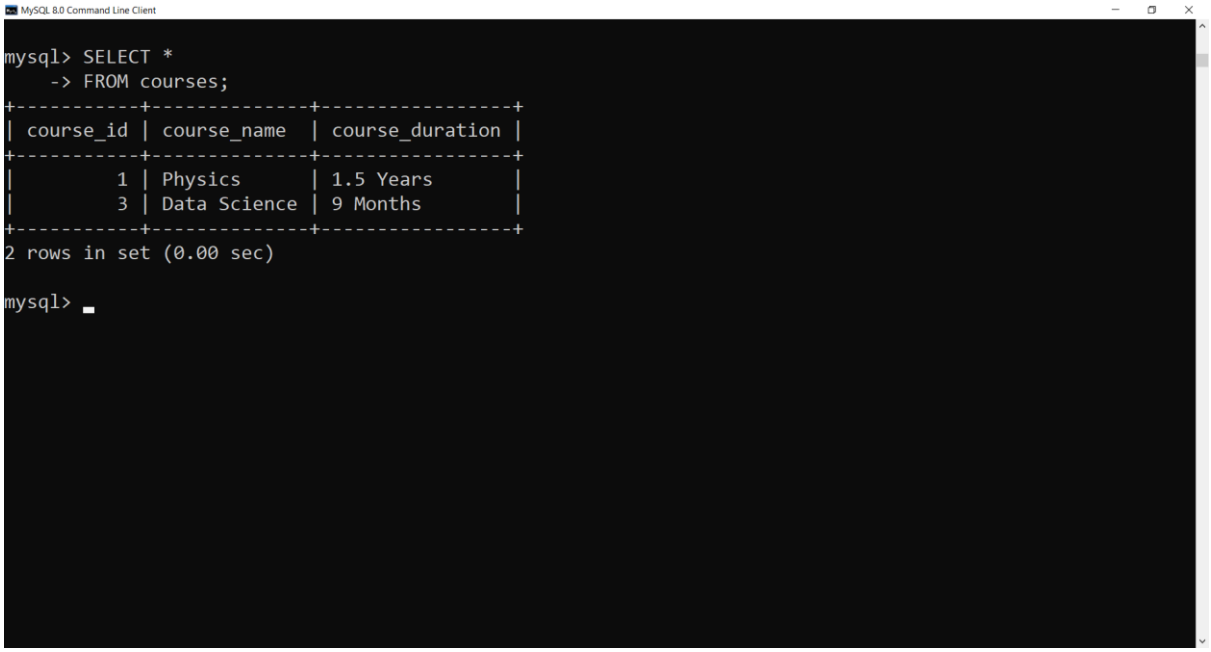
14. : Update the course duration of a specific course using the UPDATE command



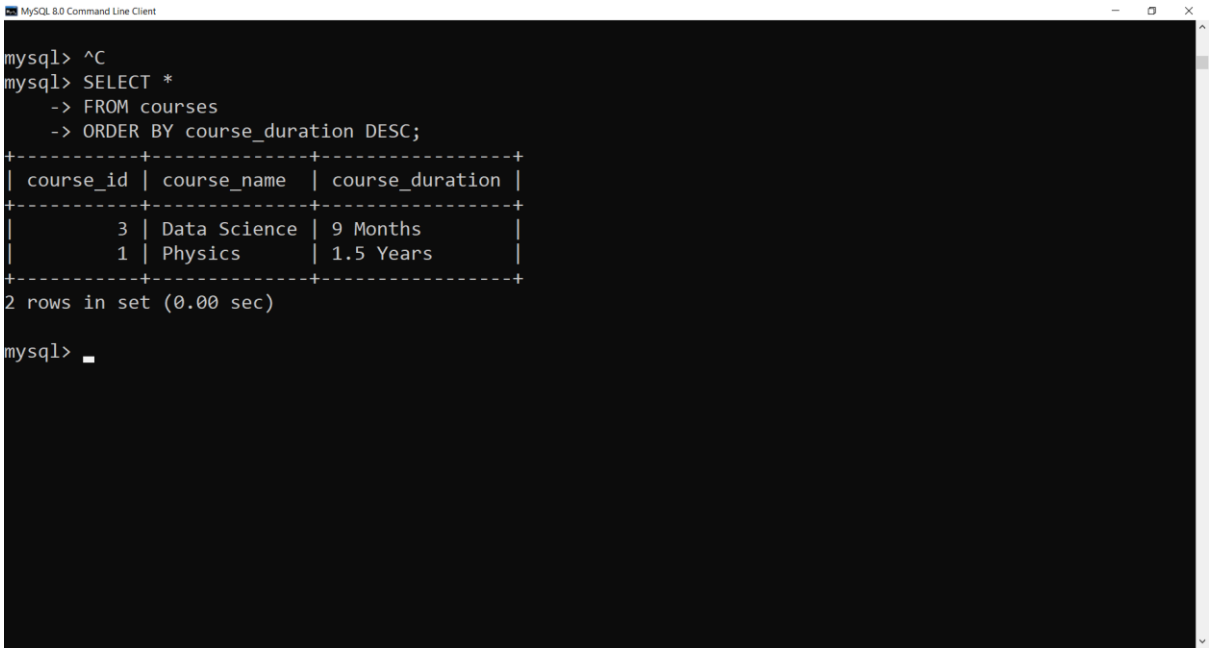
15. : Delete a course with a specific course_id from the courses table using the DELETE command.



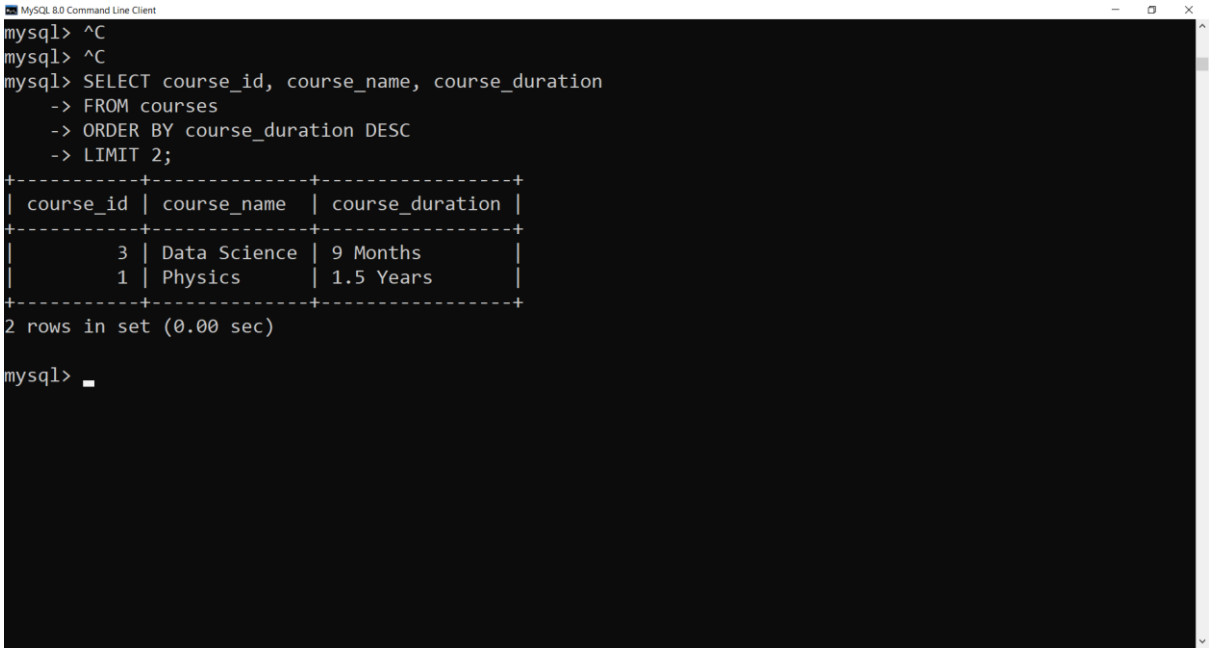
16. Retrieve all courses from the courses table using the SELECT statement.



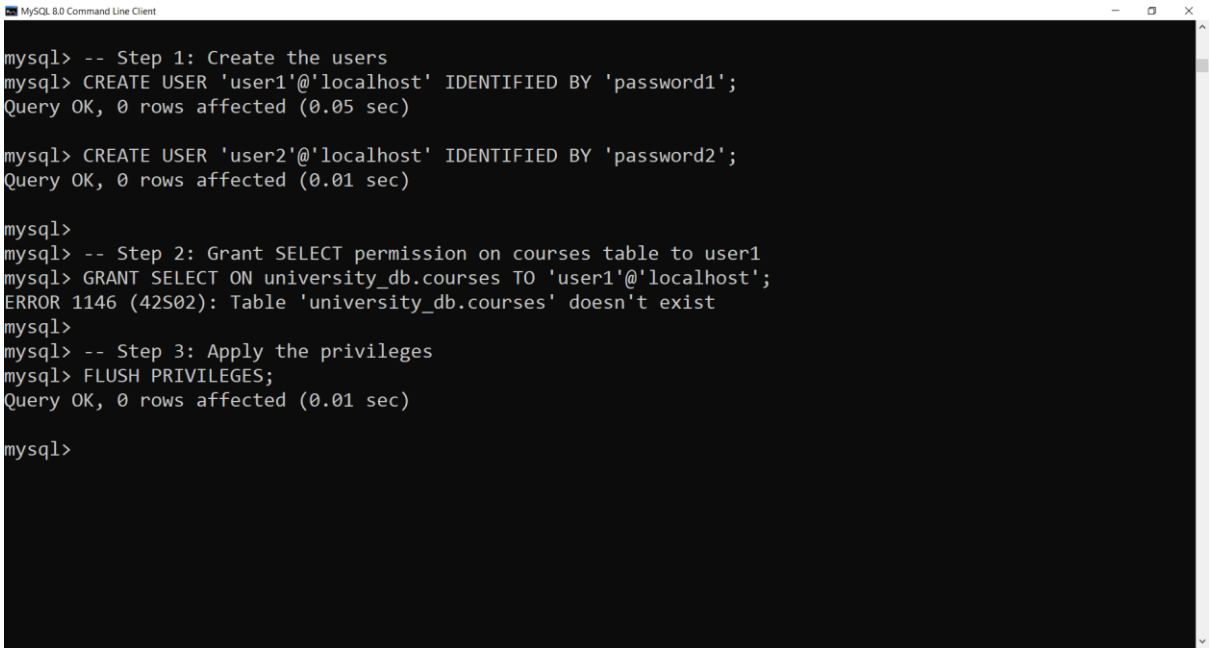
17. Sort the courses based on course_duration in descending order using ORDER BY.



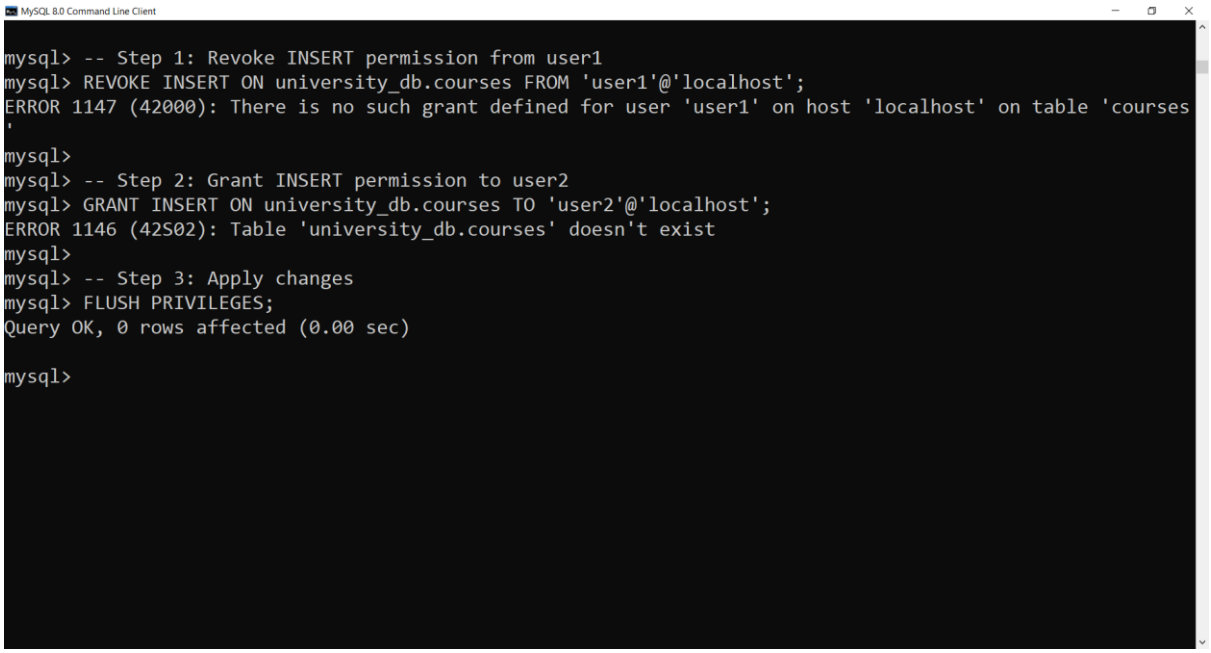
18. Limit the results of the SELECT query to show only the top two courses using LIMIT.



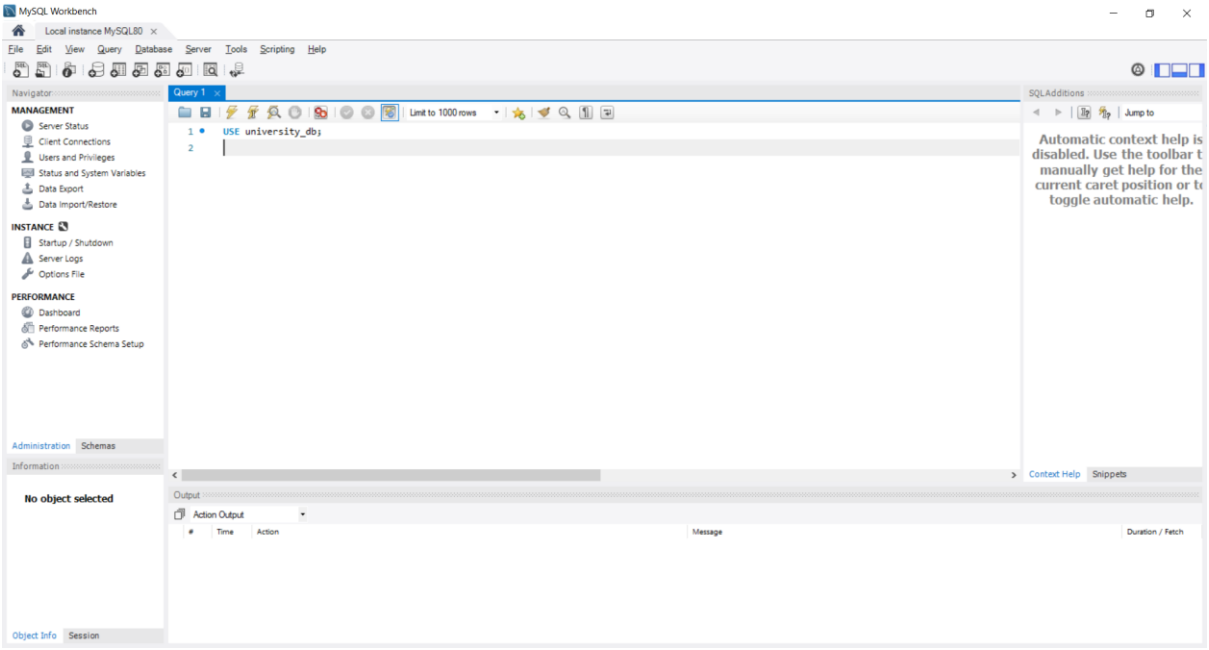
19. Create two new users user1 and user2 and grant user1 permission to SELECT from the courses table.

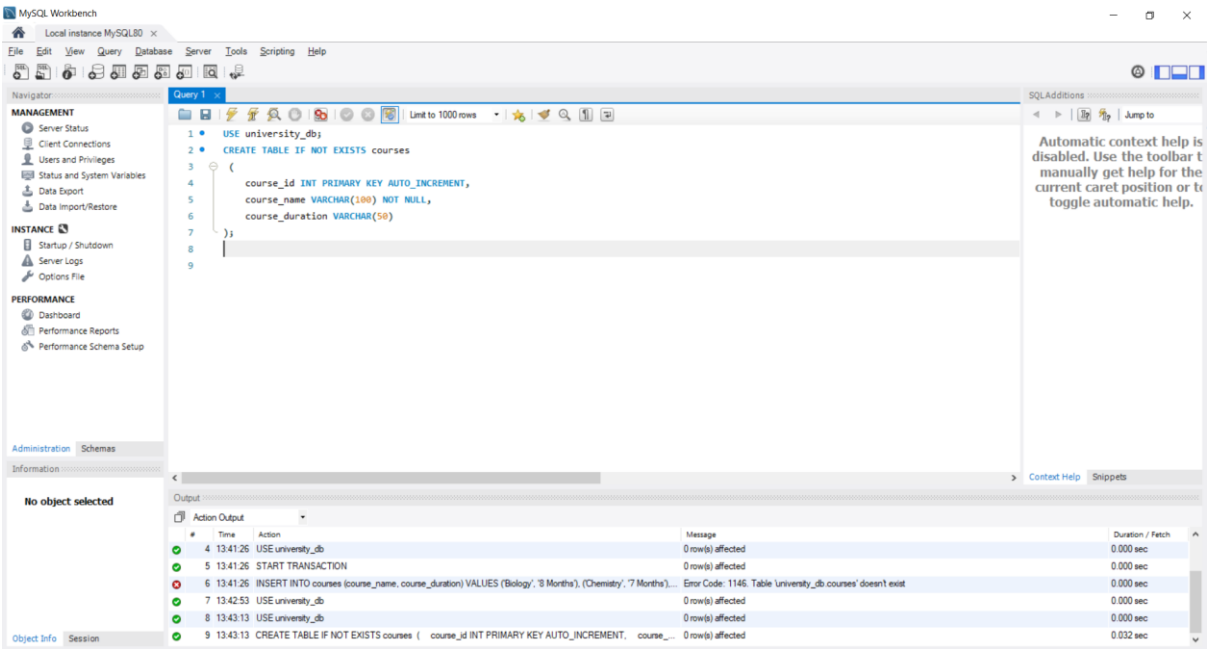


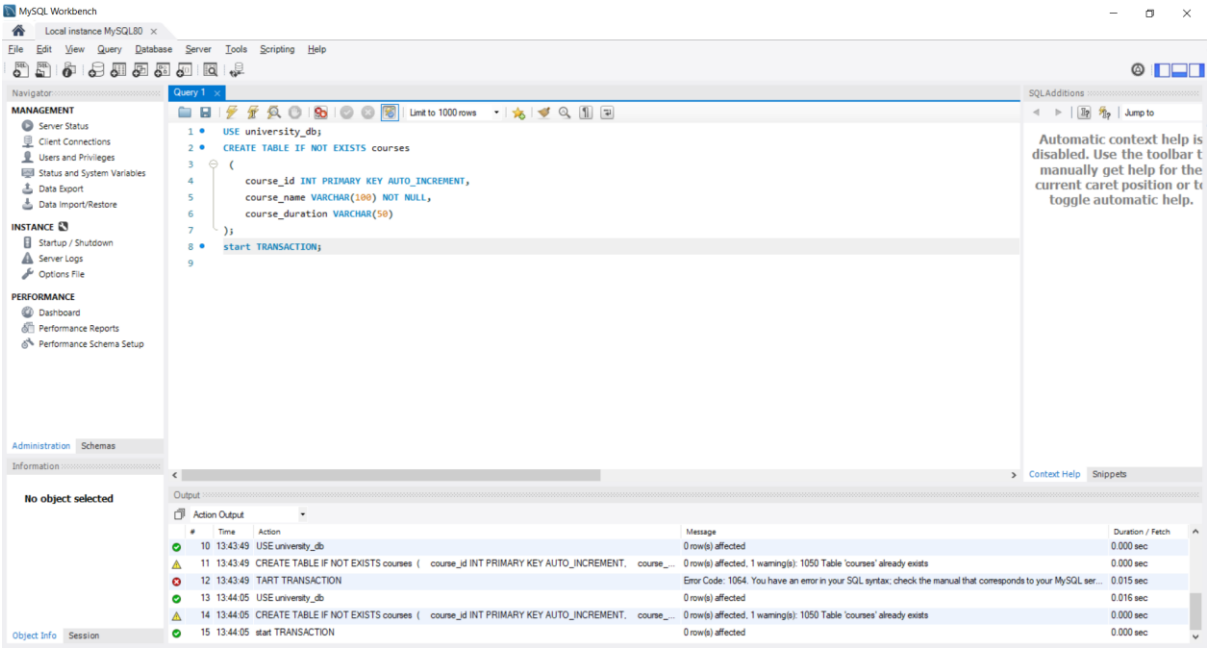
20. : Revoke the INSERT permission from user1 and give it to user2

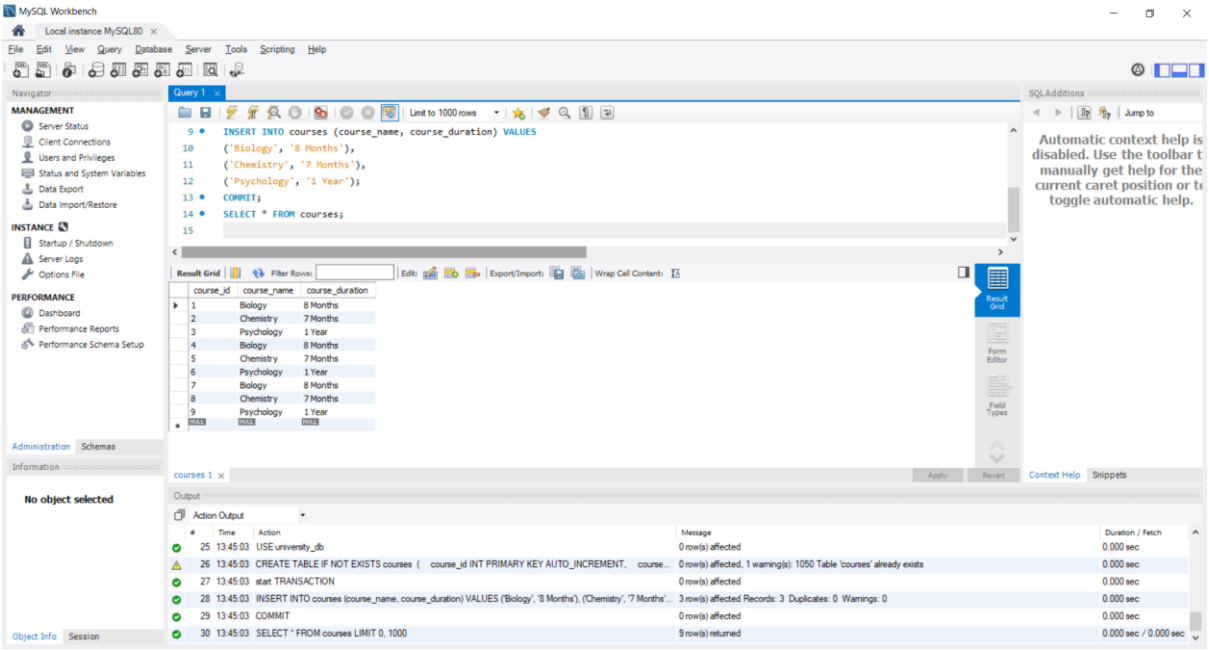


21. Insert a few rows into the courses table and use COMMIT to save the changes

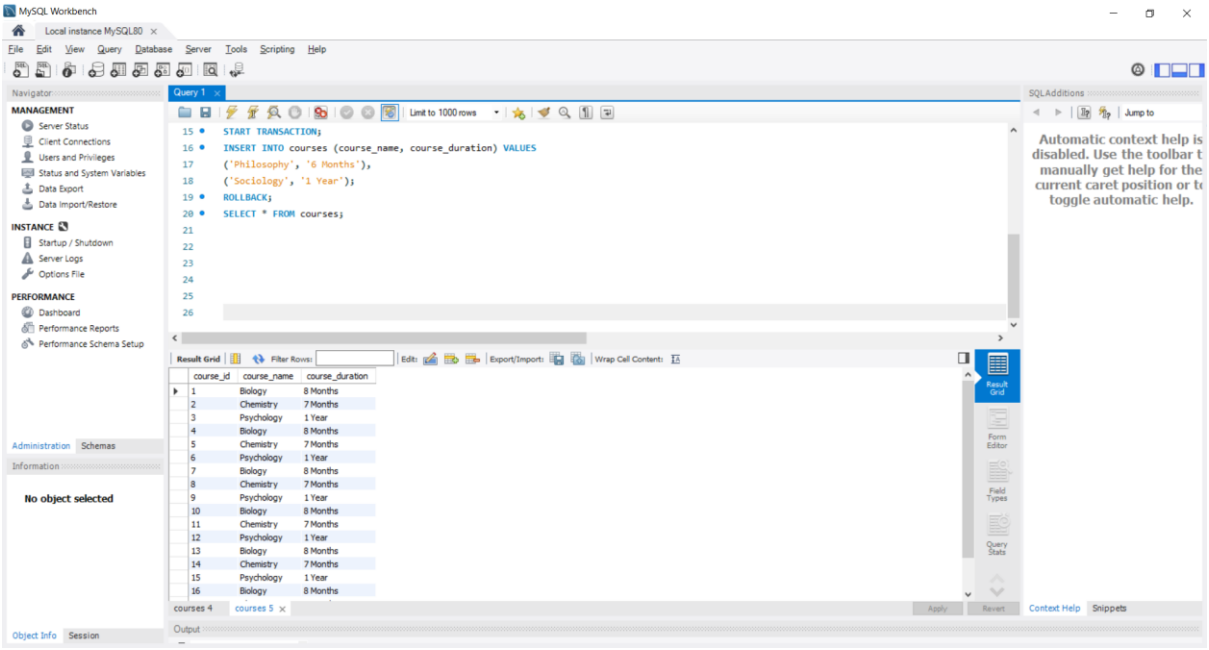




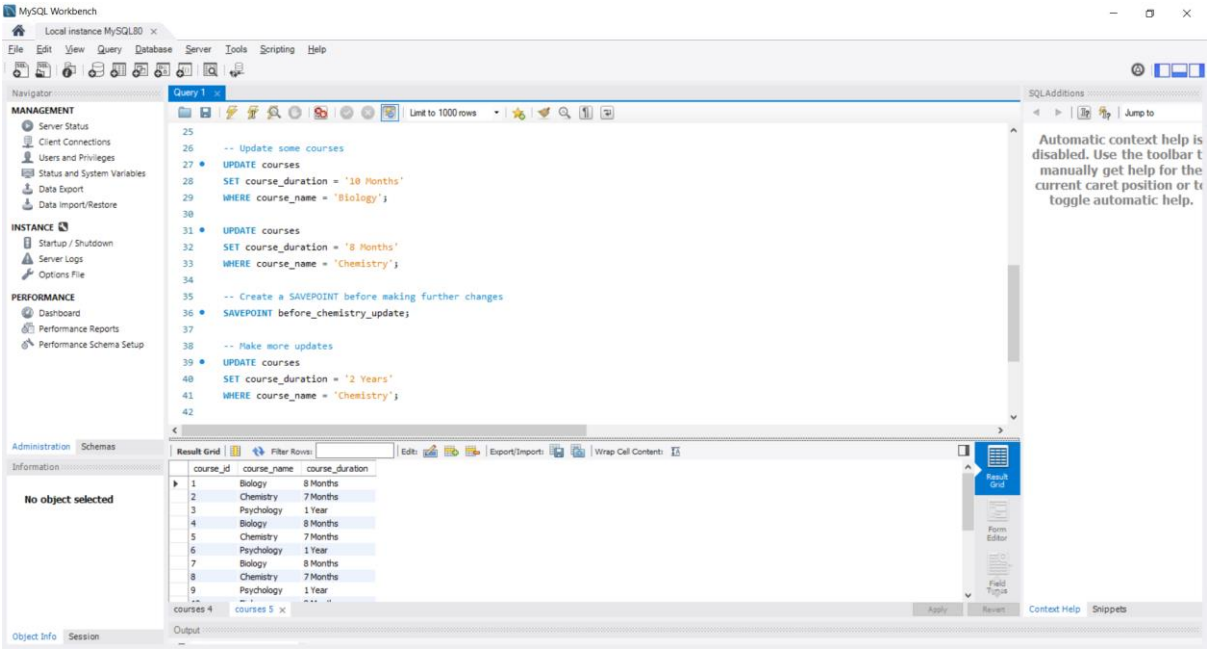


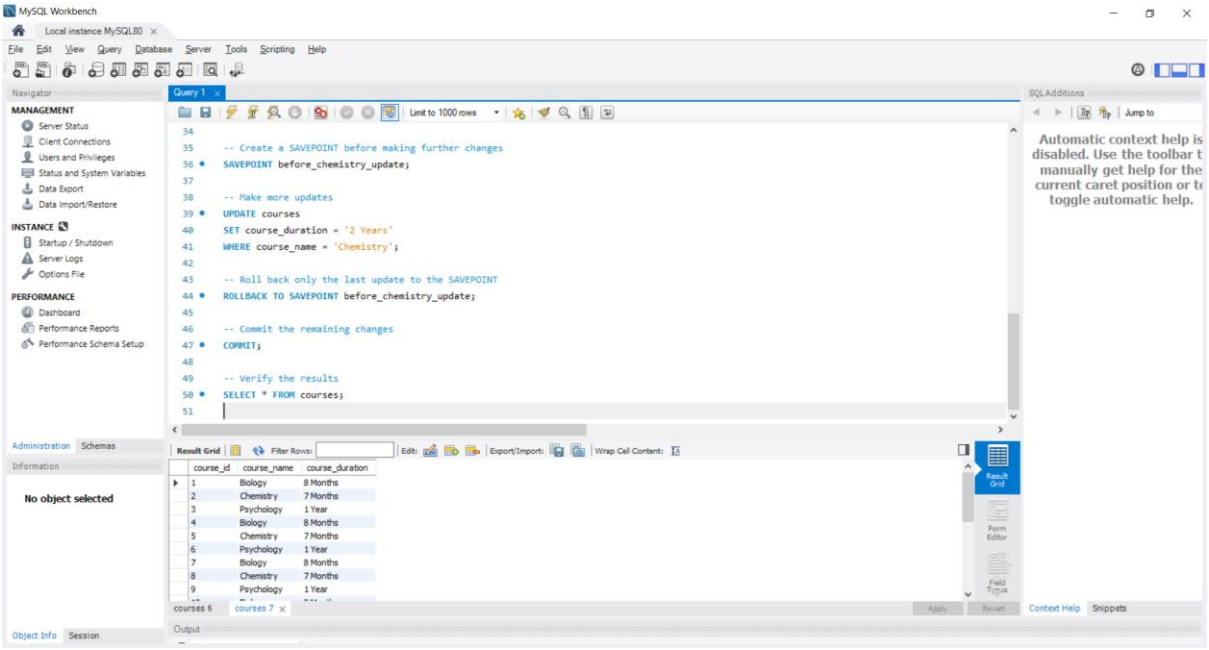


22. Insert additional rows, then use ROLLBACK to undo the last insert operation.

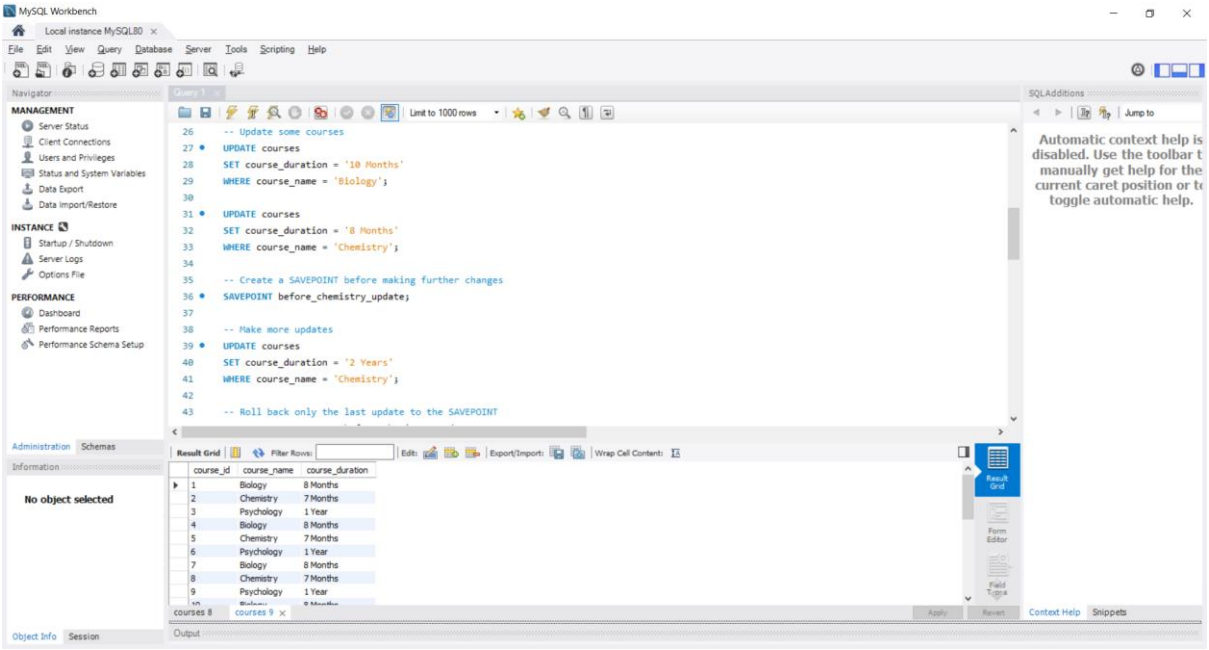


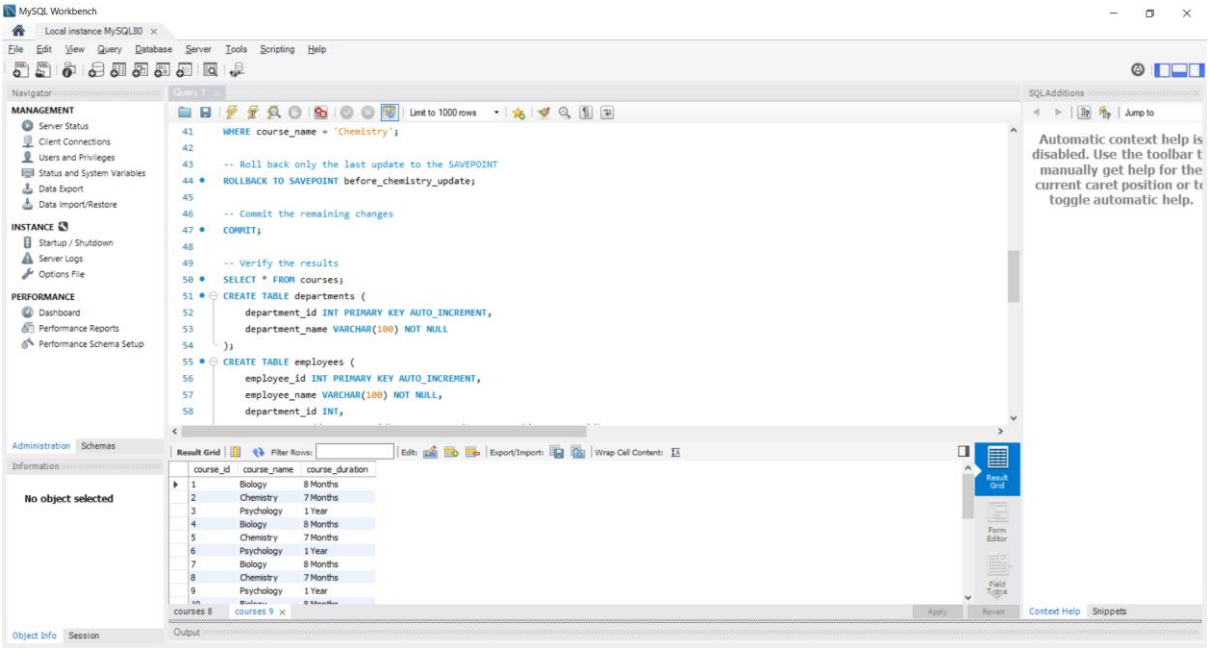
23. Create a SAVEPOINT before updating the courses table, and use it to roll back specific changes.

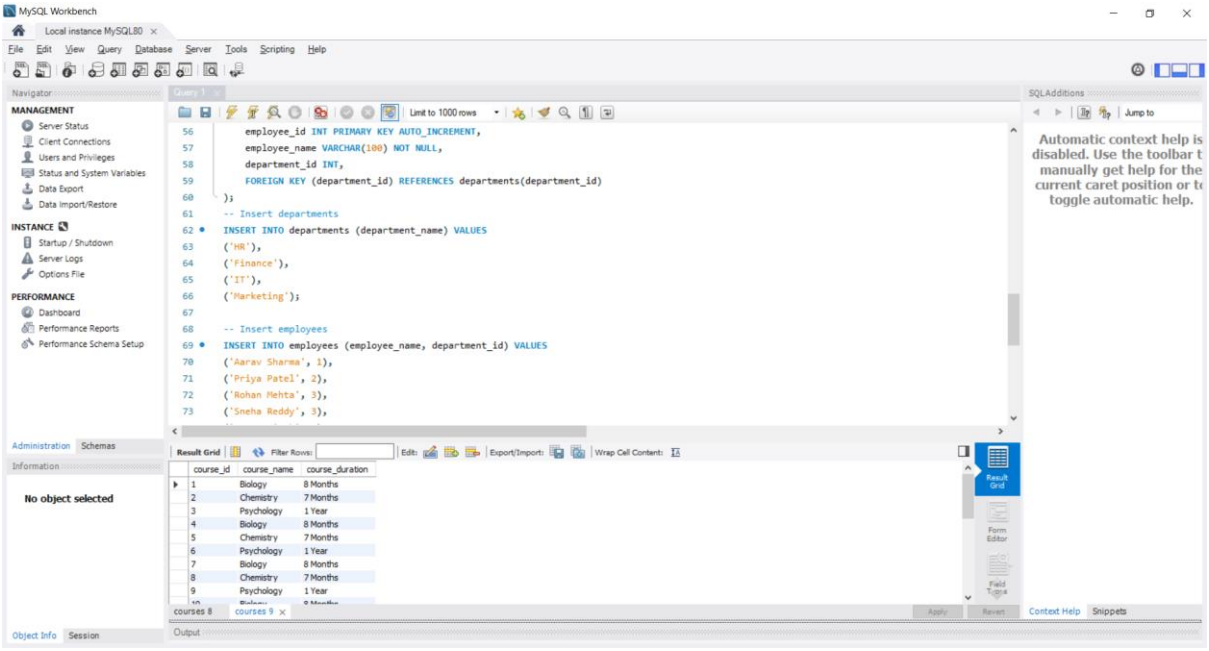


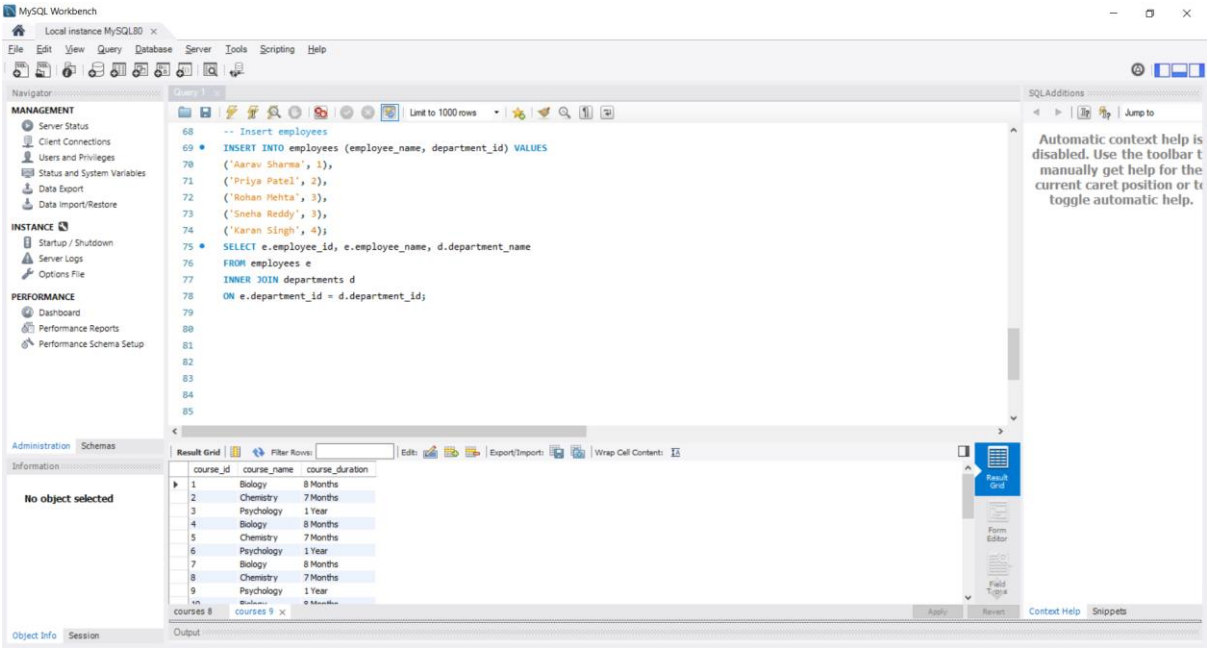


24. Create two tables: departments and employees. Perform an INNER JOIN to display employees along with their respective departments

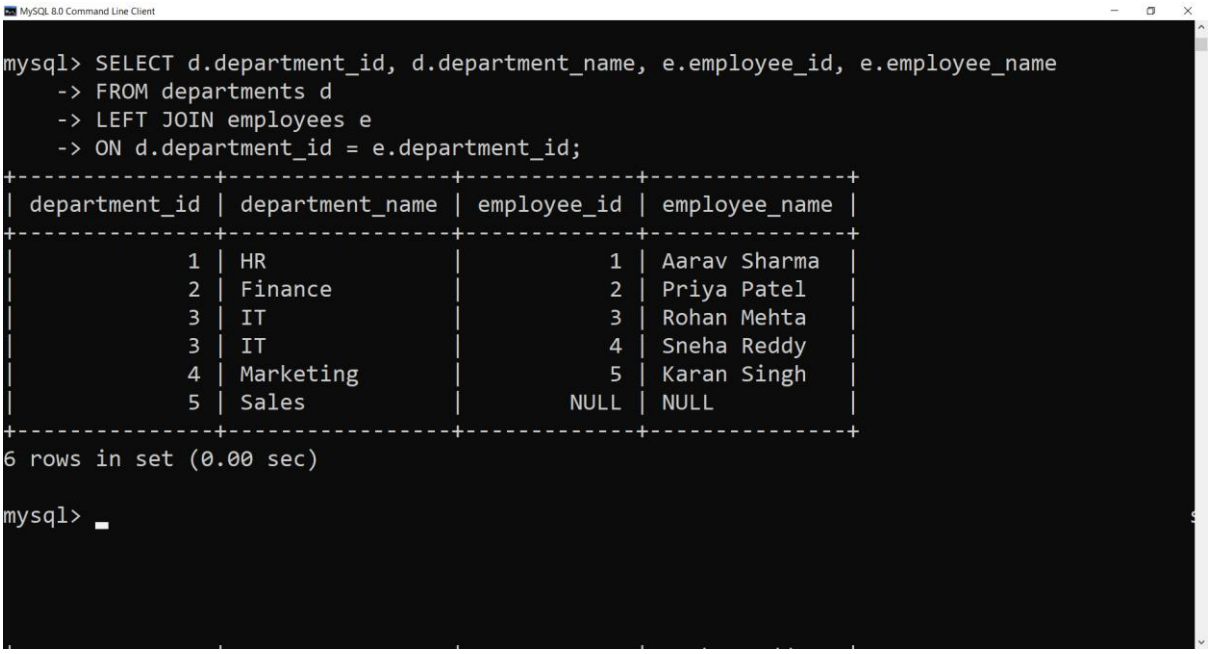



MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
MANAGEMENT
Server Status
Client Connections
Users and Privileges
Status and System Variables
Data Export
Data Import/Restore
INSTANCE
Startup / Shutdown
Server Logs
Options File
PERFORMANCE
Dashboard
Performance Reports
Performance Schema Setup
Administration Schemas
Information
No object selected
Object Info Session
Limit to 1000 rows
41 WHERE course_name = 'Chemistry';
42
43 -- Roll back only the last update to the SAVEPOINT
44 ROLLBACK TO SAVEPOINT before_chemistry_update;
45
46 -- Commit the remaining changes
47 COMMIT;
48
49 -- Verify the results
50 SELECT * FROM courses;
51 CREATE TABLE departments (
52 department_id INT PRIMARY KEY AUTO_INCREMENT,
53 department_name VARCHAR(100) NOT NULL
54);
55 CREATE TABLE employees (
56 employee_id INT PRIMARY KEY AUTO_INCREMENT,
57 employee_name VARCHAR(100) NOT NULL,
58 department_id INT,
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
course_id course_name course_duration
1 Biology 8 Months
2 Chemistry 7 Months
3 Psychology 1 Year
4 Biology 8 Months
5 Chemistry 7 Months
6 Psychology 1 Year
7 Biology 8 Months
8 Chemistry 7 Months
9 Psychology 1 Year
courses 8 courses 9
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Context Help Snippets
Output

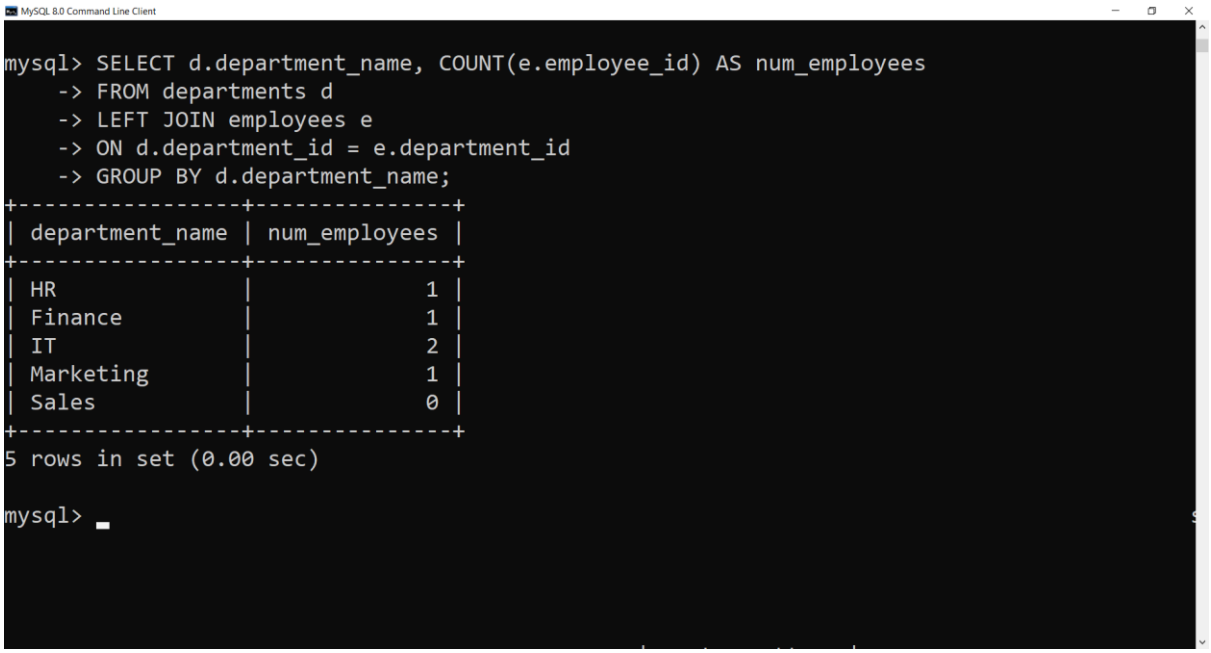

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
MANAGEMENT
Server Status
Client Connections
Users and Privileges
Status and System Variables
Data Export
Data Import/Restore
INSTANCE
Startup / Shutdown
Server Logs
Options File
PERFORMANCE
Dashboard
Performance Reports
Performance Schema Setup
Administration Schemas
Information
No object selected
Object Info Session
Limit to 1000 rows
56 employee_id INT PRIMARY KEY AUTO_INCREMENT,
57 employee_name VARCHAR(100) NOT NULL,
58 department_id INT,
59 FOREIGN KEY (department_id) REFERENCES departments(department_id)
60);
61 -- Insert departments
62 INSERT INTO departments (department_name) VALUES
63 ('HR'),
64 ('Finance'),
65 ('IT'),
66 ('Marketing');
67
68 -- Insert employees
69 INSERT INTO employees (employee_name, department_id) VALUES
70 ('Aarav Sharma', 1),
71 ('Priya Patel', 2),
72 ('Rohan Mehta', 3),
73 ('Sneha Reddy', 3),
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
course_id course_name course_duration
1 Biology 8 Months
2 Chemistry 7 Months
3 Psychology 1 Year
4 Biology 8 Months
5 Chemistry 7 Months
6 Psychology 1 Year
7 Biology 8 Months
8 Chemistry 7 Months
9 Psychology 1 Year
courses 8 courses 9
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Context Help Snippets
Output



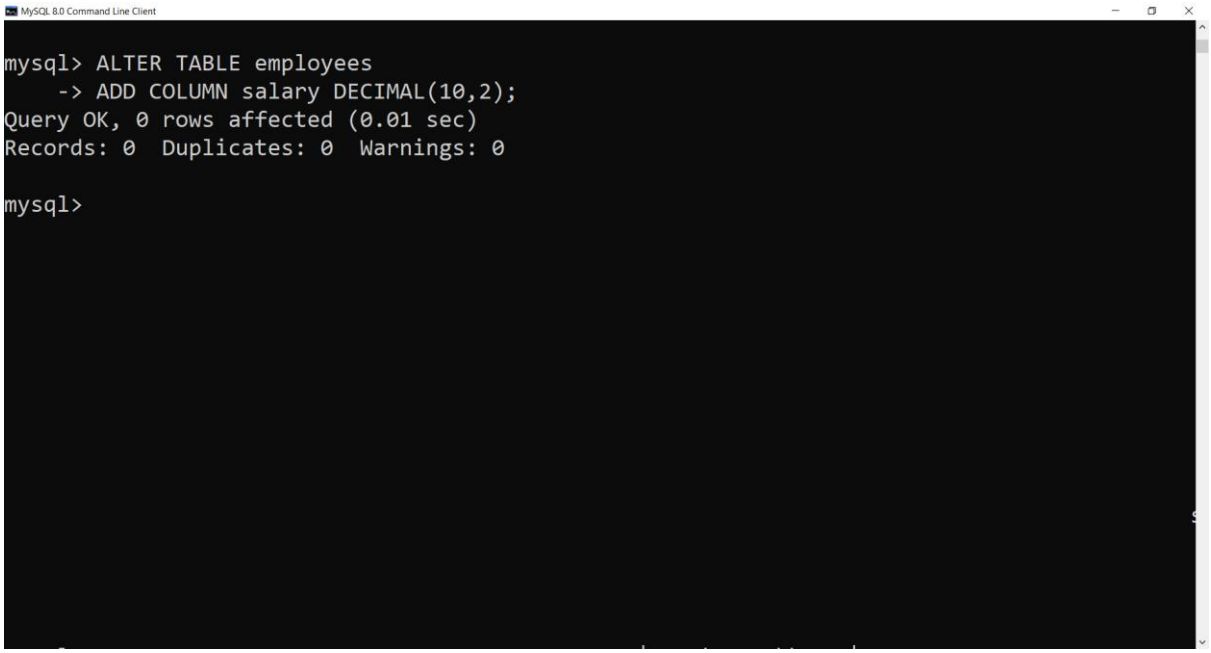
25. Use a LEFT JOIN to show all departments, even those without employees.

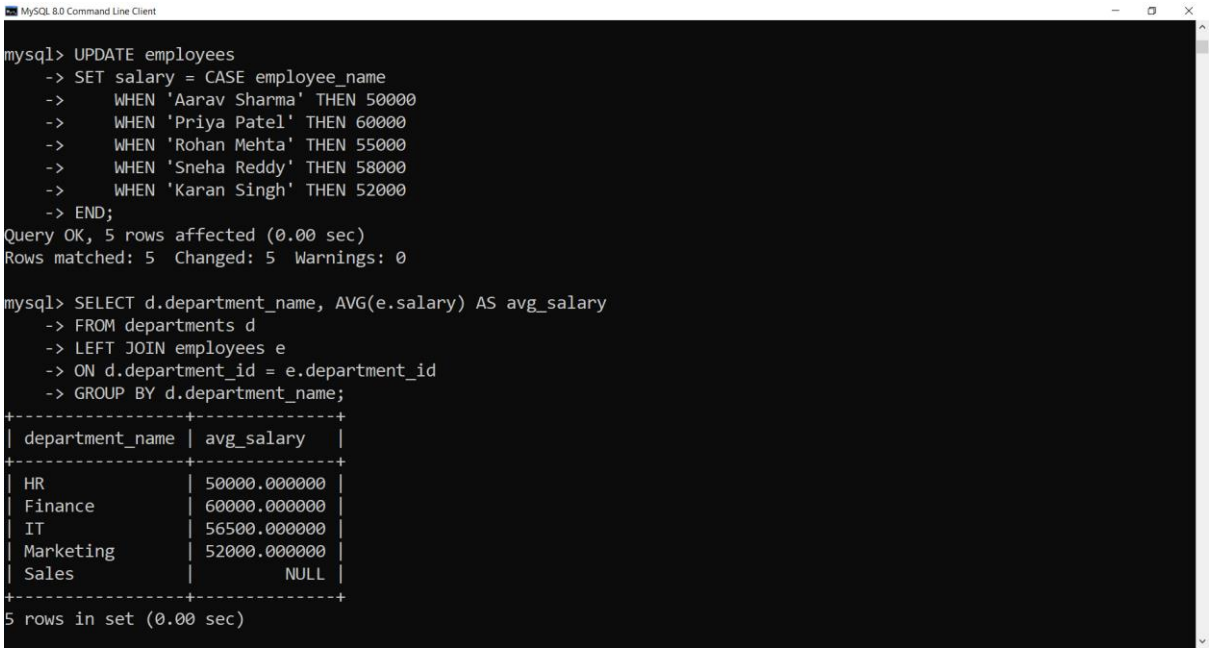


26. : Group employees by department and count the number of employees in each department using GROUP BY.

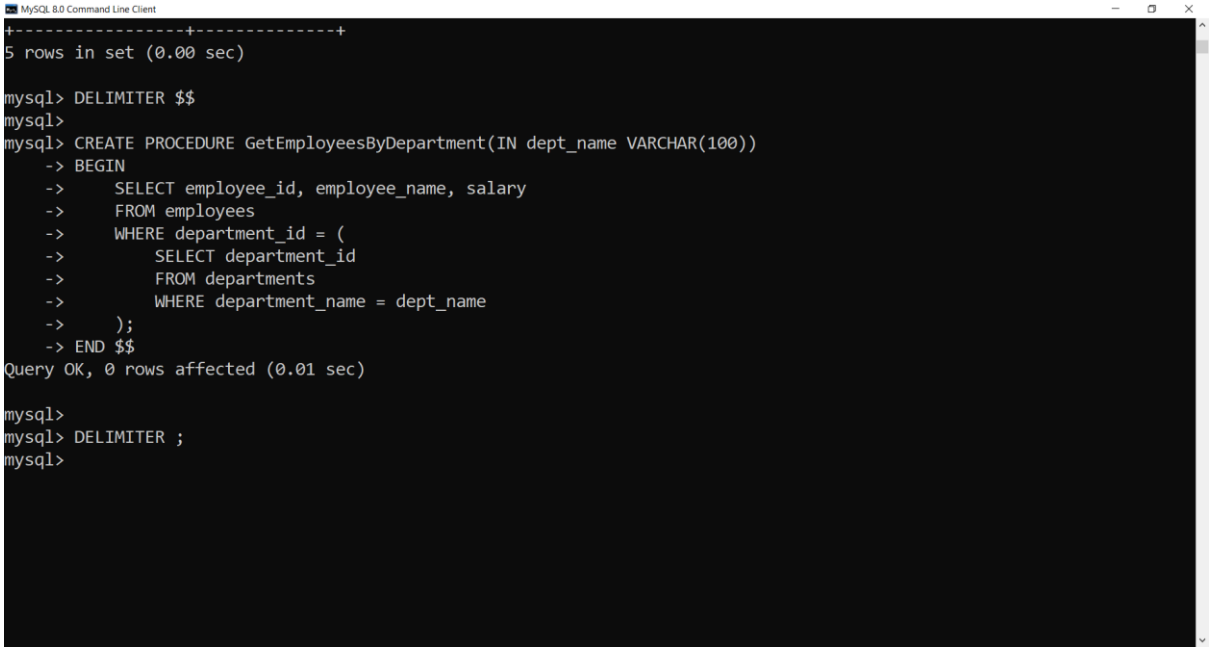


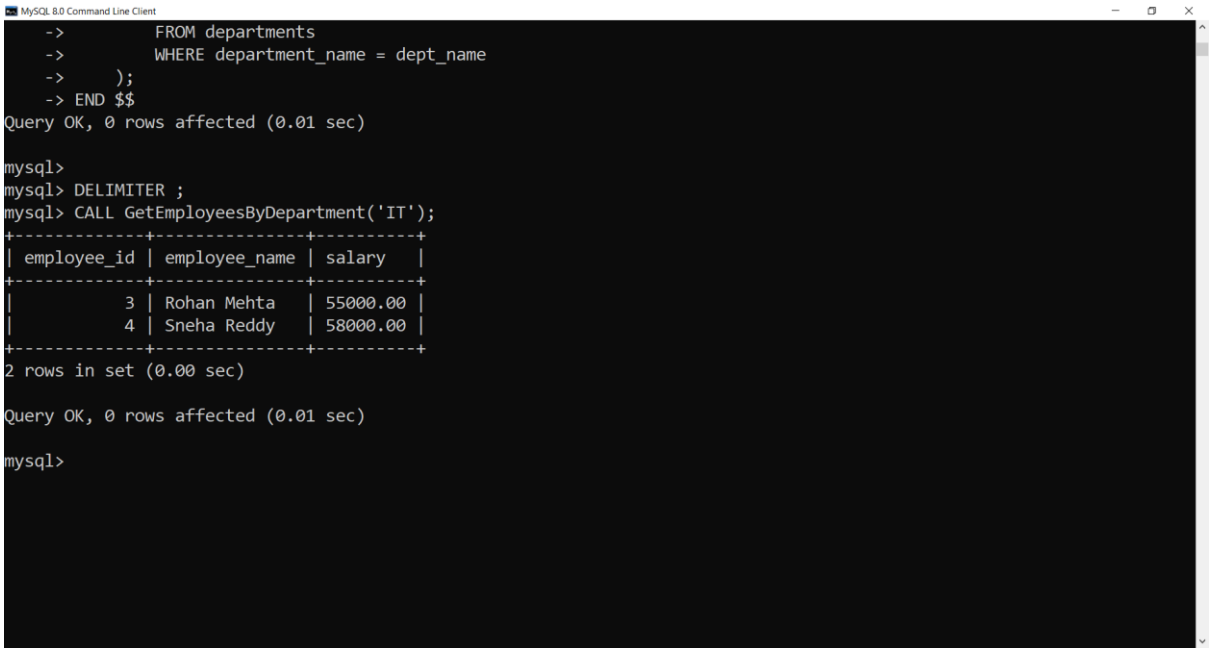
27. : Use the AVG aggregate function to find the average salary of employees in each department



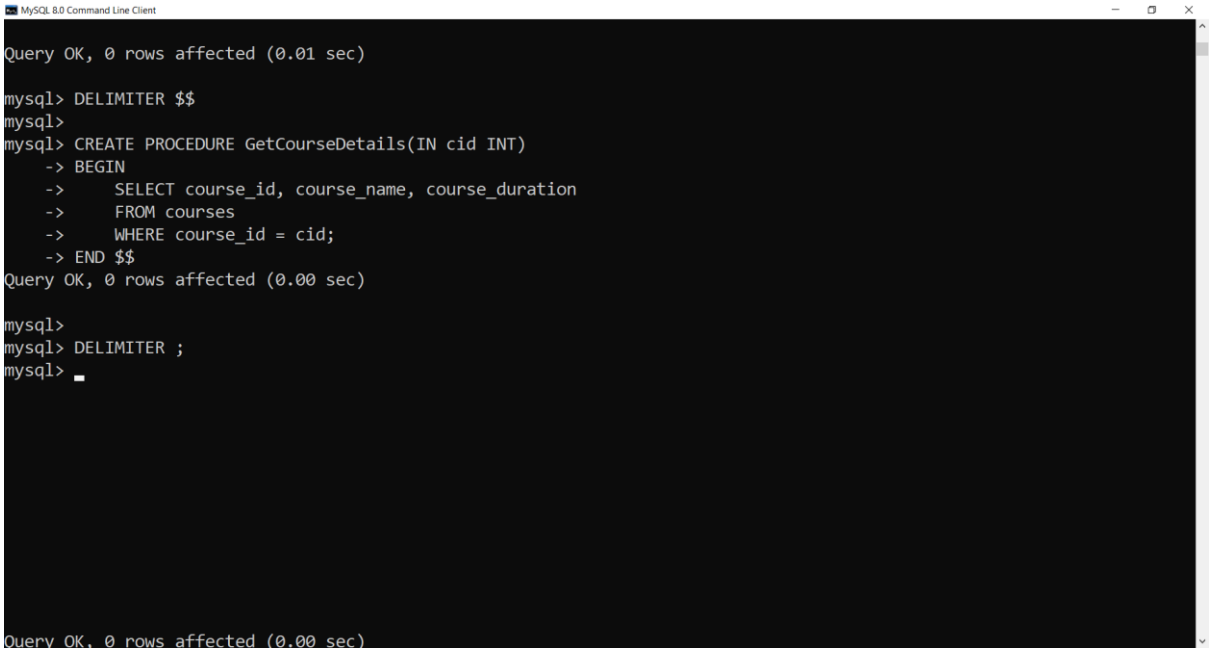


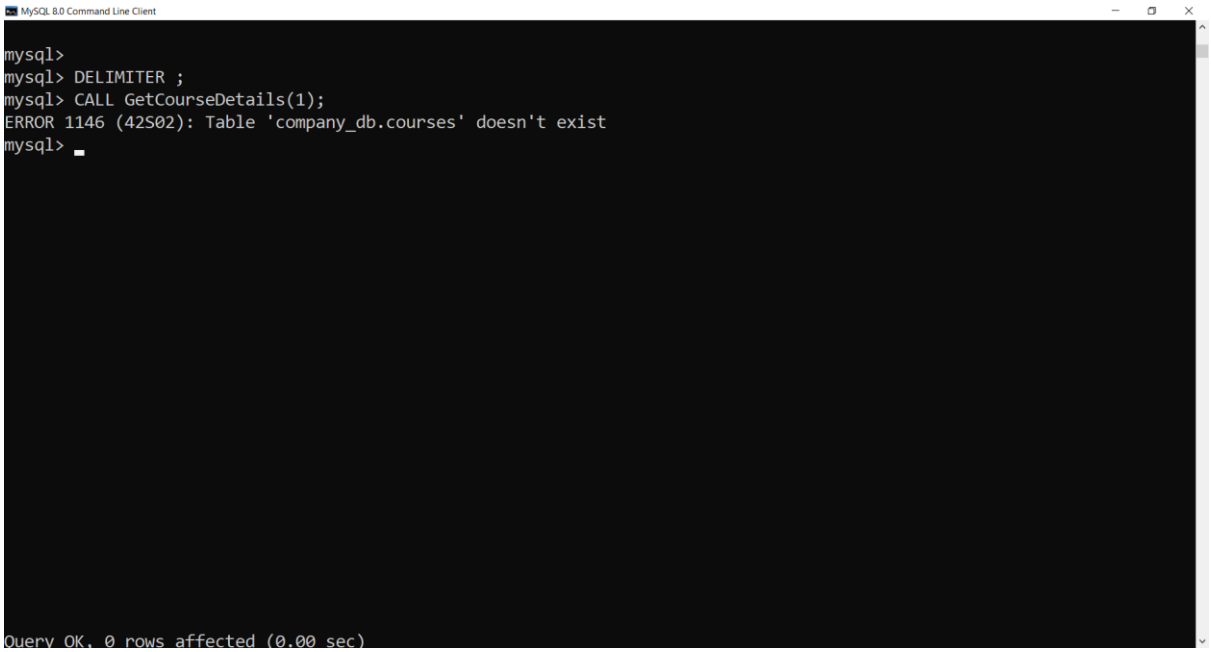
28. Write a stored procedure to retrieve all employees from the employees table based on department.



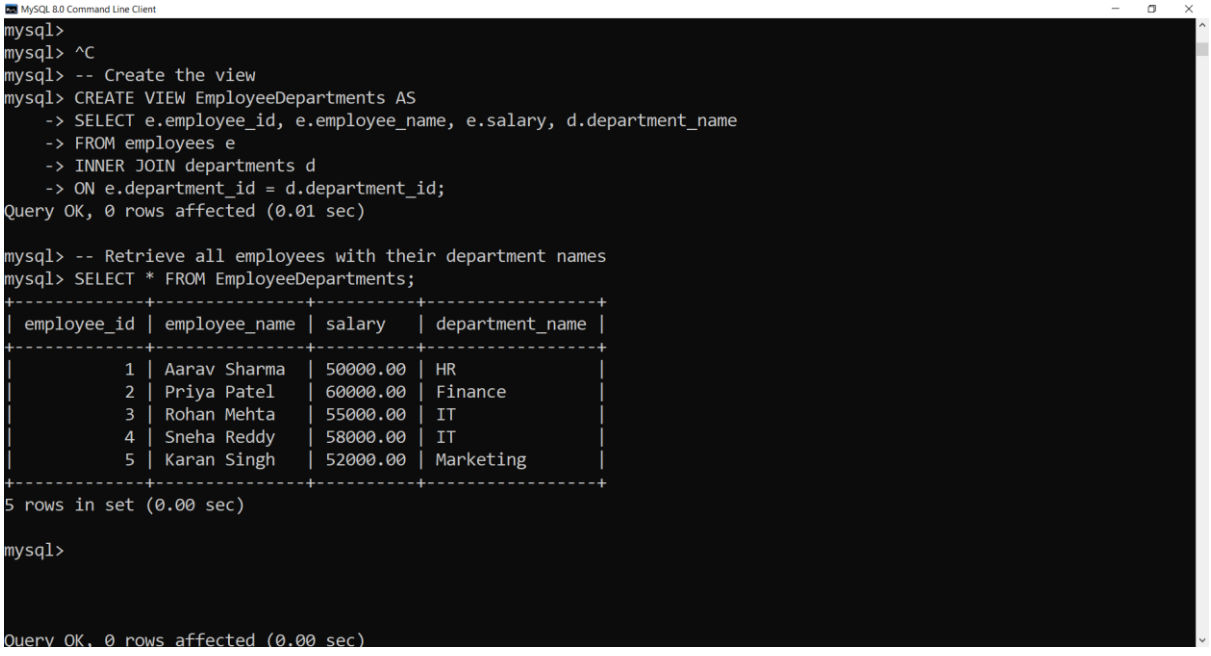


29. Write a stored procedure that accepts course_id as input and returns the course details.

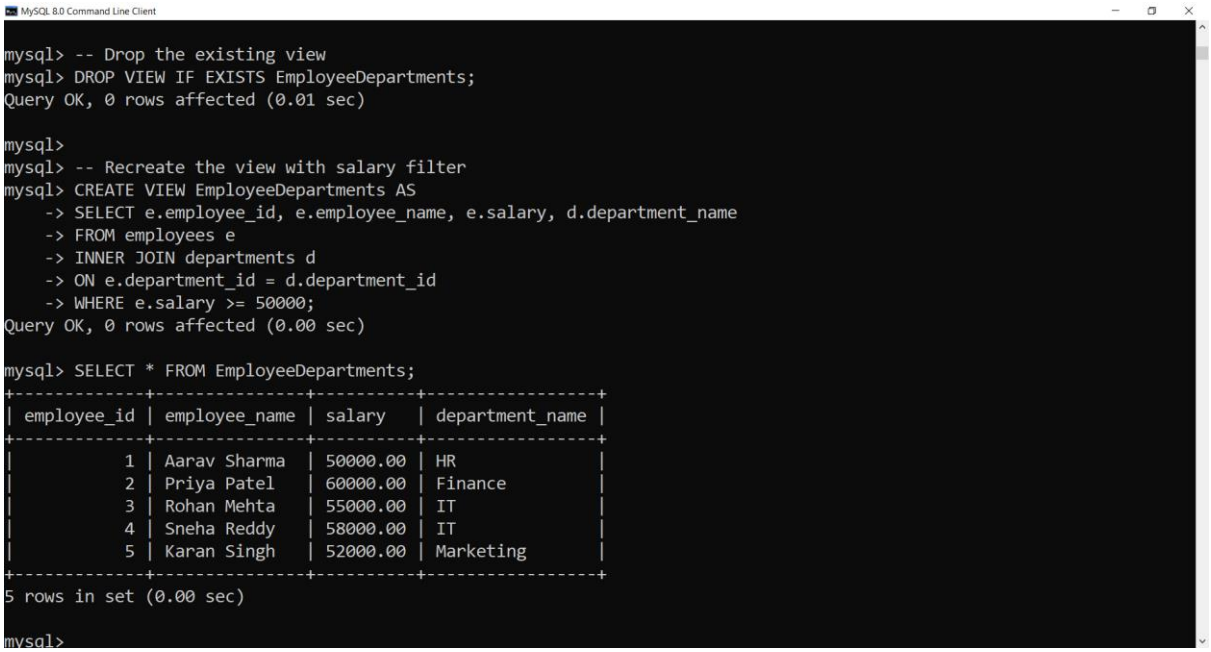




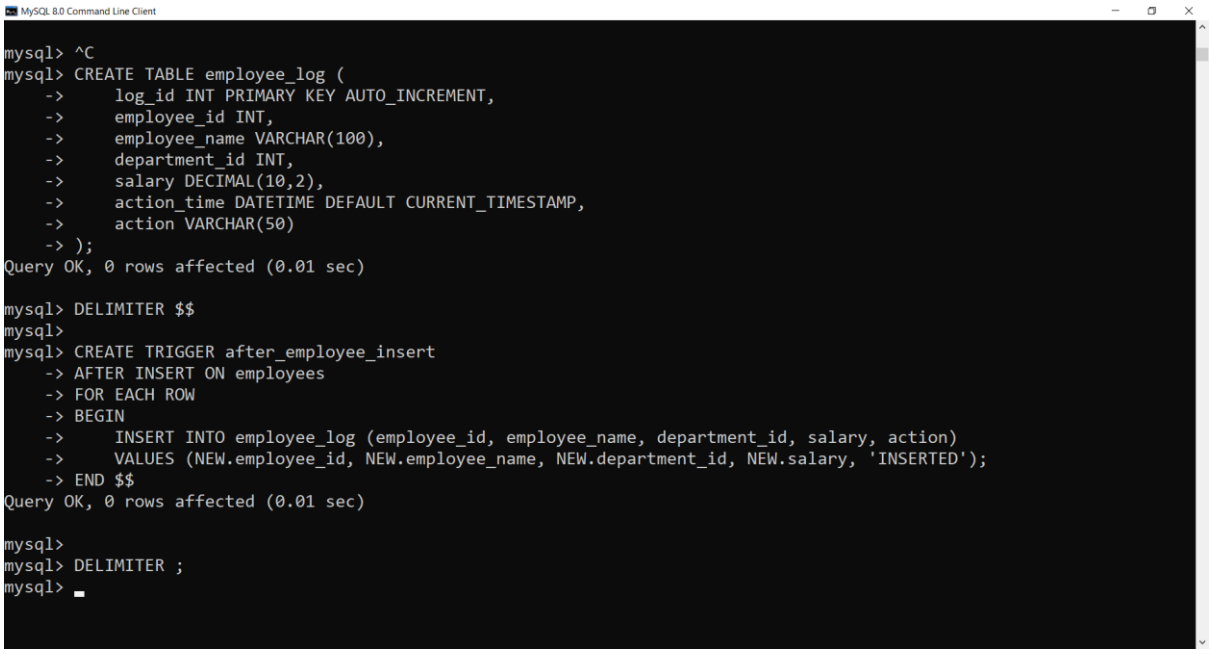
30. : Create a view to show all employees along with their department names



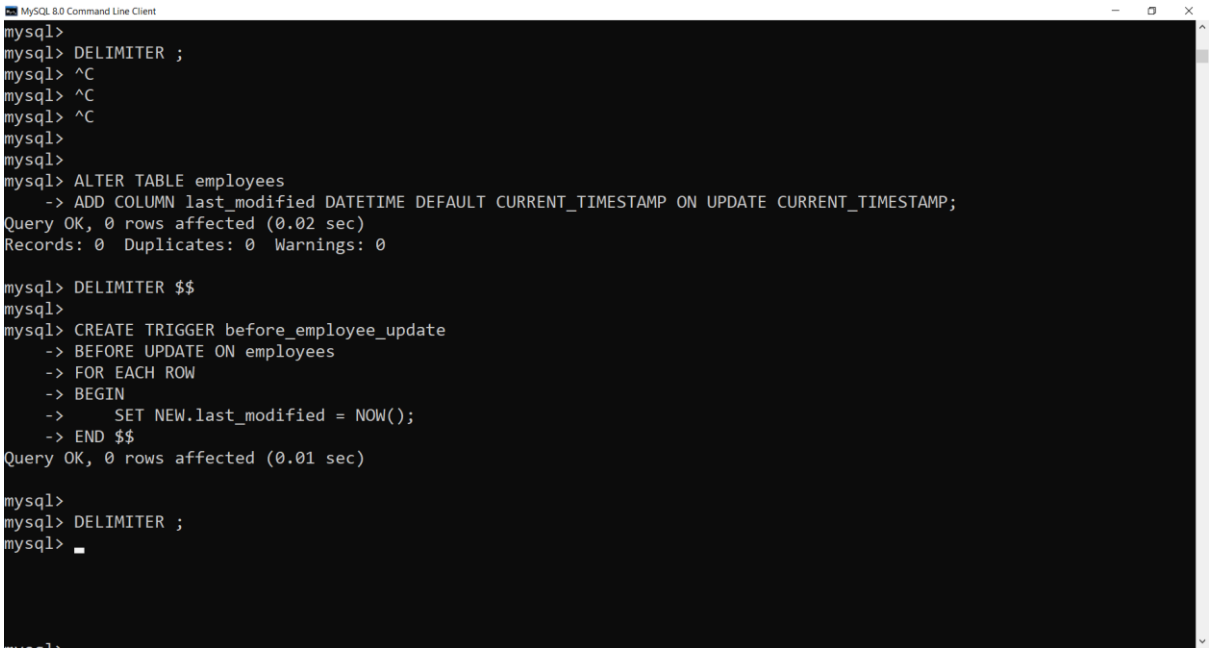
31. Modify the view to exclude employees whose salaries are below $50,000.



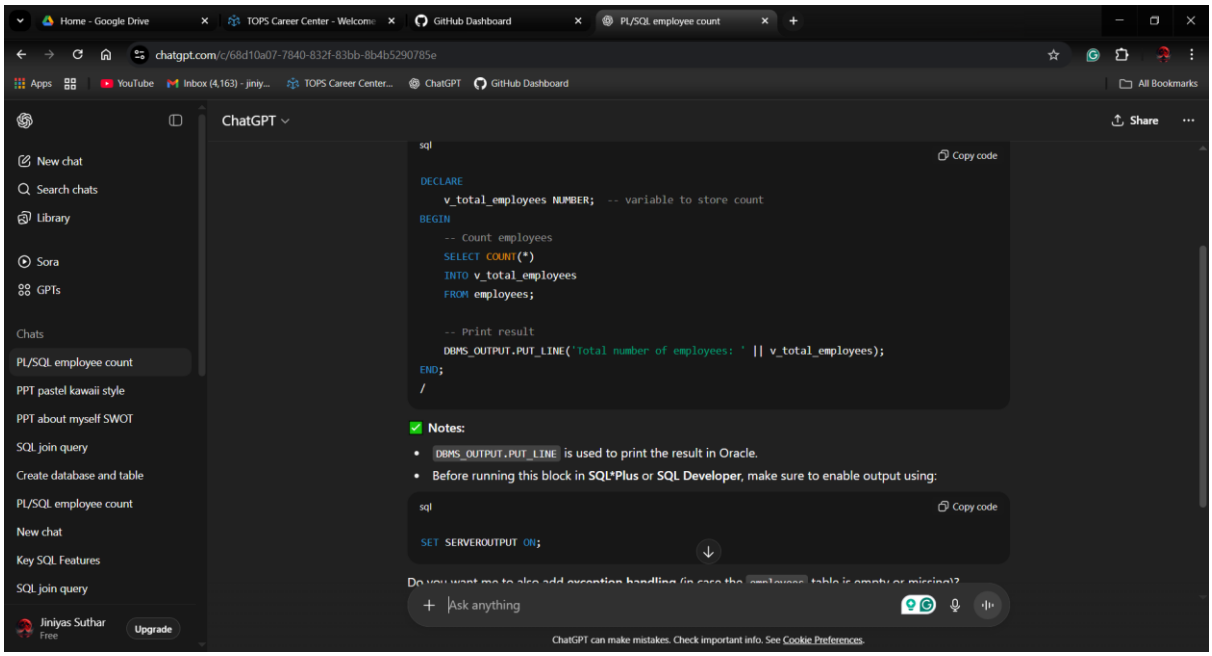
32. Create a trigger to automatically log changes to the employees table when a new employee is added.



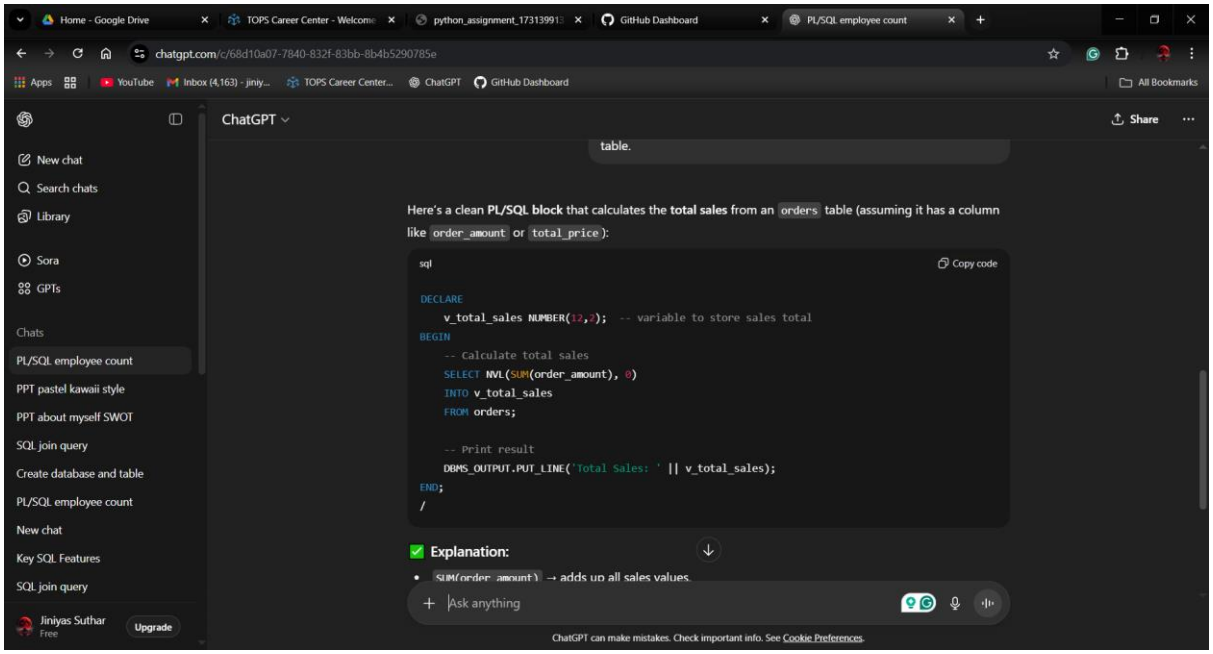
33. : Create a trigger to update the last_modified timestamp whenever an employee record is updated.



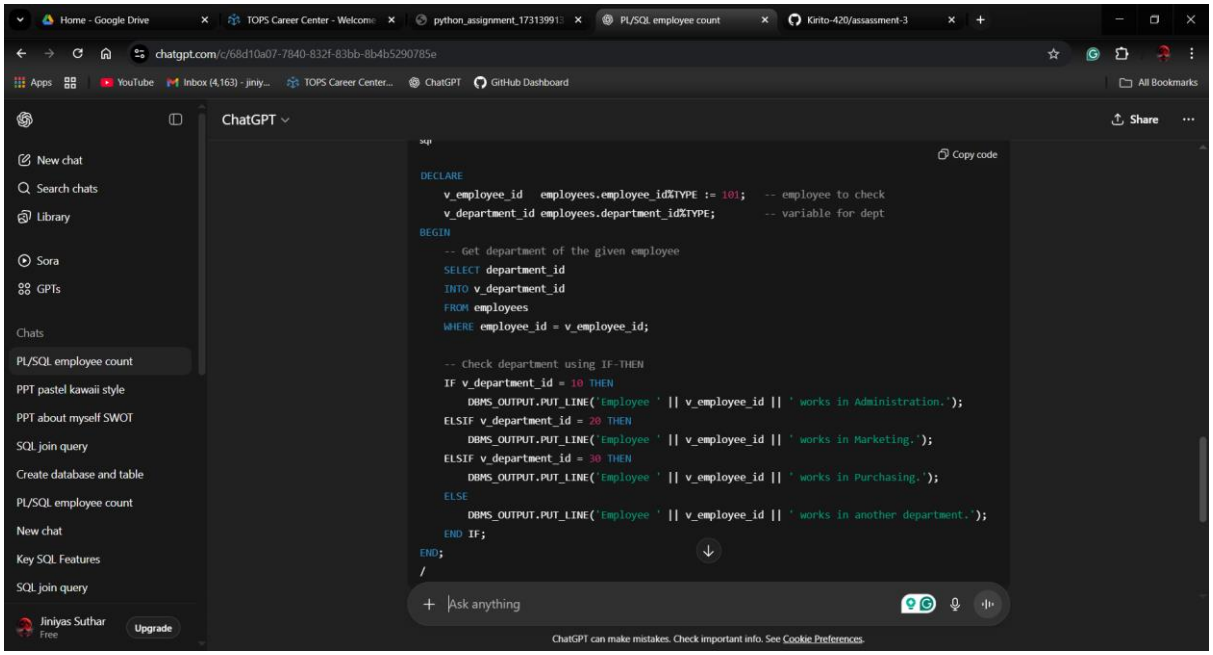
34. Write a PL/SQL block to print the total number of employees from the employees table.



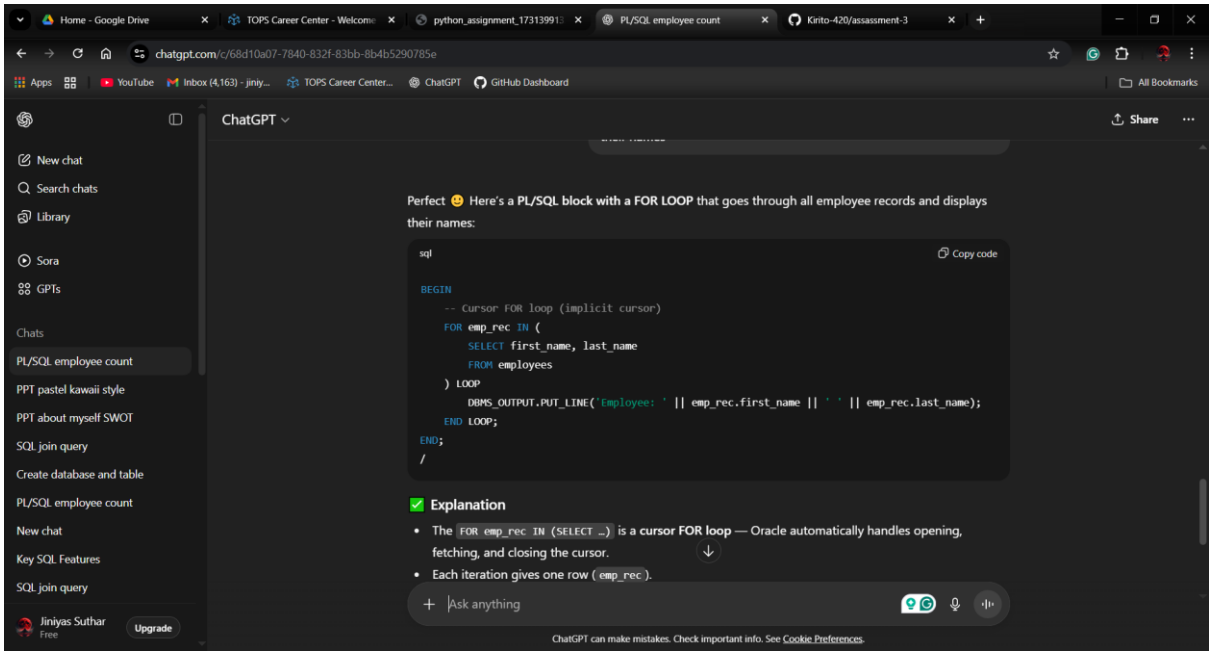
35. Create a PL/SQL block that calculates the total sales from an orders table.



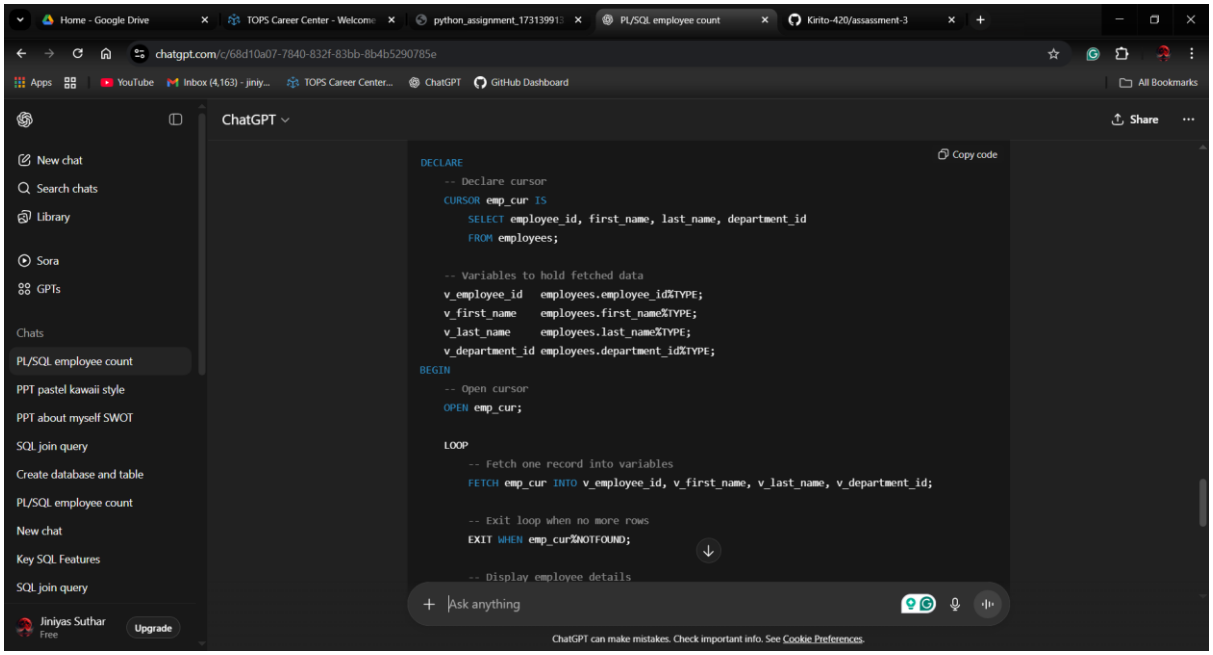
36. : Write a PL/SQL block using an IF-THEN condition to check the department of an employee.

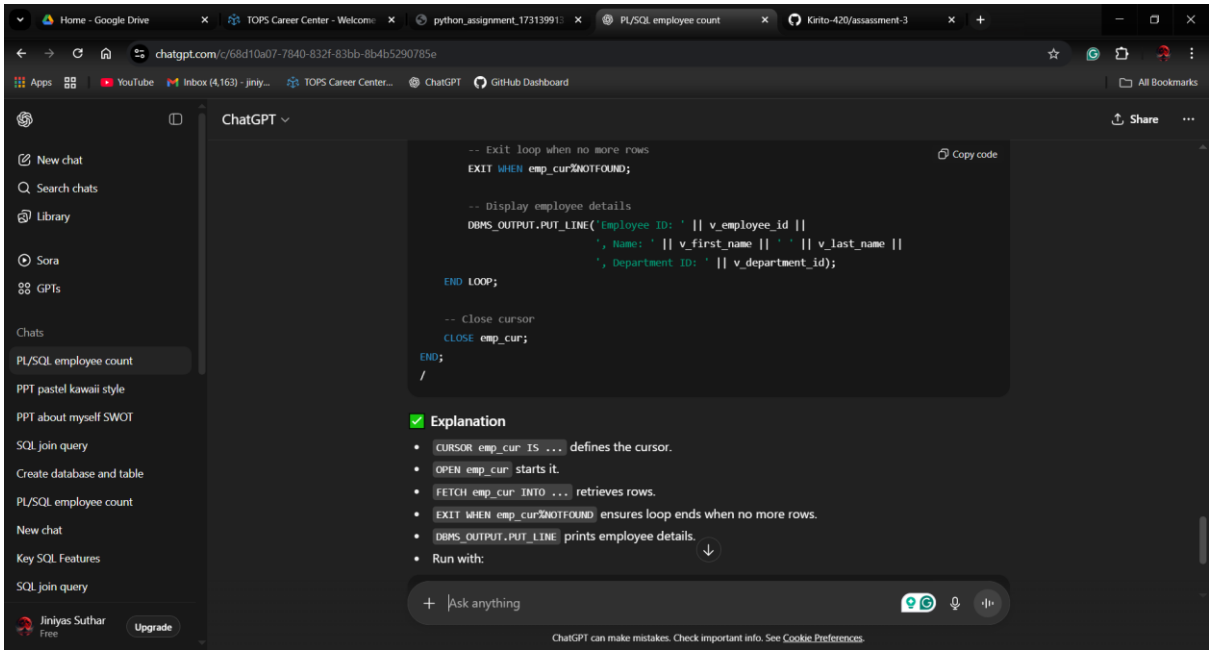


37. : Use a FOR LOOP to iterate through employee records and display their names

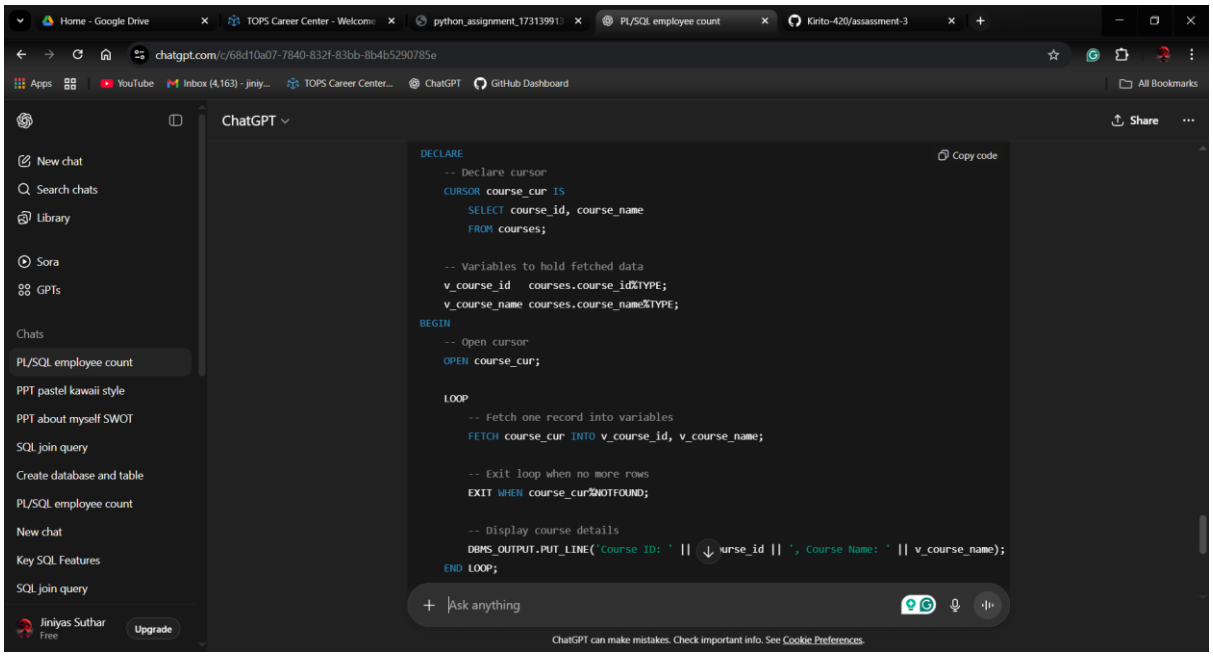


38. Write a PL/SQL block using an explicit cursor to retrieve and display employee details.

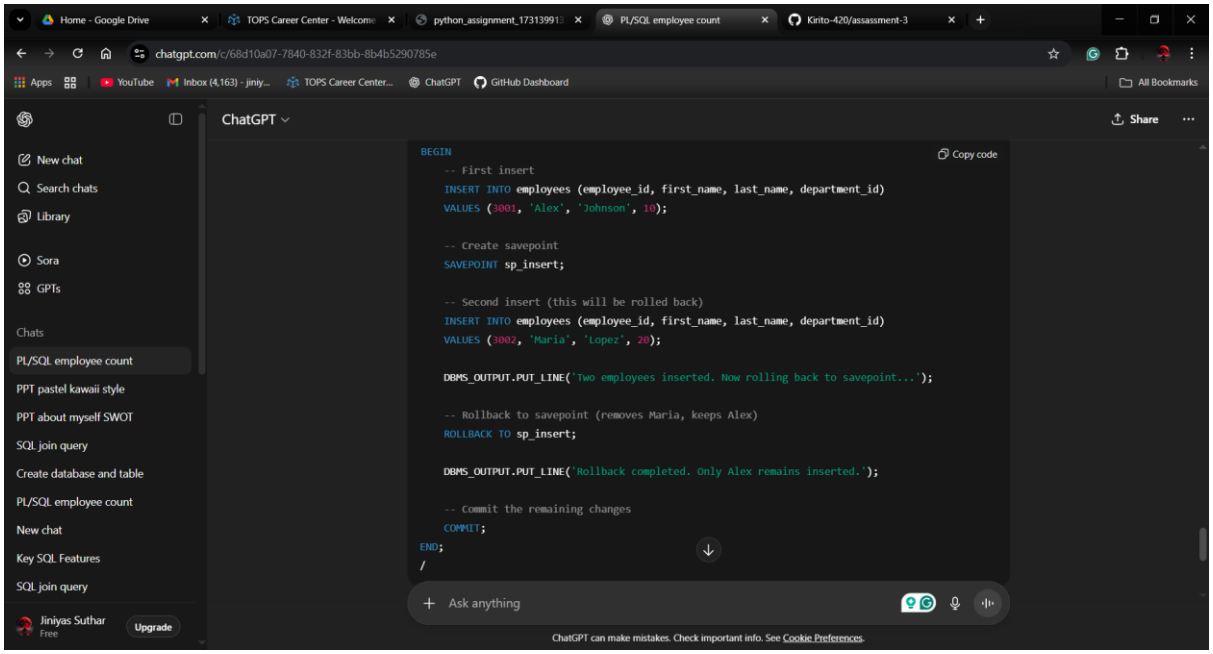




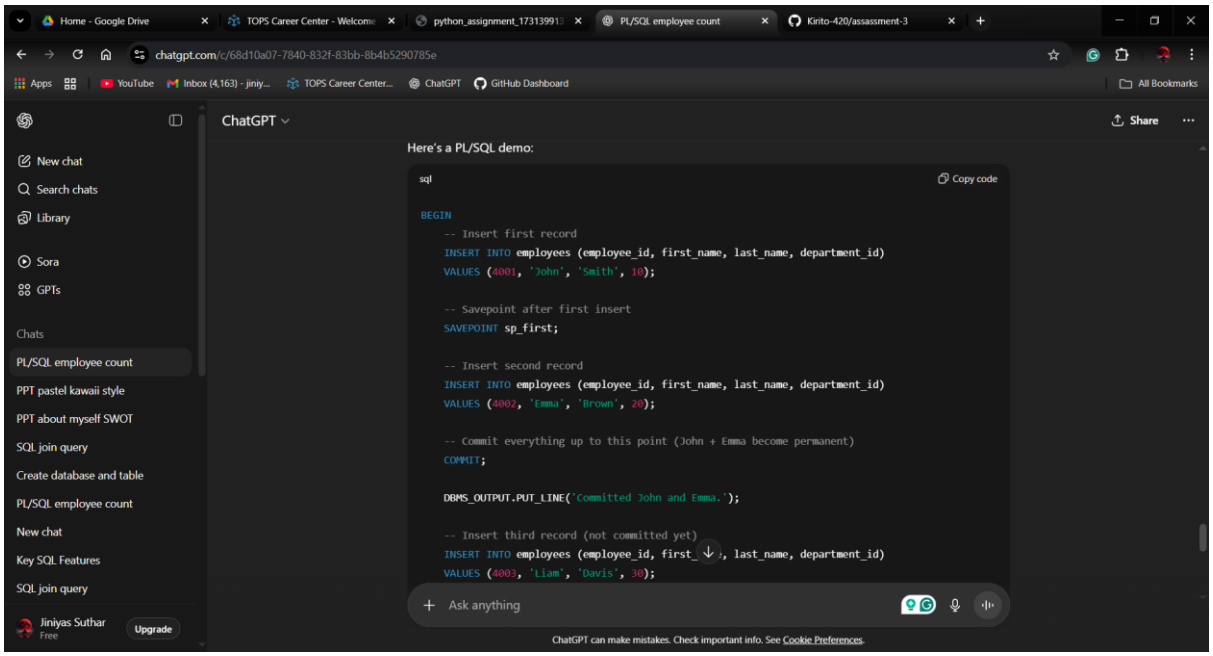
39. Create a cursor to retrieve all courses and display them one by one.

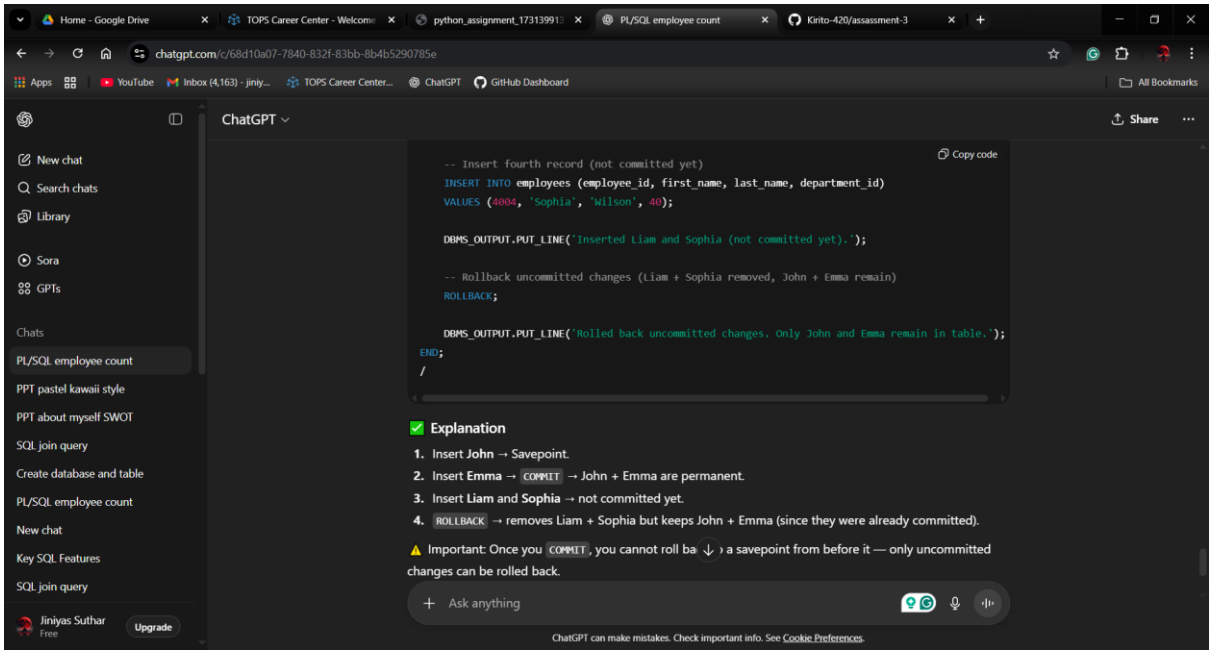


40. Perform a transaction where you create a savepoint, insert records, then rollback to the savepoint.



41. Commit part of a transaction after using a savepoint and then rollback the remaining changes.




Home - Google Drive
TOPS Career Center - Welcome
python_assignment_17313991
PL/SQL employee count
Kirito-420/assasment-3
chatgpt.com/c/68d10a07-7840-832f-83bb-8b4b5290785e
Apps
YouTube
Inbox (4,163) - jiniy...
TOPS Career Center...
ChatGPT
GitHub Dashboard
All Bookmarks
ChatGPT
Share
New chat
Search chats
Library
Sora
GPTs
Chats
PL/SQL employee count
PPT pastel kawaii style
PPT about myself SWOT
SQL join query
Create database and table
PL/SQL employee count
New chat
Key SQL Features
SQL join query
Jiniyas Suthar
Free
Upgrade
Copy code
-- Insert fourth record (not committed yet)
INSERT INTO employees (employee_id, first_name, last_name, department_id)
VALUES (4004, 'Sophia', 'Wilson', 40);
DBMS_OUTPUT.PUT_LINE('Inserted Liam and Sophia (not committed yet).');
-- Rollback uncommitted changes (Liam + Sophia removed, John + Emma remain)
ROLLBACK;
DBMS_OUTPUT.PUT_LINE('Rolled back uncommitted changes. Only John and Emma remain in table.');
END;
/
Explanation
1. Insert John → Savepoint.
2. Insert Emma → COMMIT → John + Emma are permanent.
3. Insert Liam and Sophia → not committed yet.
4. ROLLBACK → removes Liam + Sophia but keeps John + Emma (since they were already committed).
Important: Once you COMMIT, you cannot roll ba) a savepoint from before it — only uncommitted changes can be rolled back.
Ask anything
ChatGPT can make mistakes. Check important info. See Cookie Preferences.