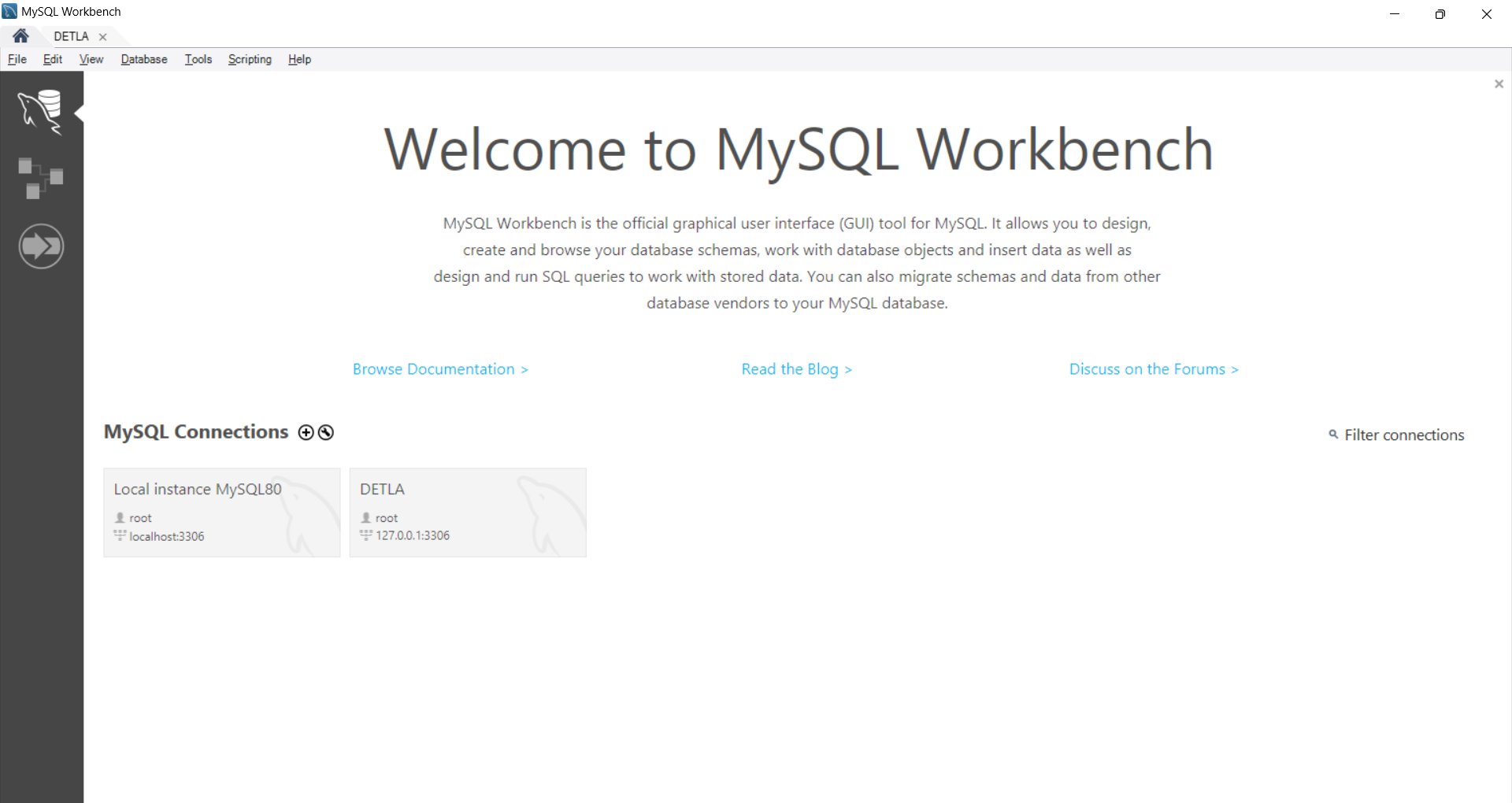
MYSQL Database Assignments

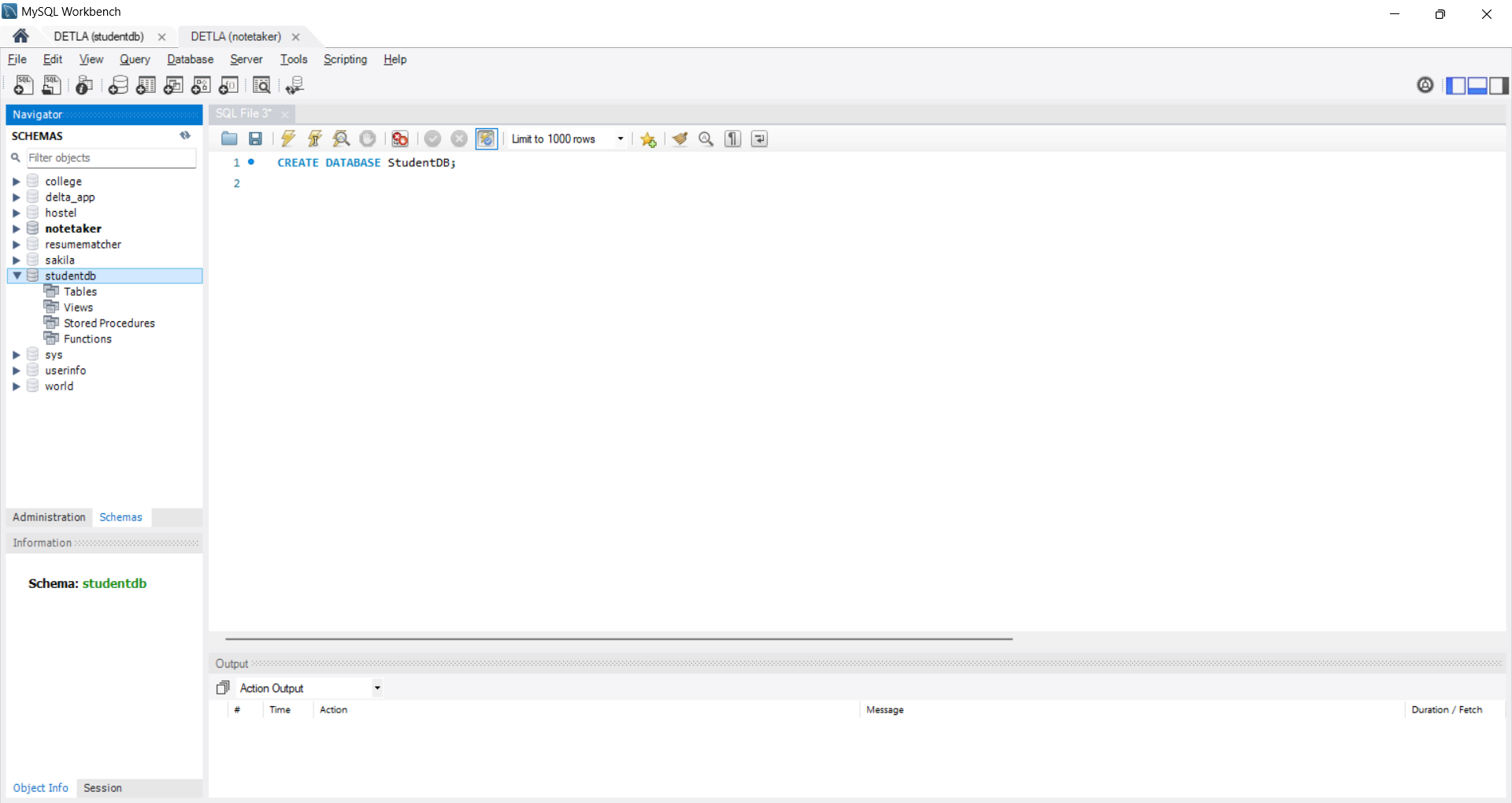
Week 1 Assignments

**Assignment 1: Workbench Basics**

1. Install MySQL Workbench and connect to your MySQL server.



1. Create a new schema StudentDB.
2. Take a screenshot of the Workbench schema navigator with StudentDB visible.



**Assignment 2: Table Operations**

1. Create tables:
   1. Students (student\_id, name, age, gender, course\_id)
   2. Courses (course\_id, course\_name, duration)
   3. Marks (mark\_id, student\_id, subject, score)

CREATE DATABASE StudentDB;

USE StudentDB;

CREATE TABLE Students(

student\_id int,

name varchar(10),

age int,

gender varchar(10),

course\_id int

);

CREATE TABLE Courses(

course\_id int,

course\_name varchar(12),

duration time

);

CREATE TABLE Marks(

mark\_id int,

student\_id int,

subject varchar(10),

score int

);

1. Modify Students table to add a new column email.

ALTER TABLE Students

ADD Email varchar(30);

1. Drop the Marks table and recreate it with the same structure.

DROP TABLE IF EXists Marks;

CREATE TABLE Marks(

mark\_id int,

student\_id int,

subject varchar(10),

score int

);

**Assignment 3: DML Operations**

1. Insert at least 5 rows into each table.

INSERT INTO Students

VALUES

(101,'Aa',21,'FEMALE',01),

(102,'Bb',22,'MALE',01),

(103,'Cc',20,'FEMALE',02),

(104,'Dd',21,'MALE',07),

(105,'Ee',24,'FEMALE',03);

INSERT INTO Courses

VALUES

(01,'java','04:00:00'),

(02,'python','03:00:00'),

(03,'C','03:00:00'),

(04,'html','03:00:00'),

(07,'css','03:00:00');

INSERT INTO Marks

VALUES

(1,101,'English',100),

(2,102,'English',95),

(3,103,'English',80),

(4,104,'English',75),

(5,105,'English',82);

1. Update one student’s course.

SET SQL\_SAFE\_UPDATES = 0;

UPDATE Students

SET course\_id='04'

WHERE

Student\_id='102';

1. Delete a student record.

DELETE FROM Students

WHERE

Student\_id='104';

**Assignment 4: Data Retrieval**

1. Write a query to fetch all students above age 20.

SELECT

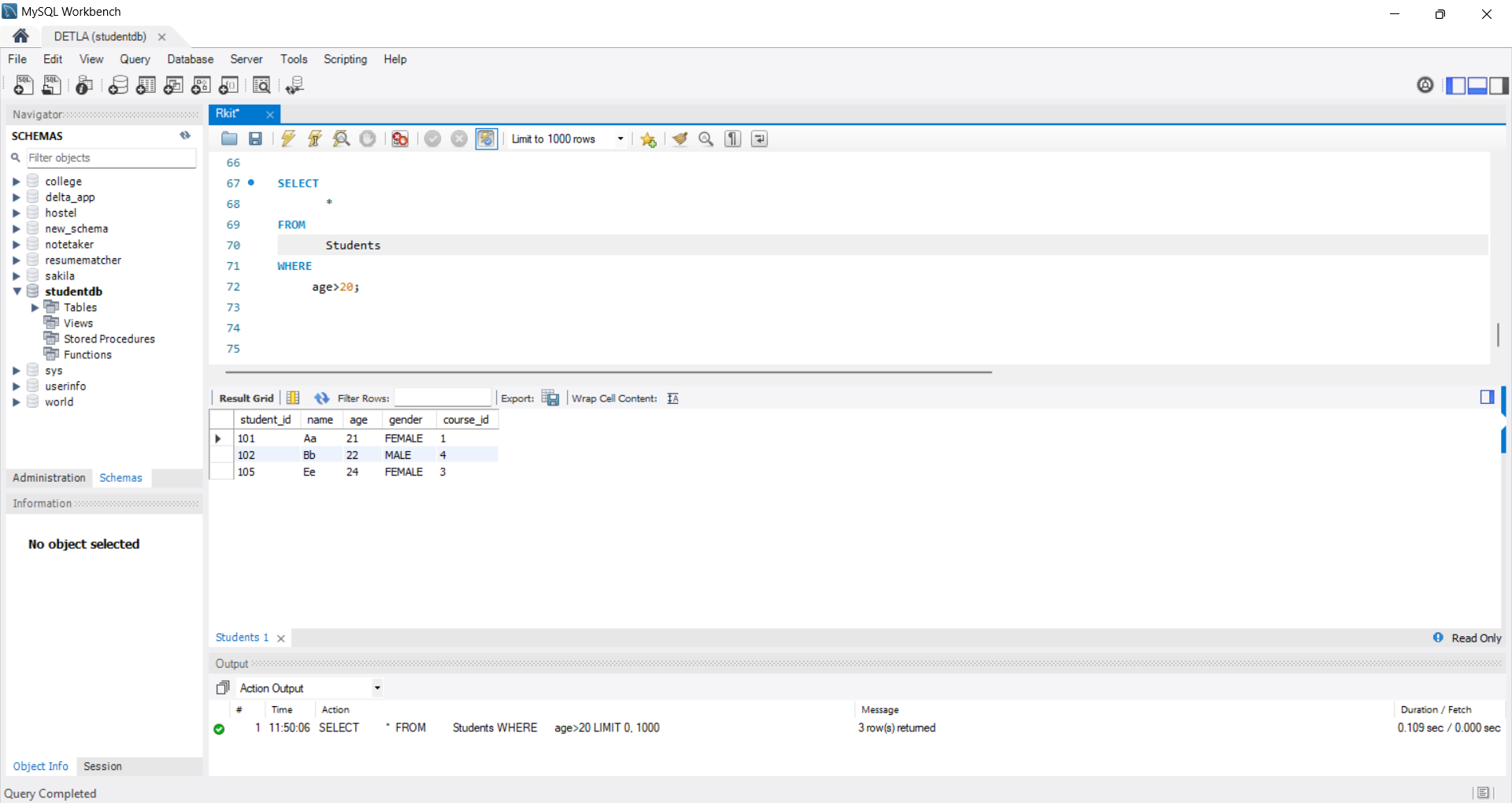
\*

FROM

Students

WHERE

age>20;



1. Fetch all students ordered by name alphabetically.

SELECT

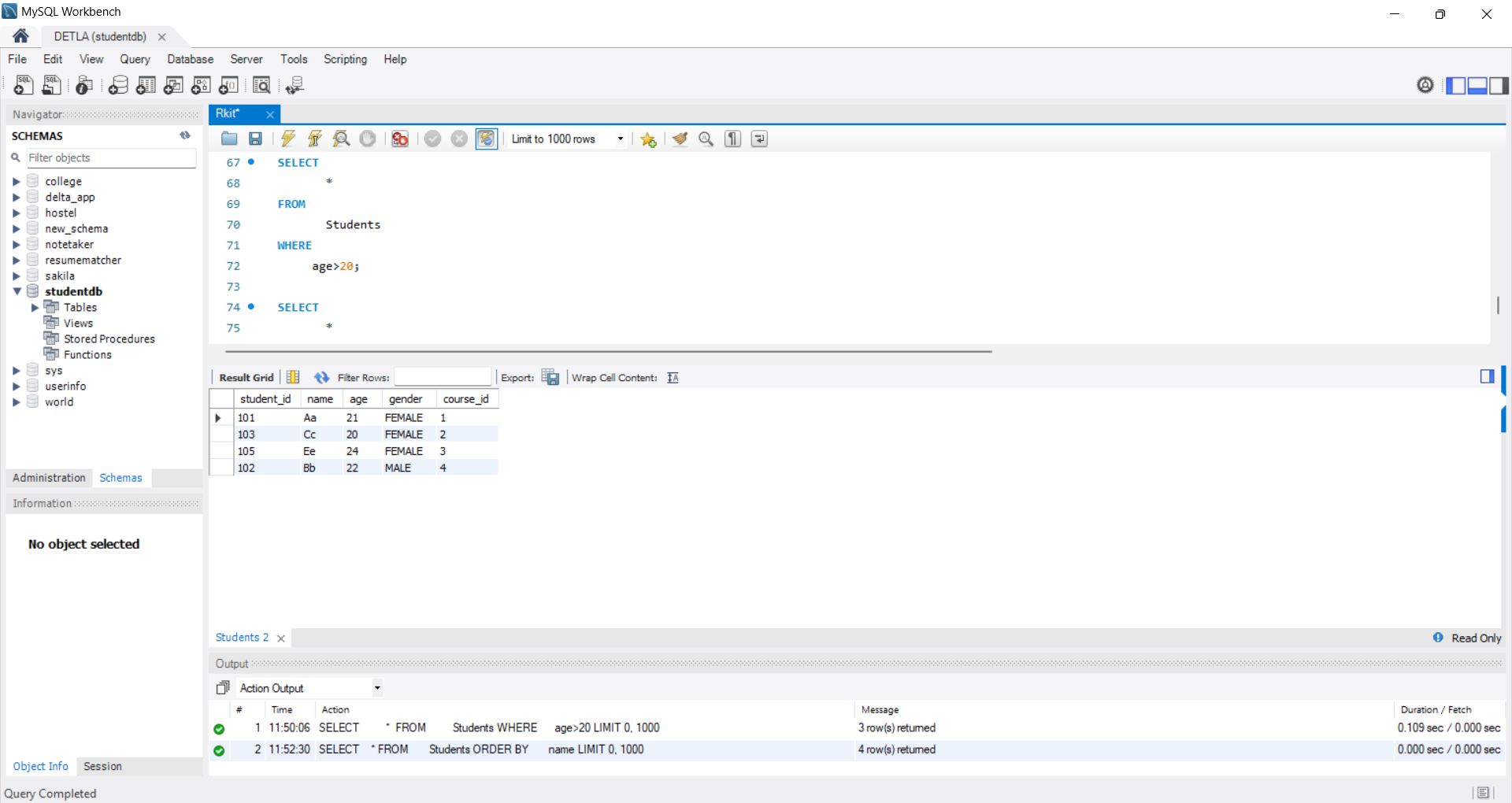
\*

FROM

Students

ORDER BY

name;



1. Show total number of students enrolled in each course using GROUP BY.

SELECT

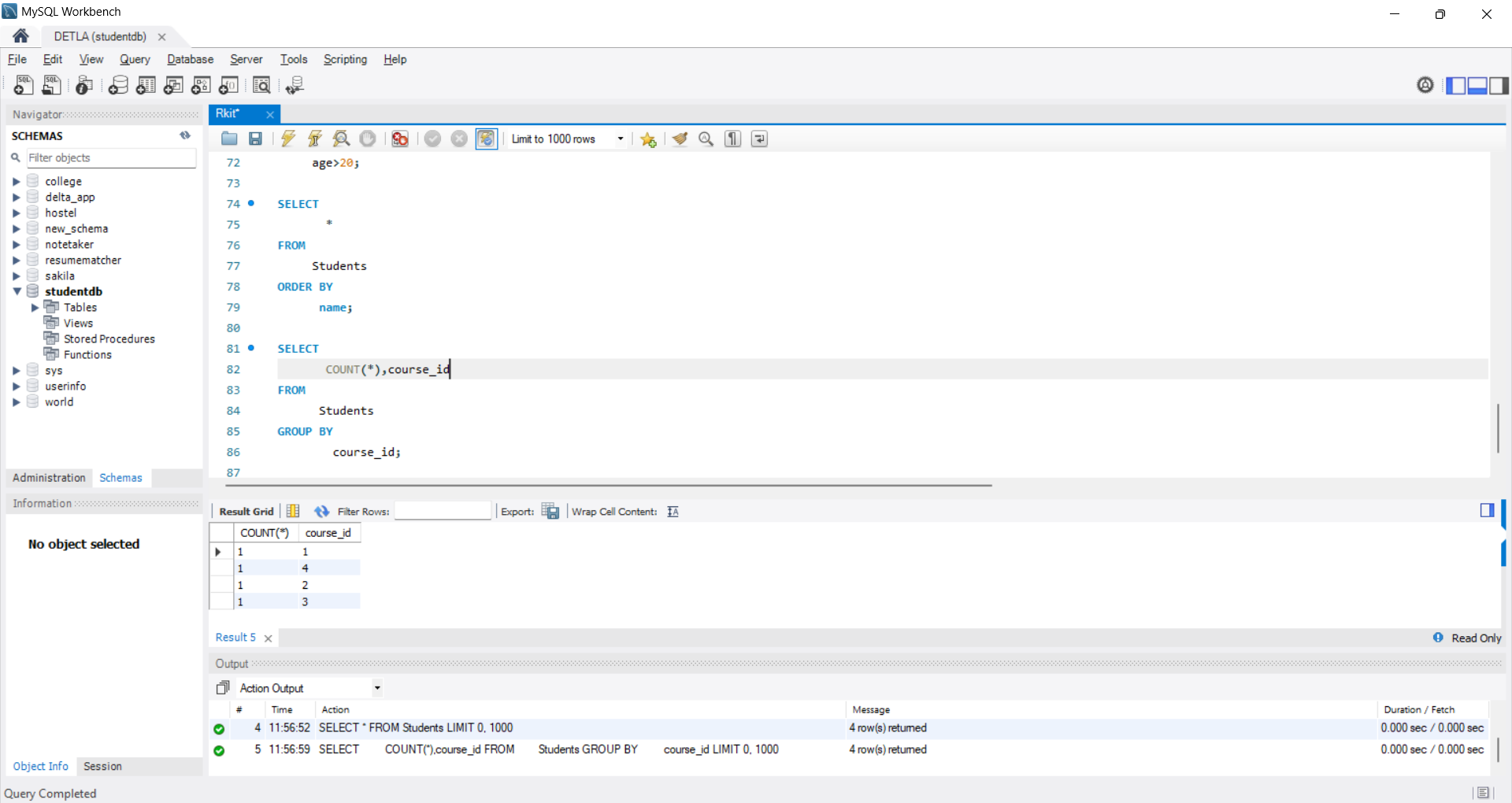
COUNT(\*),course\_id

FROM

Students

GROUP BY

course\_id;



1. Show courses that have more than 2 students using HAVING.

SELECT

course\_id

FROM

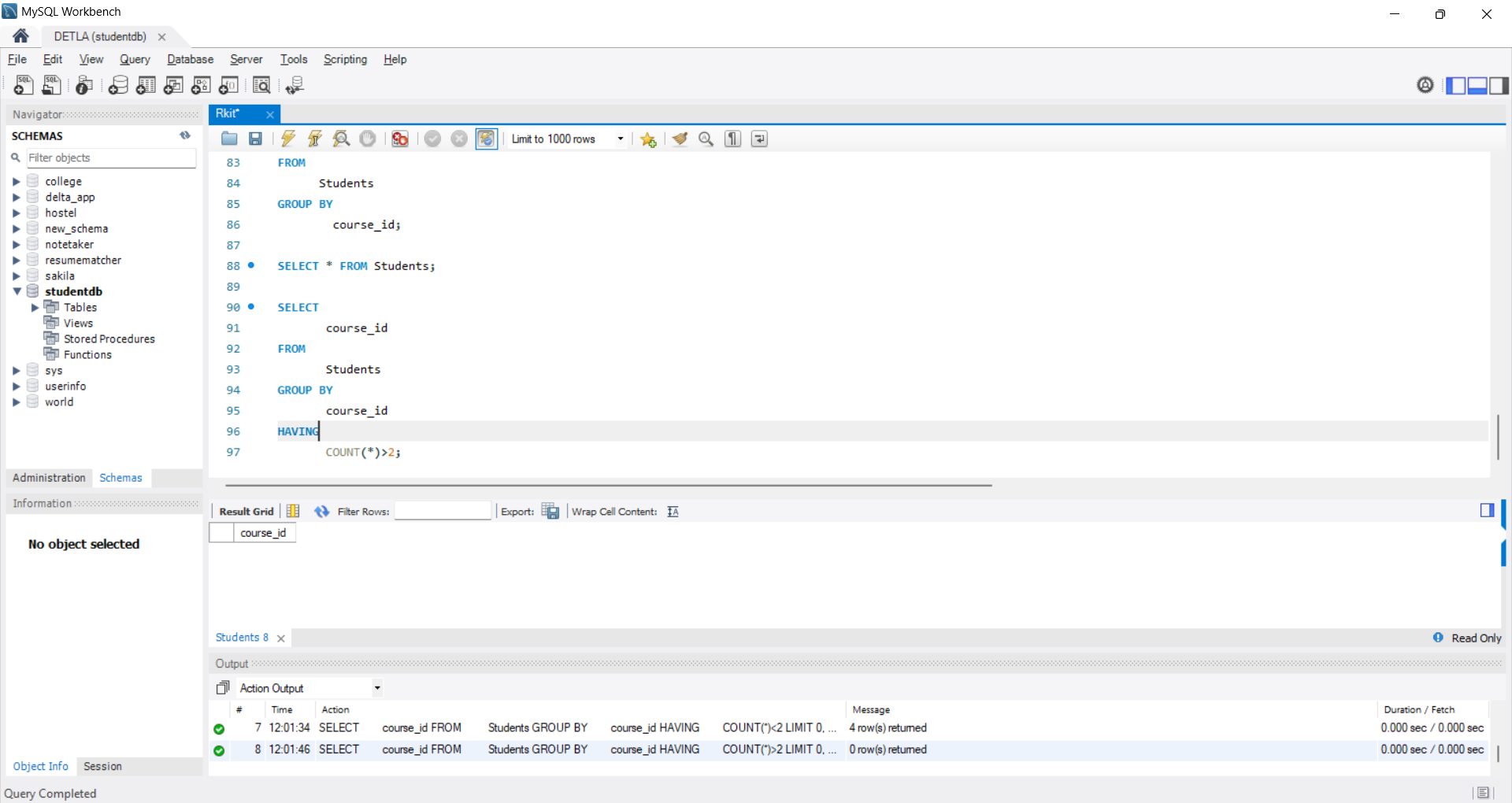
Students

GROUP BY

course\_id

HAVING

COUNT(\*)>2;



**Assignment 5: Joins + Aggregates**

1. Display students with their enrolled course names using INNER JOIN.

SELECT

s.\*,c.course\_name

FROM

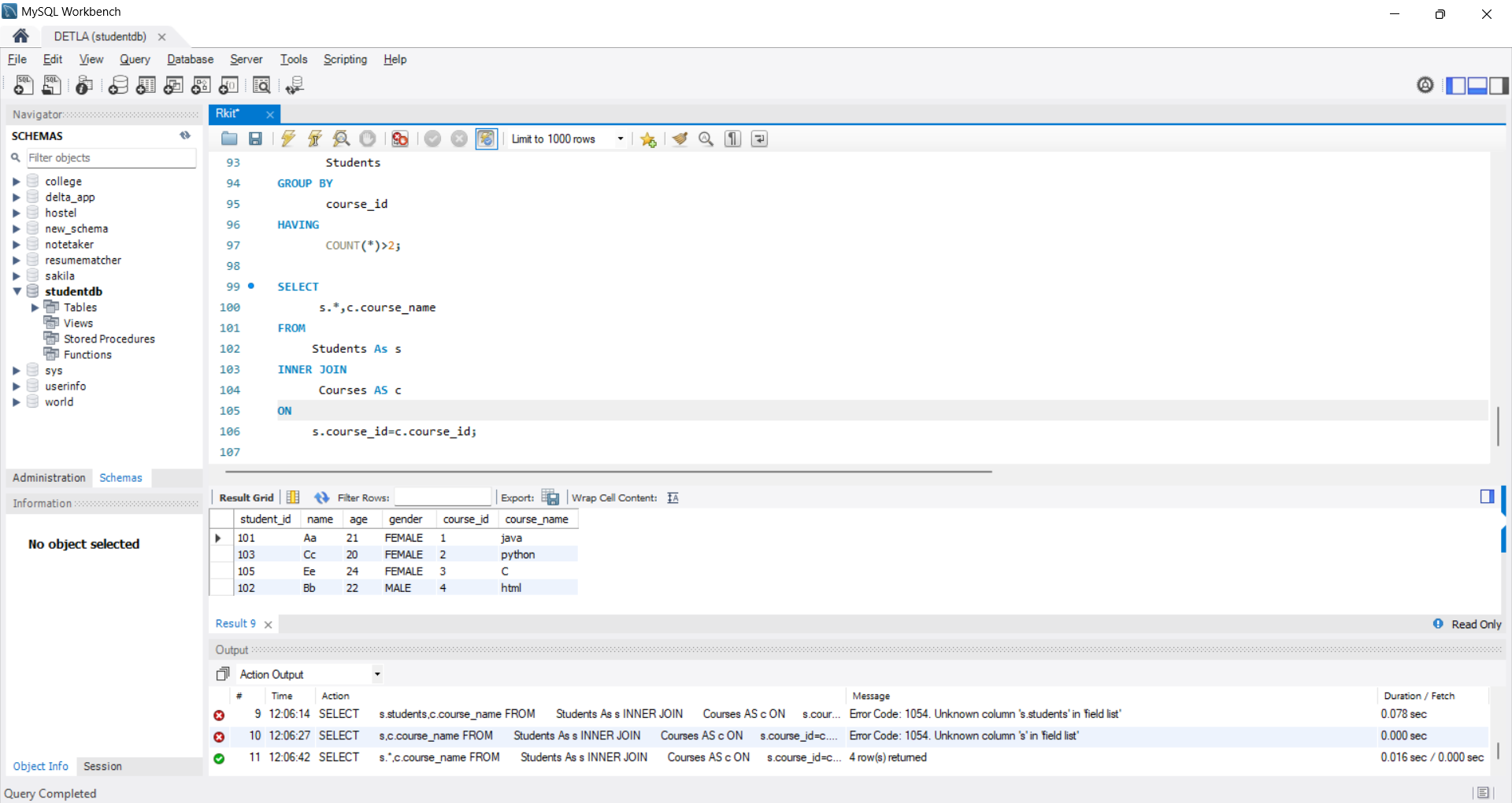
Students As s

INNER JOIN

Courses AS c

ON

s.course\_id=c.course\_id;



1. Display all students even if they are not enrolled in any course (LEFT JOIN).

SELECT

Students.\*

FROM

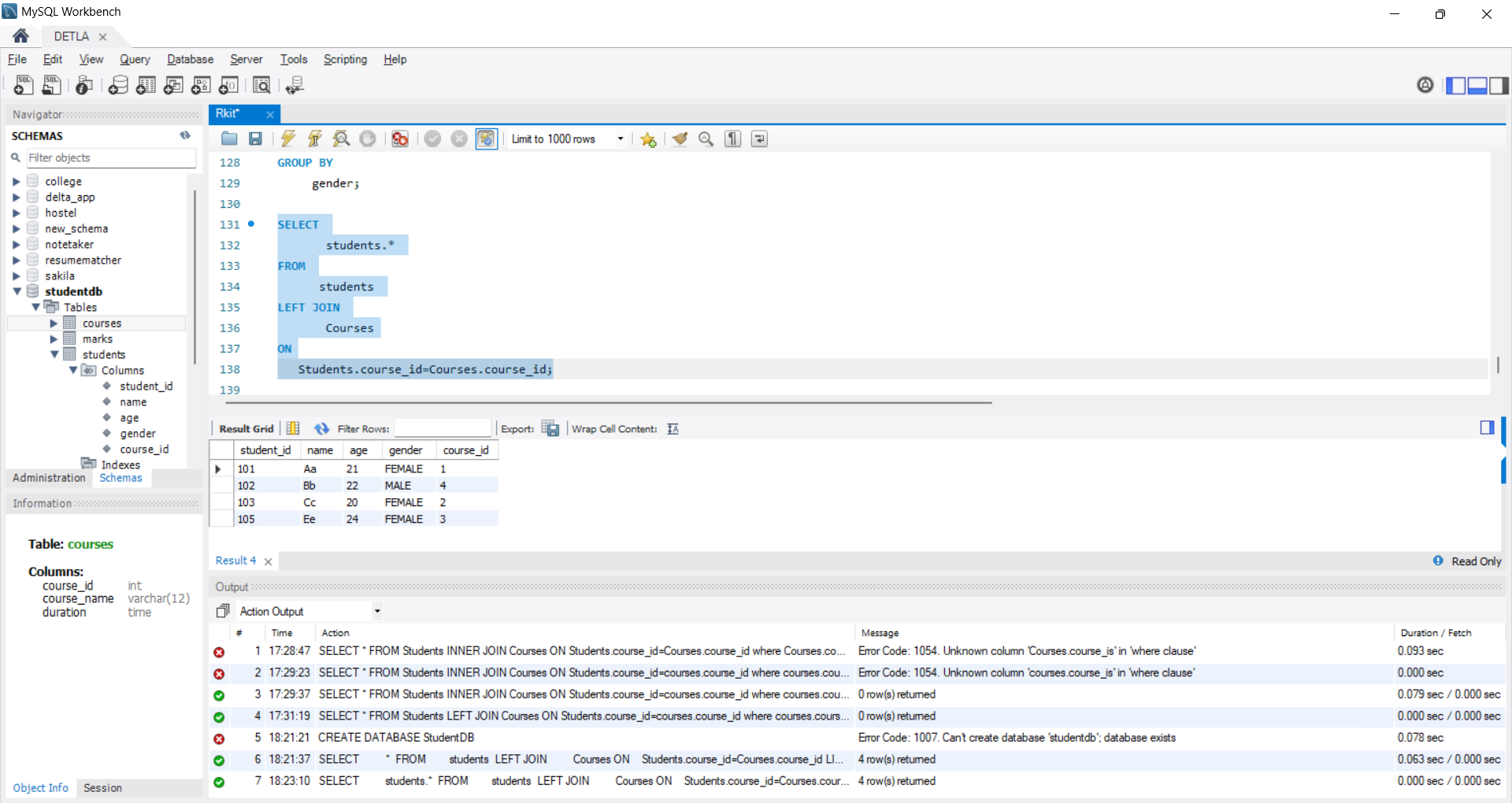
students

LEFT JOIN

Courses

ON

Students.course\_id=Courses.course\_id;



1. Display all courses and their students (RIGHT JOIN).

SELECT

c.\*,s.\*

FROM

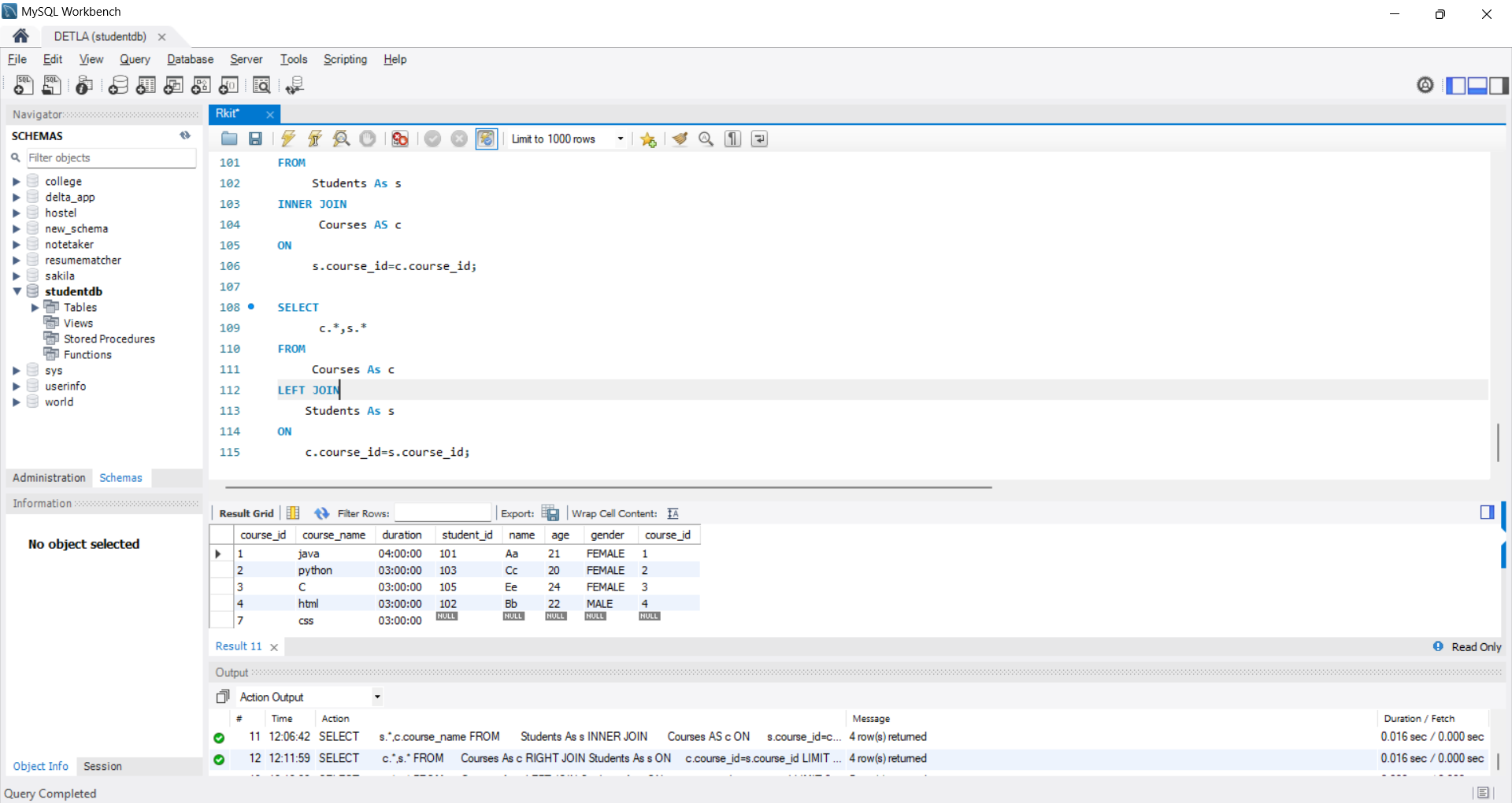
Courses As c

LEFT JOIN

Students As s

ON

c.course\_id=s.course\_id;



1. Find highest, lowest, and average marks per subject.

SELECT

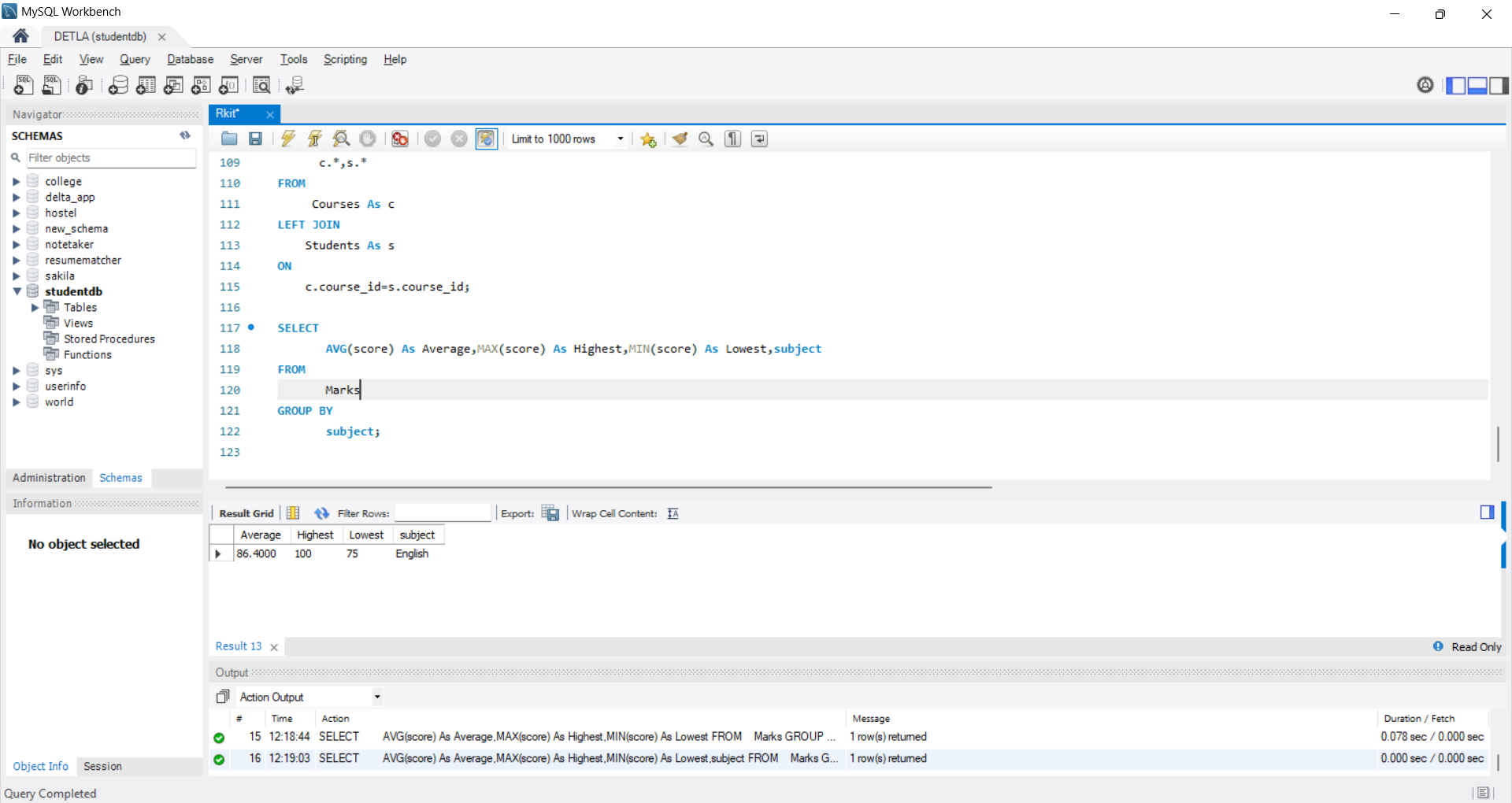
AVG(score) As Average,MAX(score) As Highest,MIN(score) As Lowest,subject

FROM

Marks

GROUP BY

subject;



1. Count how many male and female students exist.

SELECT

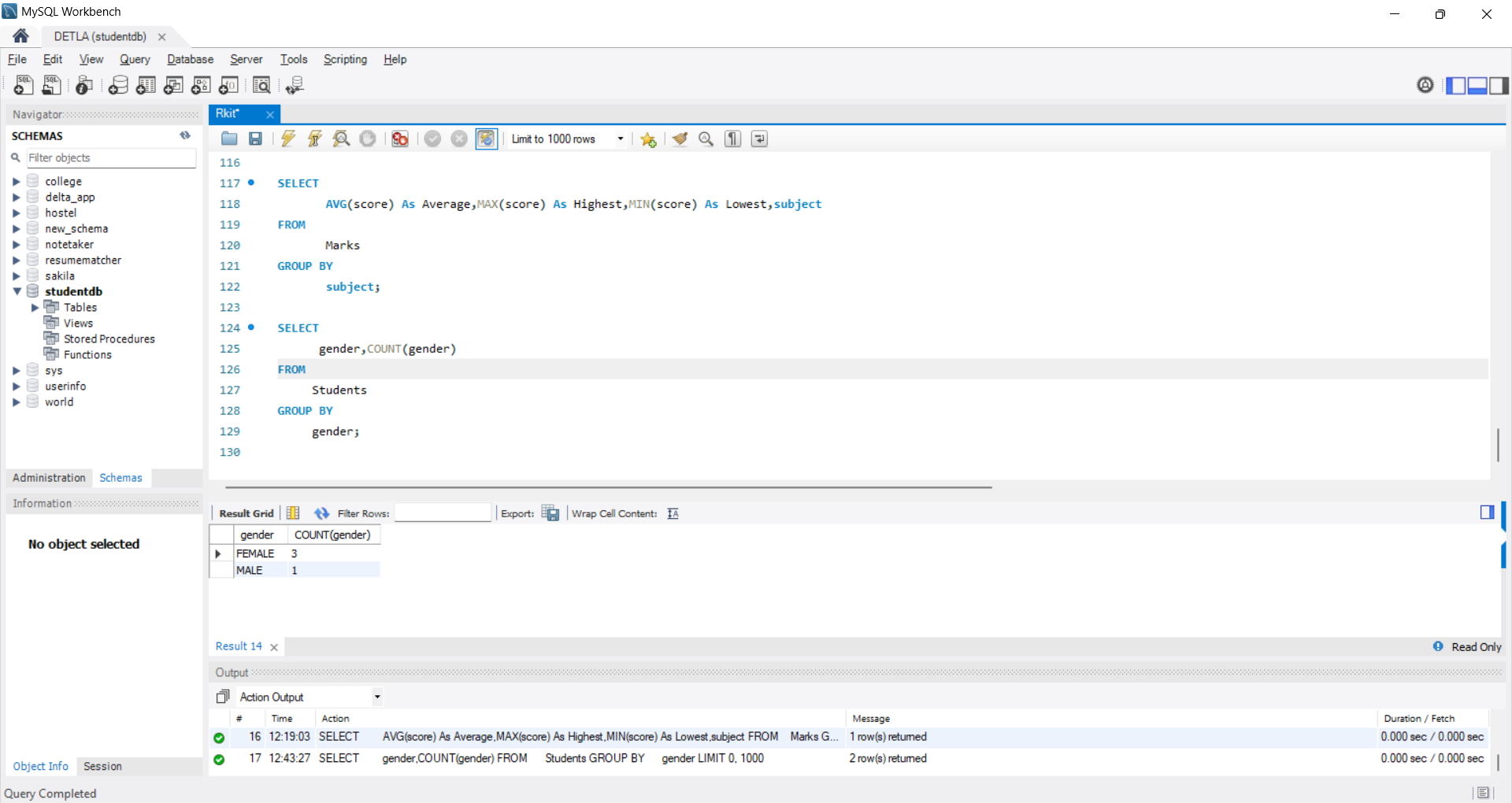
gender,COUNT(gender)

FROM

Students

GROUP BY

gender;



**Test/Quiz for Week 1**: Write SQL queries for a small **LibraryDB** with Books, Authors, and Borrowers tables (to check joins, aggregates, and filtering).

CREATE DATABASE LibraryDB;

USE LibraryDB;

CREATE TABLE Authors (

author\_id INT AUTO\_INCREMENT PRIMARY KEY,

author\_name VARCHAR(50) NOT NULL,

country VARCHAR(50)

);

CREATE TABLE Books (

book\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(200) NOT NULL,

author\_id INT,

published\_year INT,

genre VARCHAR(50),

available\_copies INT DEFAULT 0,

FOREIGN KEY (author\_id) REFERENCES Authors(author\_id)

);

CREATE TABLE Borrowers (

borrower\_id INT AUTO\_INCREMENT PRIMARY KEY,

borrower\_name VARCHAR(100) NOT NULL,

membership\_date DATE

);

INSERT INTO Authors (author\_name, country) VALUES

('J.K. Rowling', 'UK'),

('George Orwell', 'UK'),

('Mark Twain', 'USA');

INSERT INTO Books (title, author\_id, published\_year, genre, available\_copies) VALUES

('Harry Potter', 1, 1997, 'Fantasy', 5),

('1984', 2, 1949, 'Dystopian', 3),

('Animal Farm', 2, 1945, 'Political Satire', 4),

('Adventures of Tom Sawyer', 3, 1876, 'Adventure', 2);

INSERT INTO Borrowers (borrower\_name, membership\_date) VALUES

('Alice', '2022-01-15'),

('Bob', '2022-02-10'),

('Charlie', '2022-03-05');

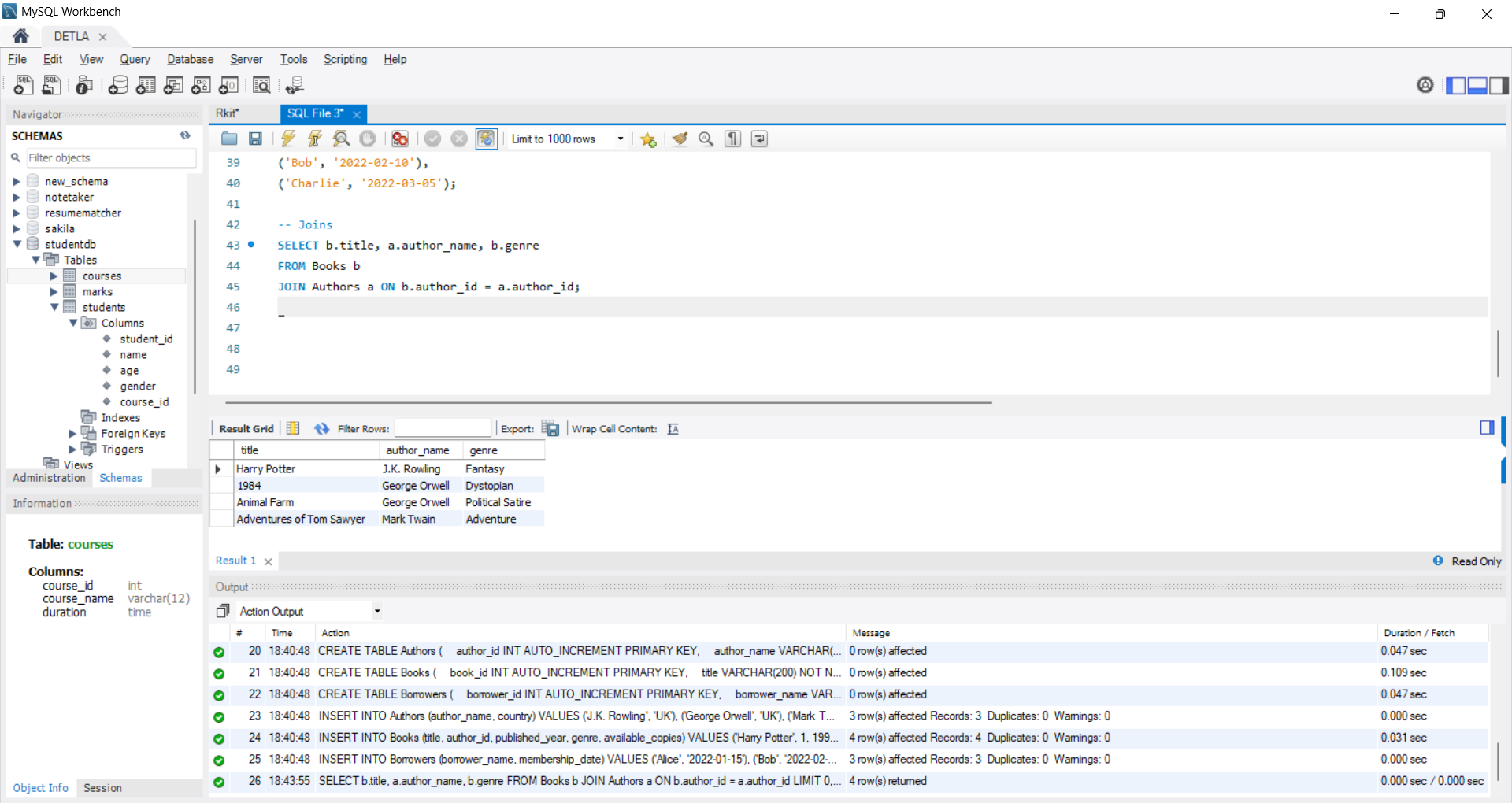
Queries for joins:

List all books with their authors:

SELECT b.title, a.author\_name, b.genre

FROM Books b

JOIN Authors a ON b.author\_id = a.author\_id;



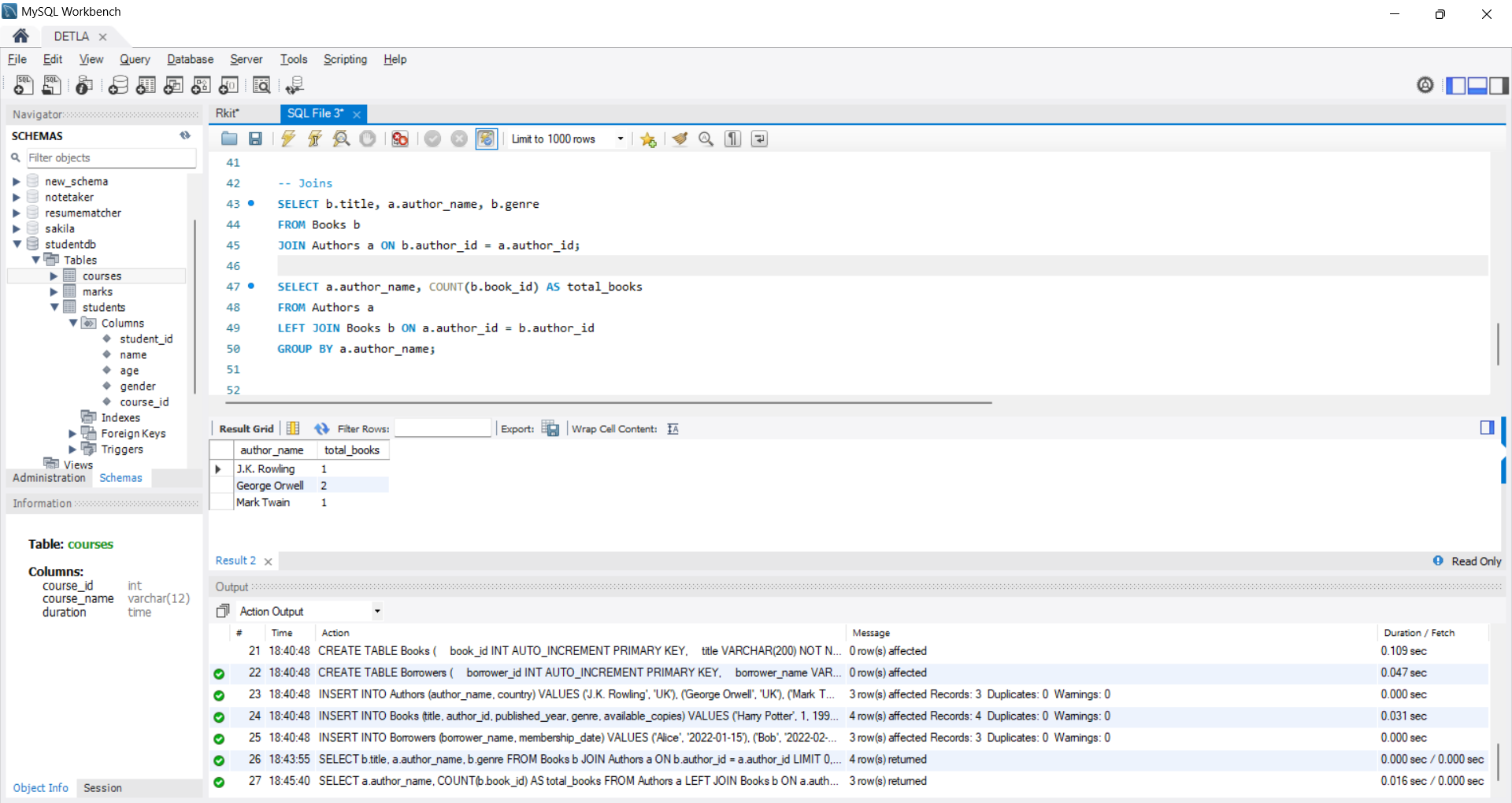
Count how many books each author has written:

SELECT a.author\_name, COUNT(b.book\_id) AS total\_books

FROM Authors a

LEFT JOIN Books b ON a.author\_id = b.author\_id

GROUP BY a.author\_name;



Find the average available copies of books per genre:

SELECT genre, AVG(available\_copies) AS avg\_copies

FROM Books

GROUP BY genre;

