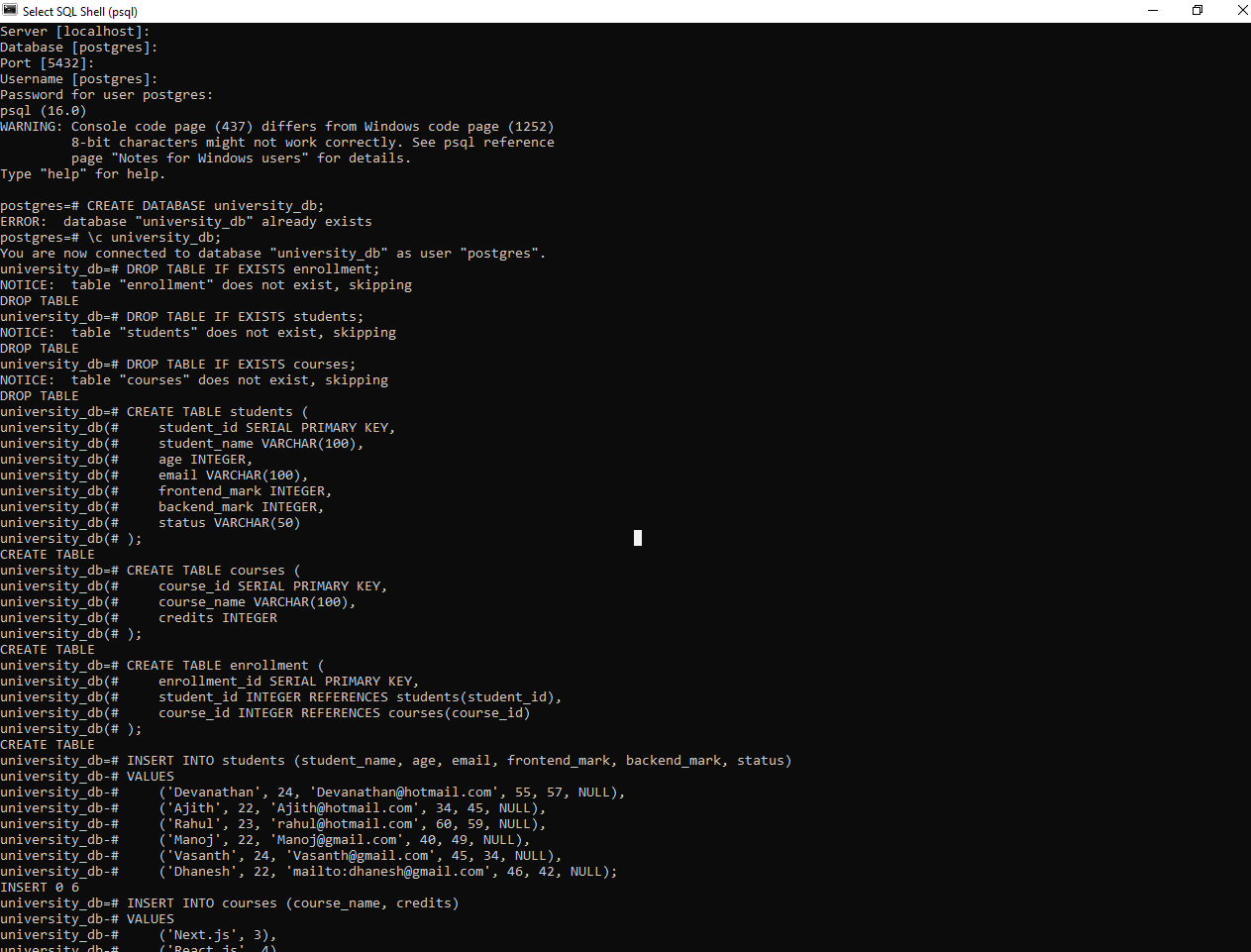
PSQL ASSIGNMENT -1

-- Create database if not exists

CREATE DATABASE IF NOT EXISTS university\_db;

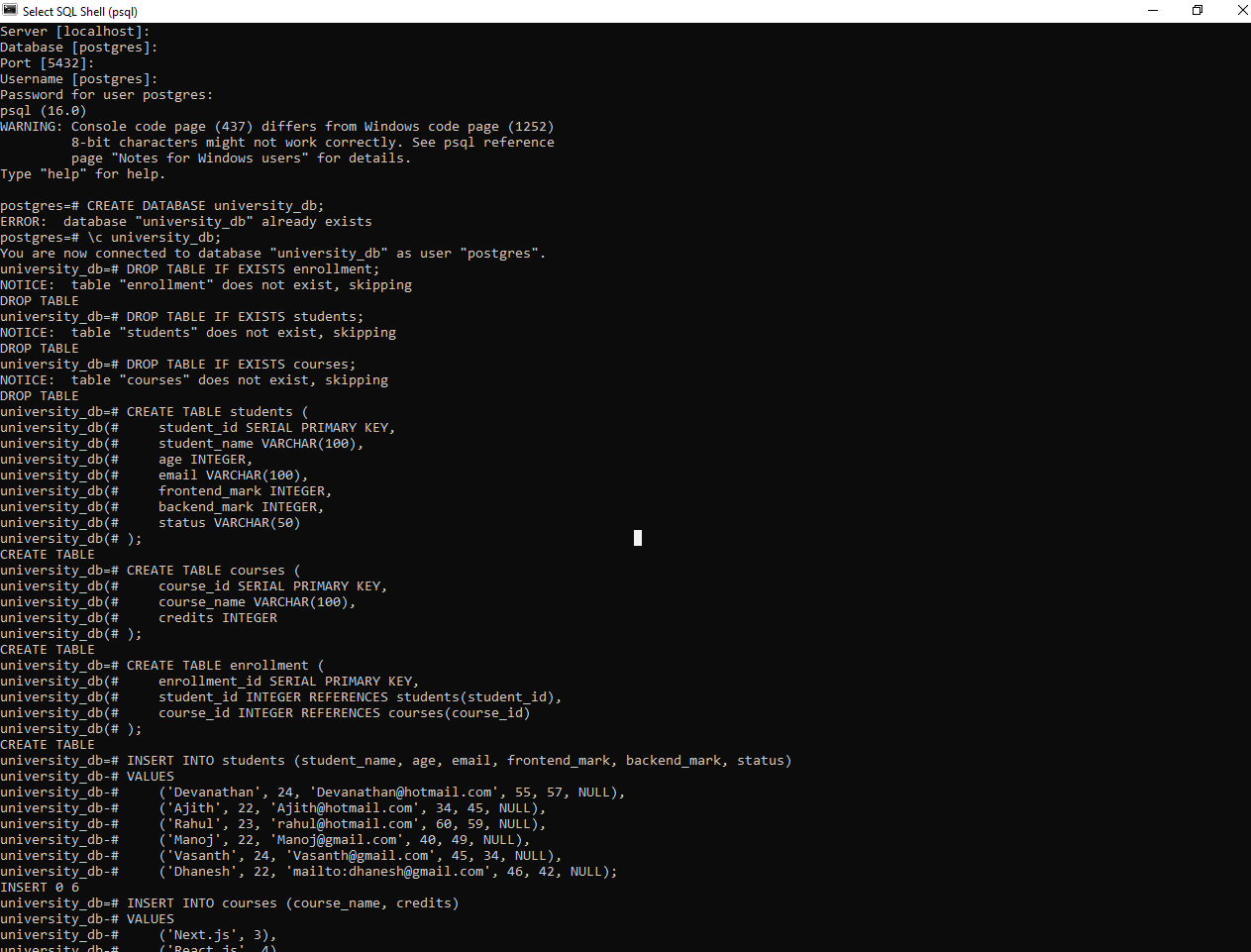
\c university\_db;



-- Drop tables if they exist to start fresh

DROP TABLE IF EXISTS enrollment;

DROP TABLE IF EXISTS students;

DROP TABLE IF EXISTS courses;

-- Create students table

CREATE TABLE students (

student\_id SERIAL PRIMARY KEY,

student\_name VARCHAR(100),

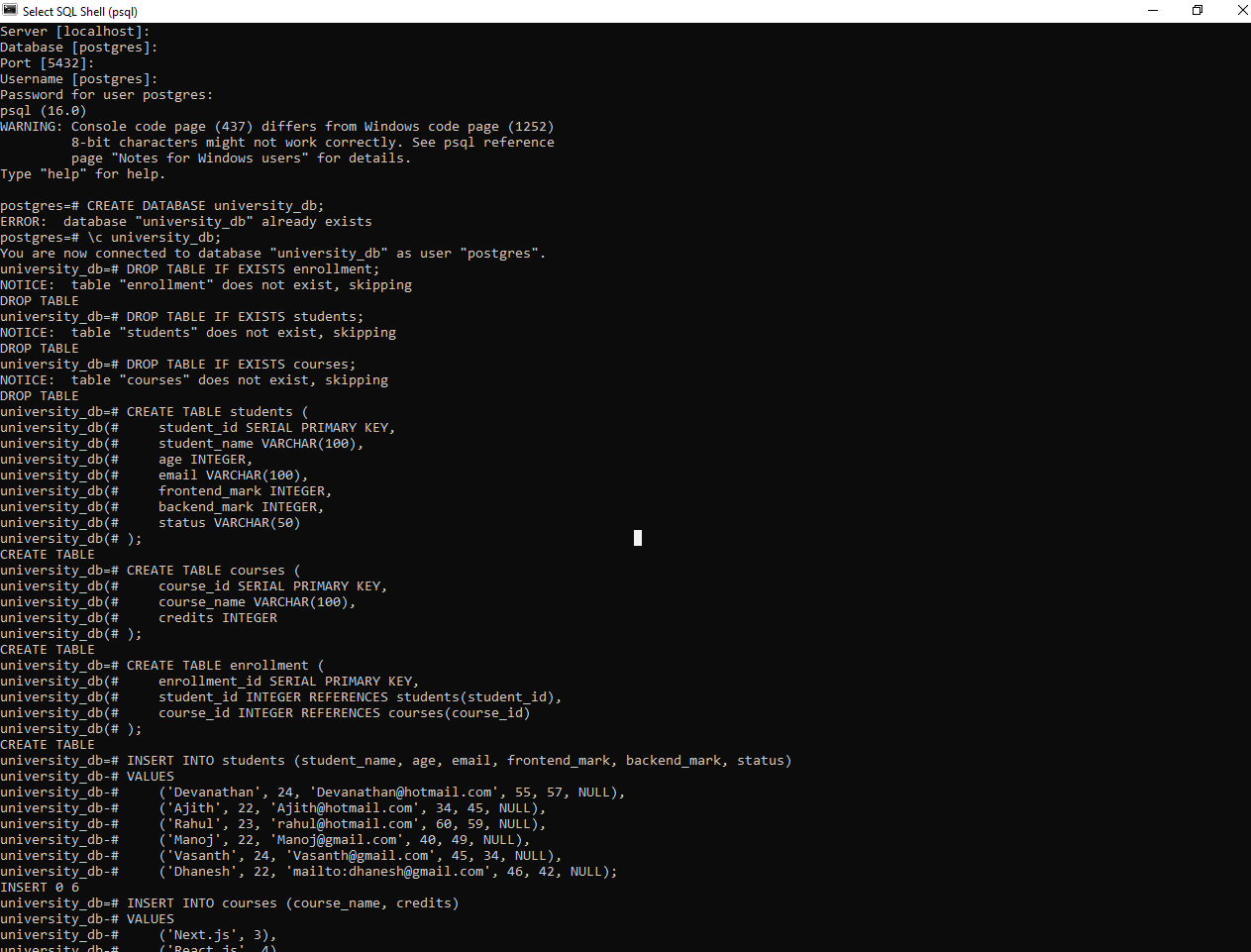
age INTEGER,

email VARCHAR(100),

frontend\_mark INTEGER,

backend\_mark INTEGER,

status VARCHAR(50)

);

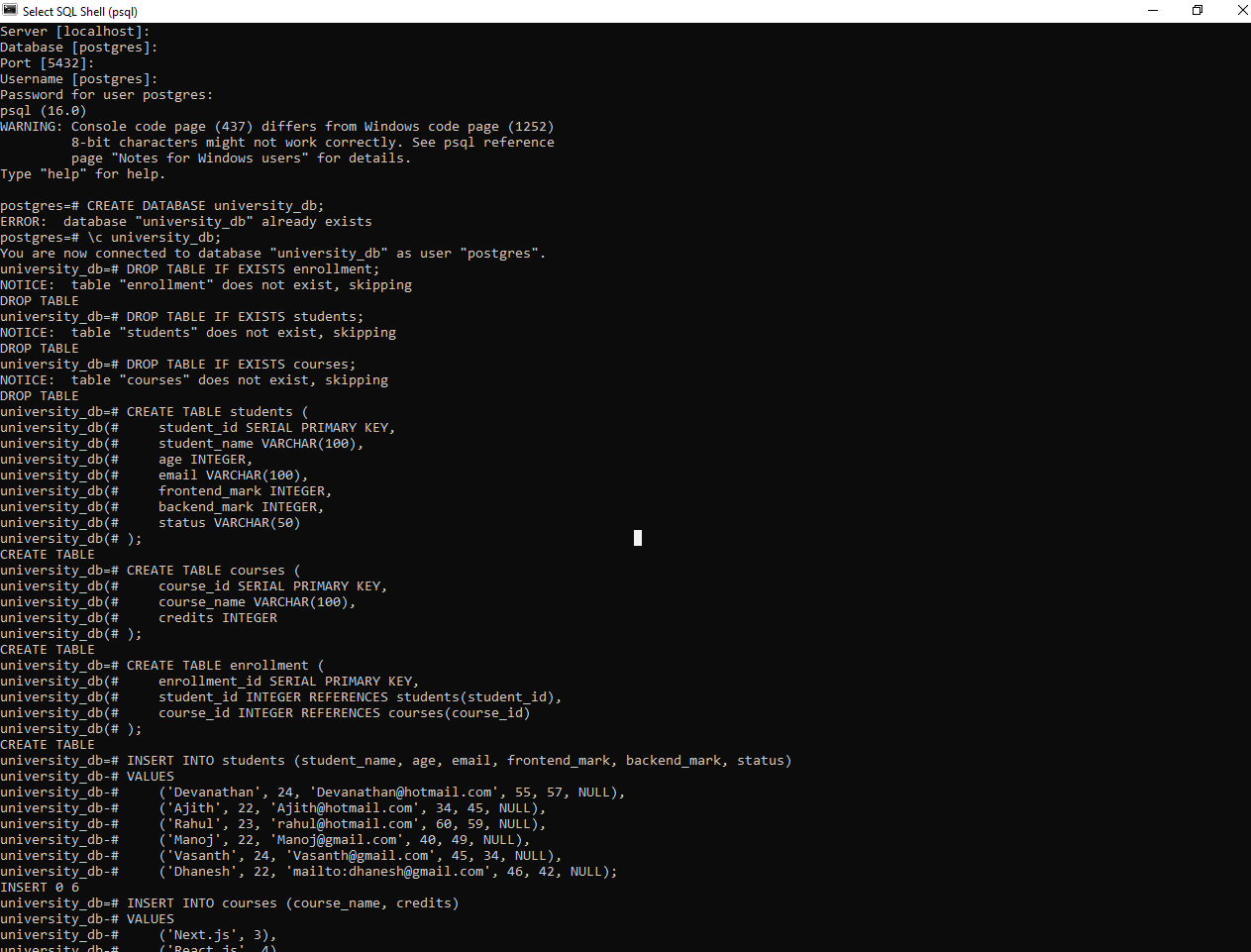
-- Create courses table

CREATE TABLE courses (

course\_id SERIAL PRIMARY KEY,

course\_name VARCHAR(100),

credits INTEGER

);

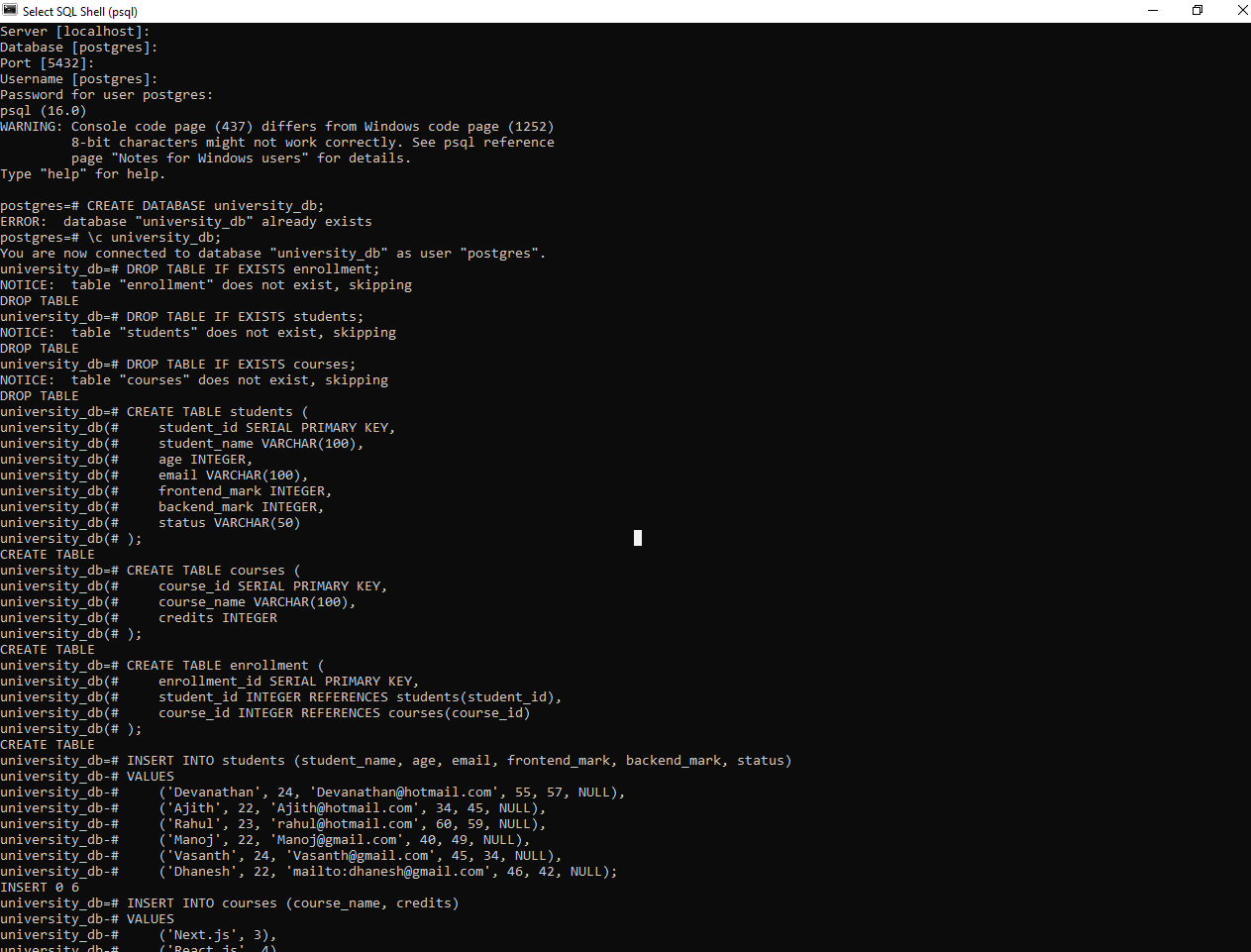
-- Create enrollment table

CREATE TABLE enrollment (

enrollment\_id SERIAL PRIMARY KEY,

student\_id INTEGER REFERENCES students(student\_id),

course\_id INTEGER REFERENCES courses(course\_id)

);

-- Insert sample data into students table

INSERT INTO students (student\_name, age, email, frontend\_mark, backend\_mark, status)

VALUES

('Devanathan', 24, 'Devanathan@hotmail.com', 55, 57, NULL),

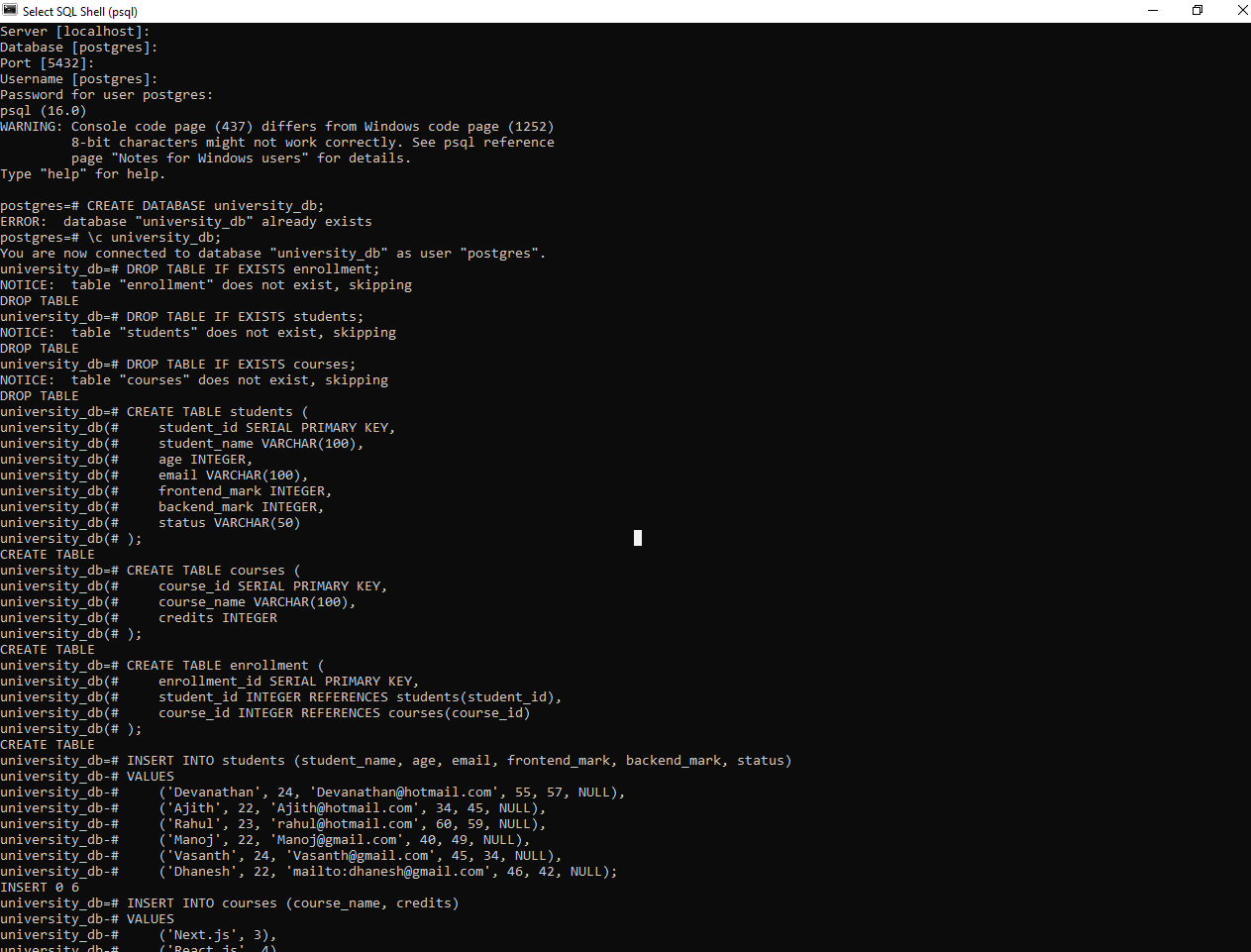
('Ajith', 22, 'Ajith@hotmail.com', 34, 45, NULL),

('Rahul', 23, 'rahul@hotmail.com', 60, 59, NULL),

('Manoj', 22, 'Manoj@gmail.com', 40, 49, NULL),

('Vasanth', 24, 'Vasanth@gmail.com', 45, 34, NULL),

('Dhanesh', 22, 'mailto:dhanesh@gmail.com', 46, 42, NULL);



-- Insert sample data into courses table

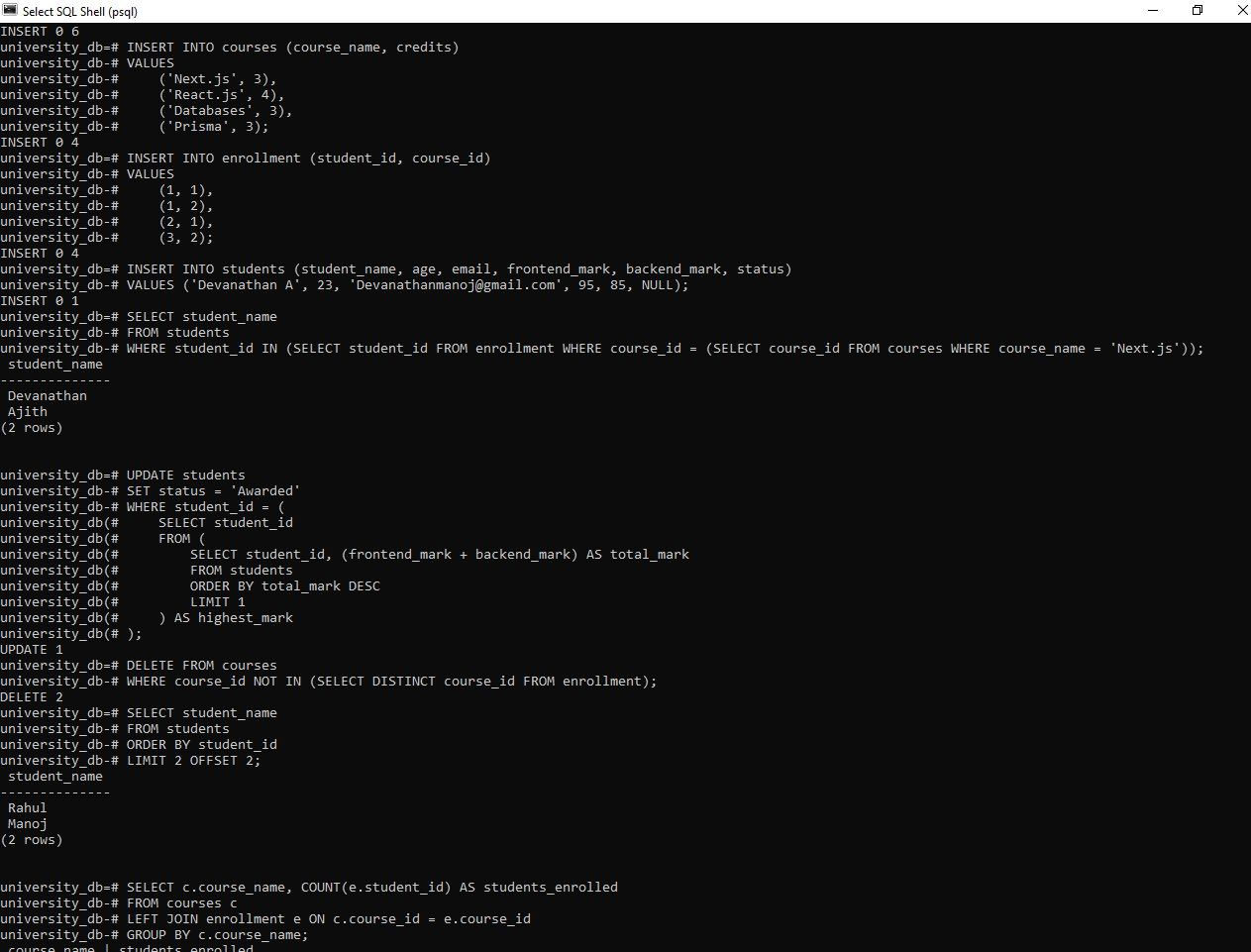
INSERT INTO courses (course\_name, credits)

VALUES

('Next.js', 3),

('React.js', 4),

('Databases', 3),

 ('Prisma', 3);

-- Insert sample data into enrollment table

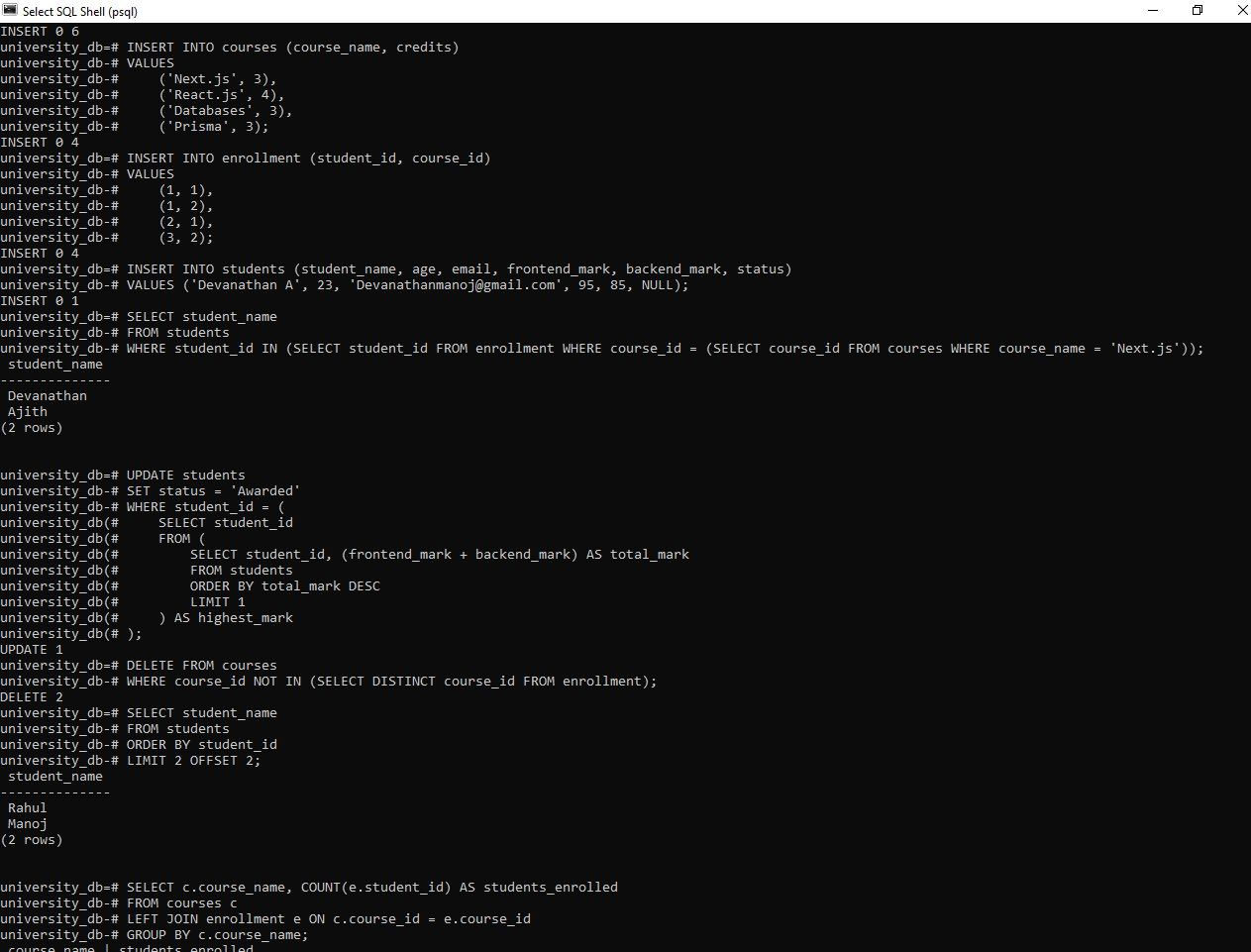
INSERT INTO enrollment (student\_id, course\_id)

VALUES

(1, 1),

(1, 2),

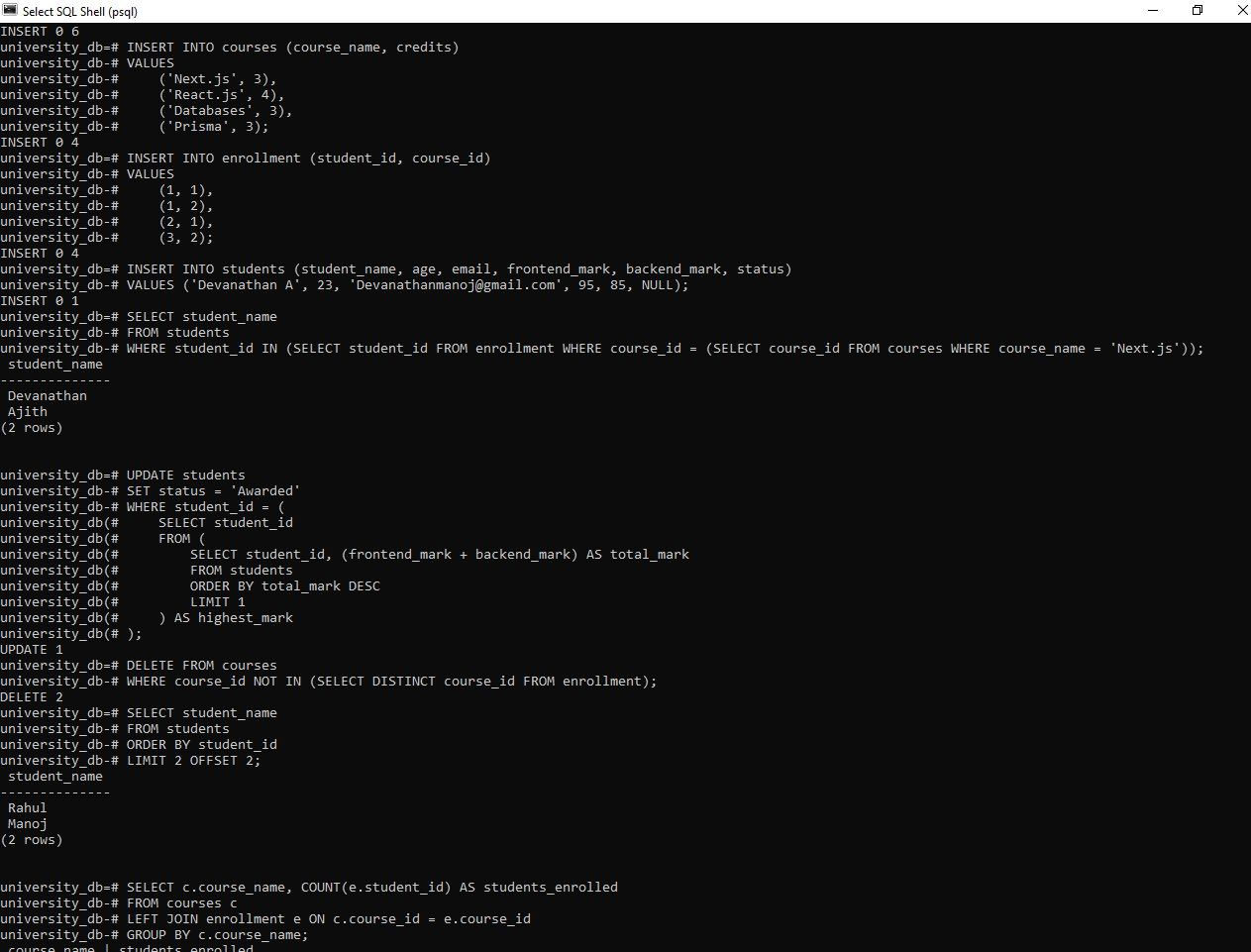
(2, 1),

 (3, 2);

-- Query 1: Insert a new student record

-- Replace with your actual details

INSERT INTO students (student\_name, age, email, frontend\_mark, backend\_mark, status)

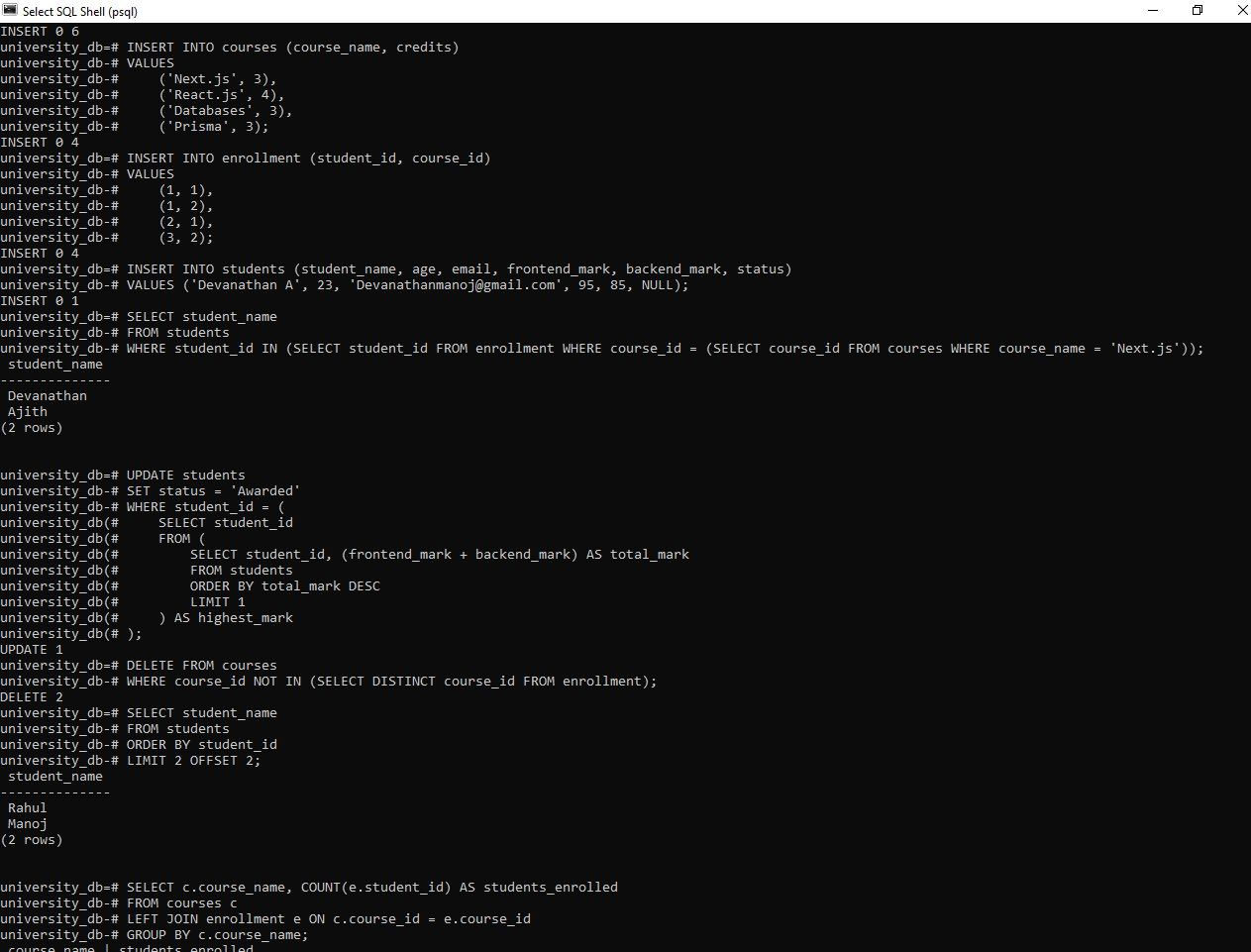
VALUES ('Devanathan A', 23, 'Devanathanmanoj@gmail.com', 95, 85, NULL);

-- Query 2: Retrieve names of students enrolled in 'Next.js'

SELECT student\_name

FROM students

WHERE student\_id IN (SELECT student\_id FROM enrollment WHERE course\_id = (SELECT course\_id FROM courses WHERE course\_name = 'Next.js'));



-- Query 3: Update status of student with highest total mark to 'Awarded'

UPDATE students

SET status = 'Awarded'

WHERE student\_id = (

SELECT student\_id

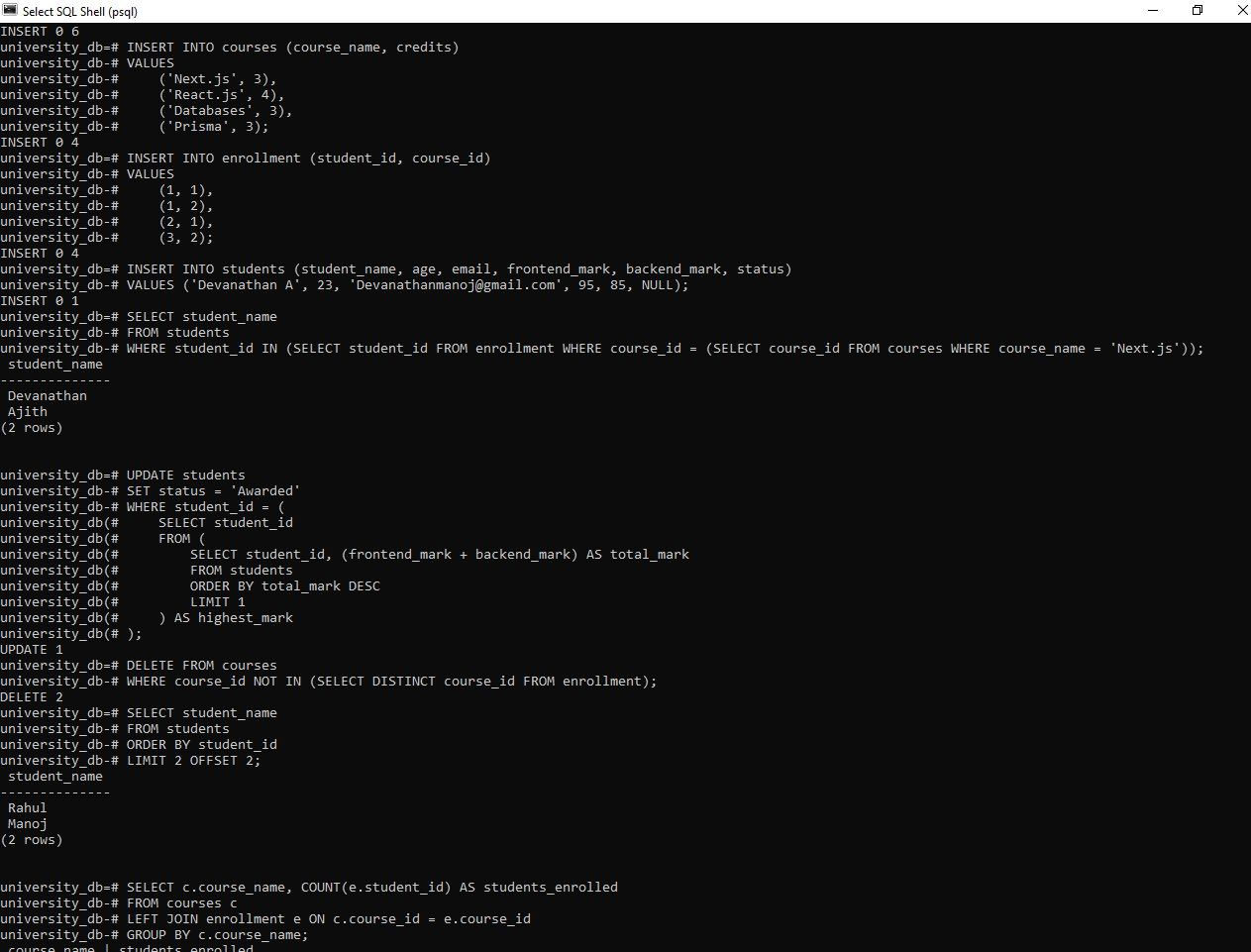
FROM (

SELECT student\_id, (frontend\_mark + backend\_mark) AS total\_mark

FROM students

ORDER BY total\_mark DESC

LIMIT 1

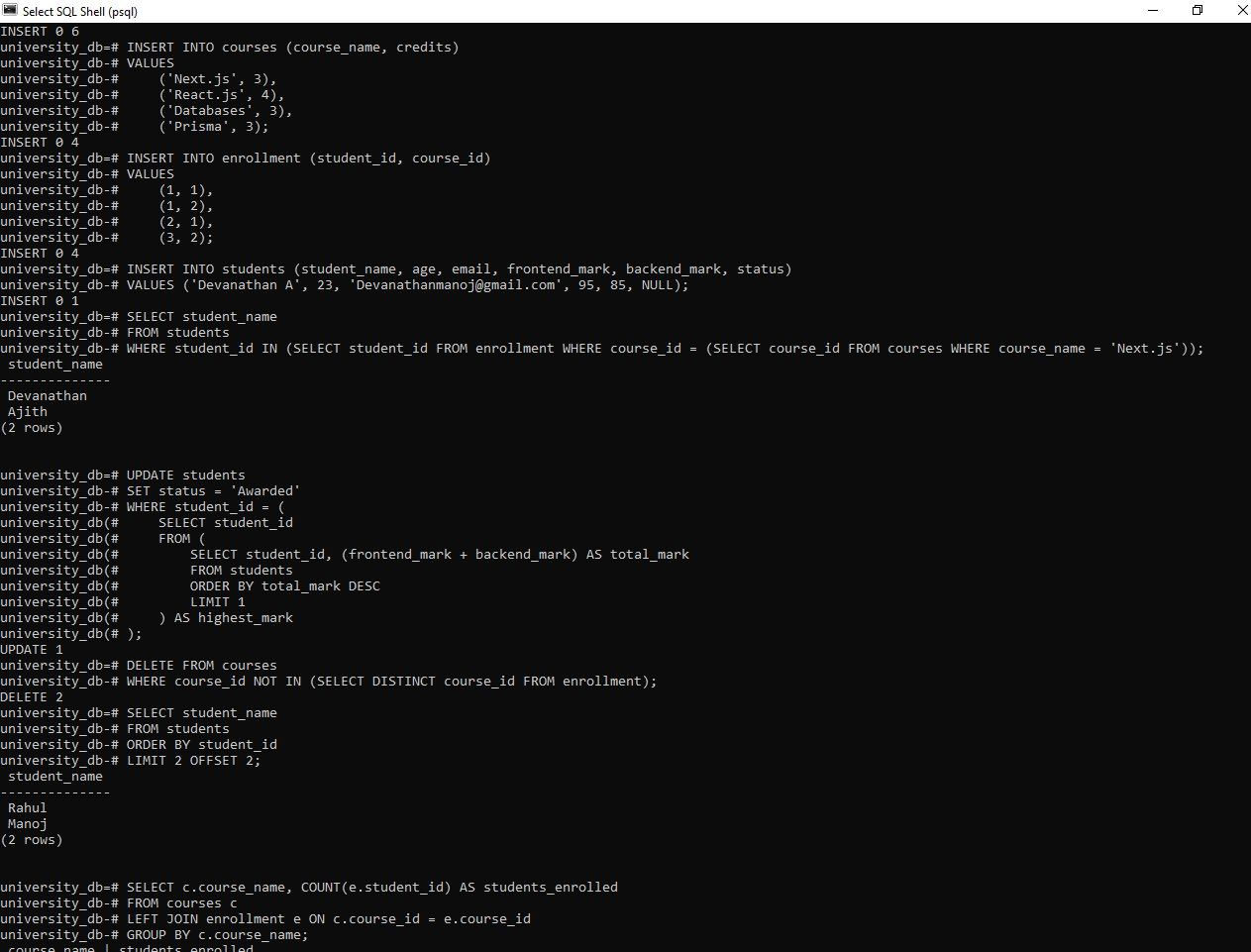
 ) AS highest\_mark

);

-- Query 4: Delete all courses that have no students enrolled

DELETE FROM courses

WHERE course\_id NOT IN (SELECT DISTINCT course\_id FROM enrollment);

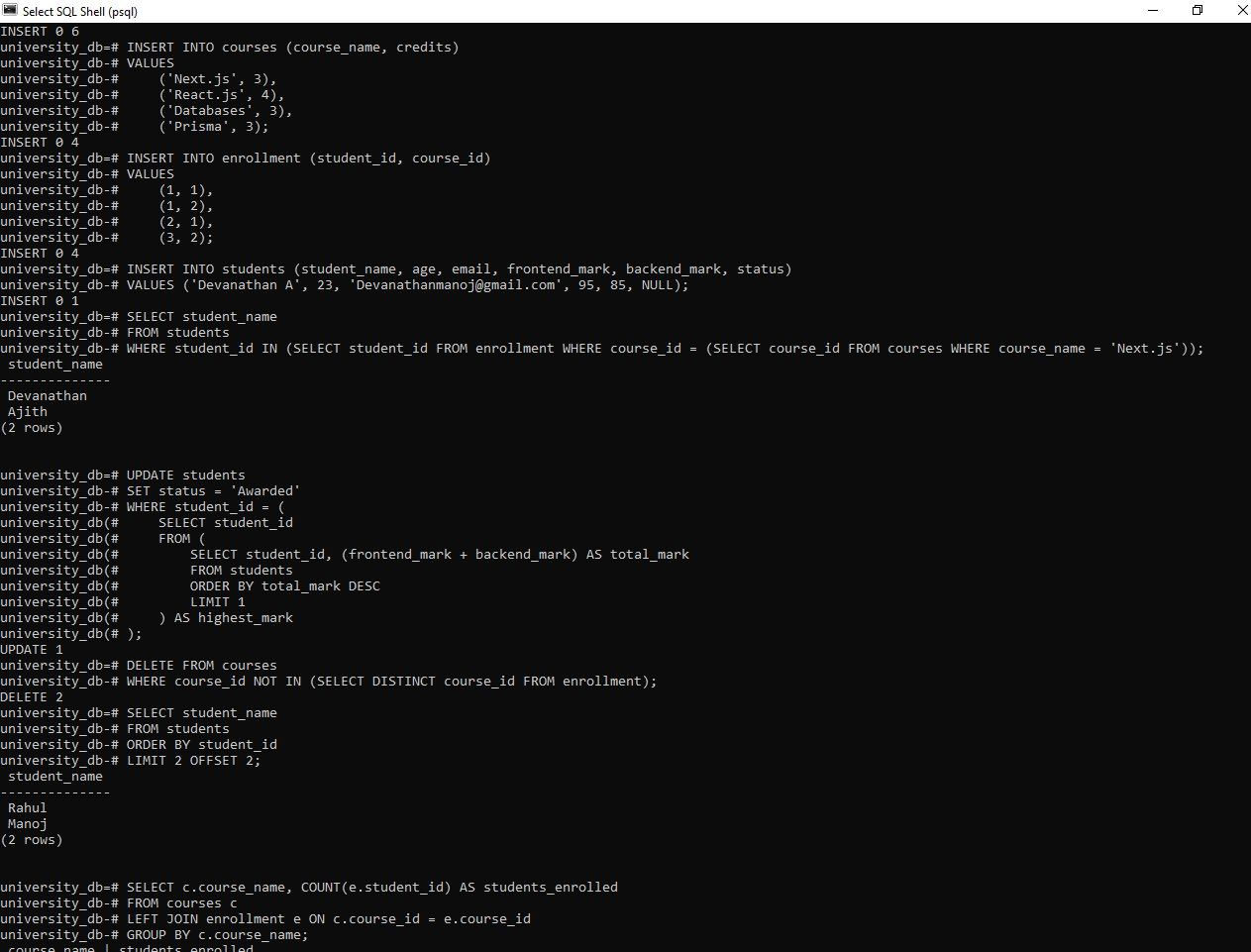


-- Query 5: Retrieve names of students using LIMIT and OFFSET

SELECT student\_name

FROM students

ORDER BY student\_id

LIMIT 2 OFFSET 2;

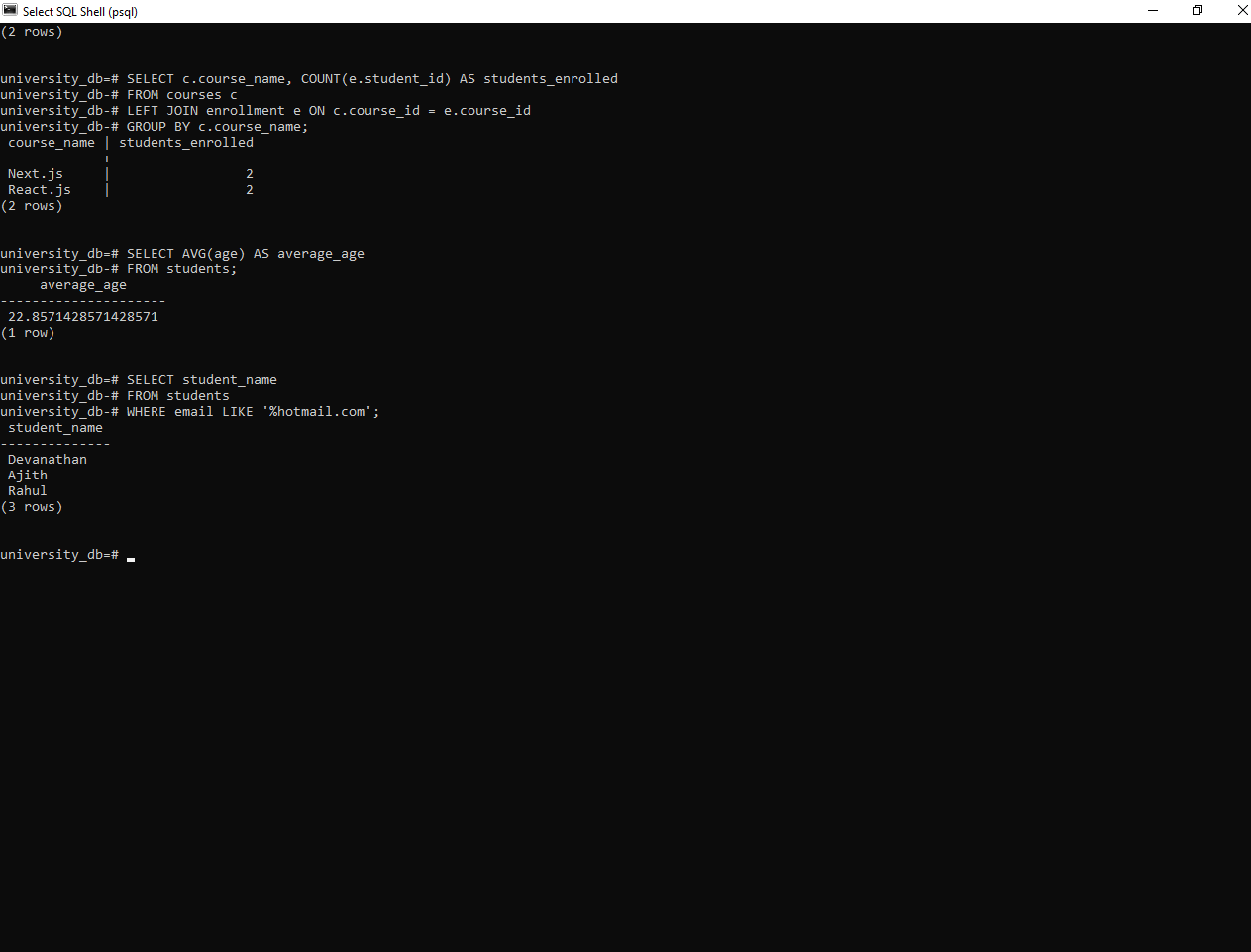
-- Query 6: Retrieve course names and number of students enrolled in each course

SELECT c.course\_name, COUNT(e.student\_id) AS students\_enrolled

FROM courses c

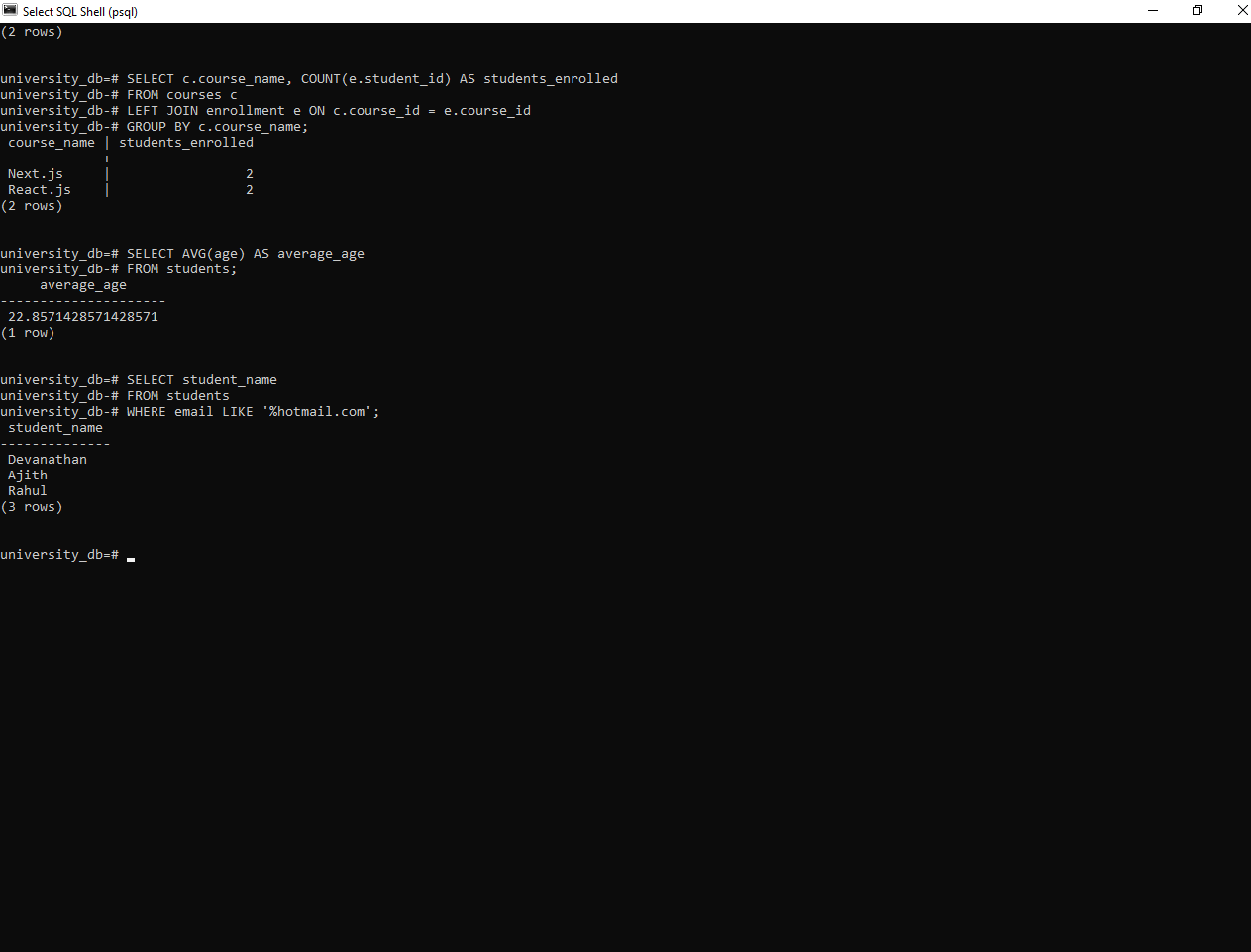
LEFT JOIN enrollment e ON c.course\_id = e.course\_id

GROUP BY c.course\_name;



-- Query 7: Calculate and display average age of all students

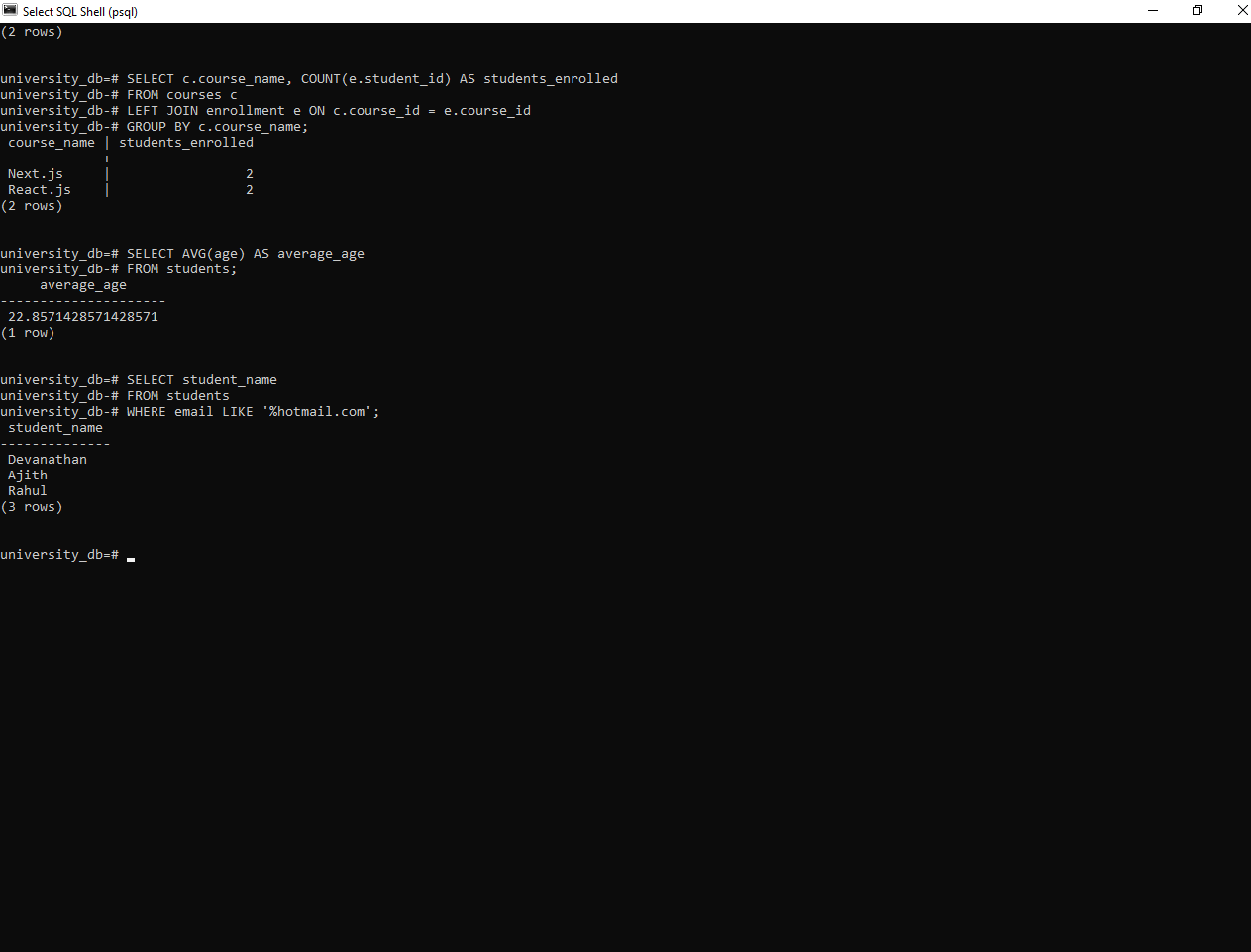
SELECT AVG(age) AS average\_age

FROM students;

-- Query 8: Retrieve names of students whose email addresses contain 'hotmail.com'

SELECT student\_name

FROM students

WHERE email LIKE '%hotmail.com';