Postgres Assignment

Database setup:

psql –U postgres

Table creation :

DROP DATABASE IF EXISTS university\_db;

CREATE DATABASE university\_db;

\c university\_db

Table creation :

create table students (

student\_id integer unique primary key ,

student\_name varchar(100),

age integer,

email text,

frontend\_mark integer,

backend\_mark integer,

status text

);

create table courses (

course\_id integer unique primary key,

course\_name varchar(100),

credits integer

);

create table enrollment (

enrollment\_id INTEGER UNIQUE PRIMARY KEY,

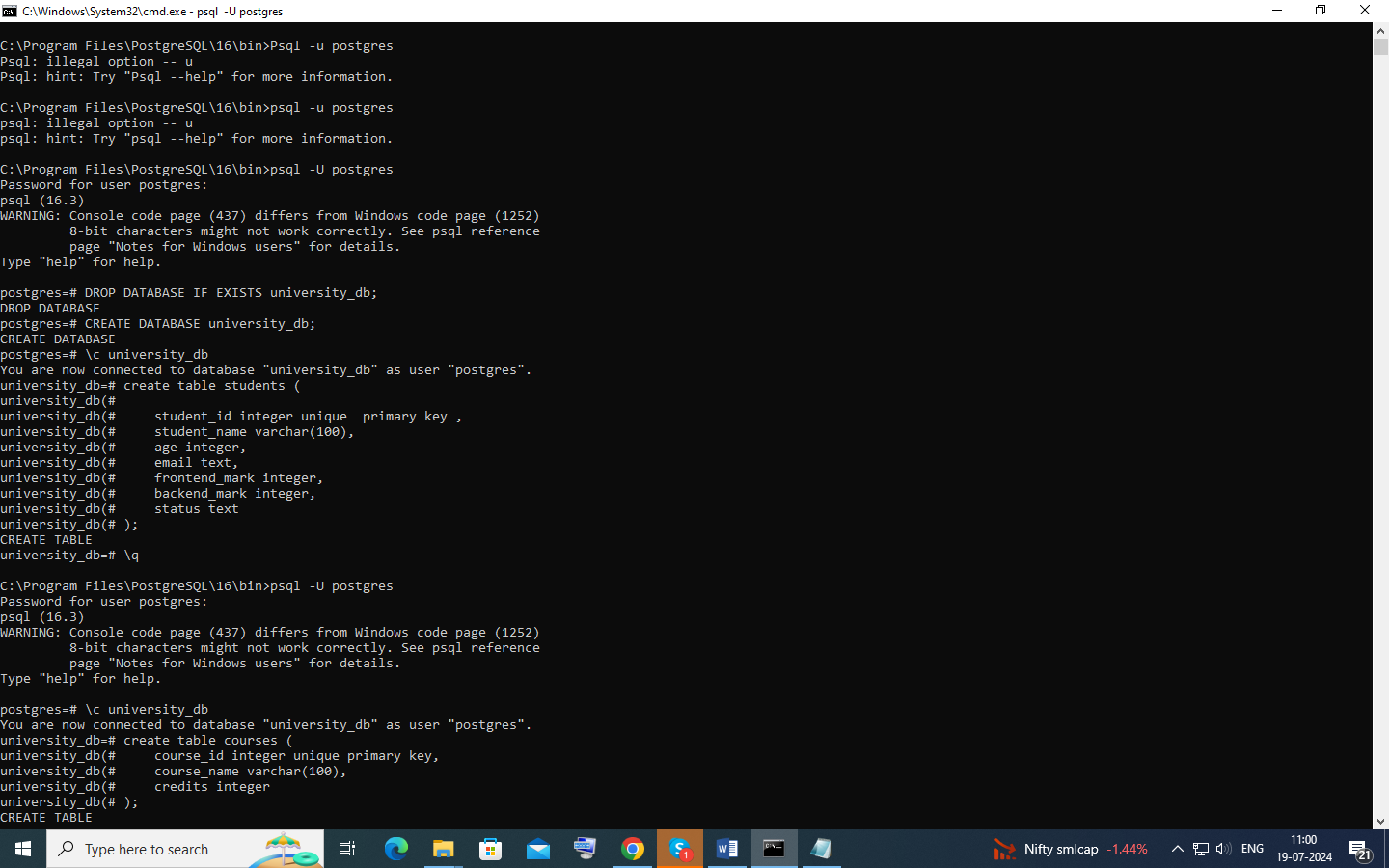
student\_id INTEGER,

course\_id INTEGER,

constraint fk\_student foreign key (student\_id) REFERENCES students(student\_id),

constraint fk\_course foreign key (course\_id) REFERENCES courses(course\_id)

);



Insert into students values(1,'Alice', 22 ,'alice@example.com' ,55, 57, NULL),

(2,'Bob', 22 ,'bob@example.com' ,34, 45, NULL),

(3,'Charlie', 22 ,'charlie@example.com' ,60, 59, NULL),

(4,'David', 22 ,'david@example.com' ,40, 49, NULL),

(5,'Eve', 22 ,'newemail@example.com' ,45, 34, NULL),

(6,'Rahim', 22 ,'rahim@gmail.com' ,46, 42, NULL);

insert into courses values (1,'Next.js',3),

(2,'React.js',4),

(3,'Databases',3),

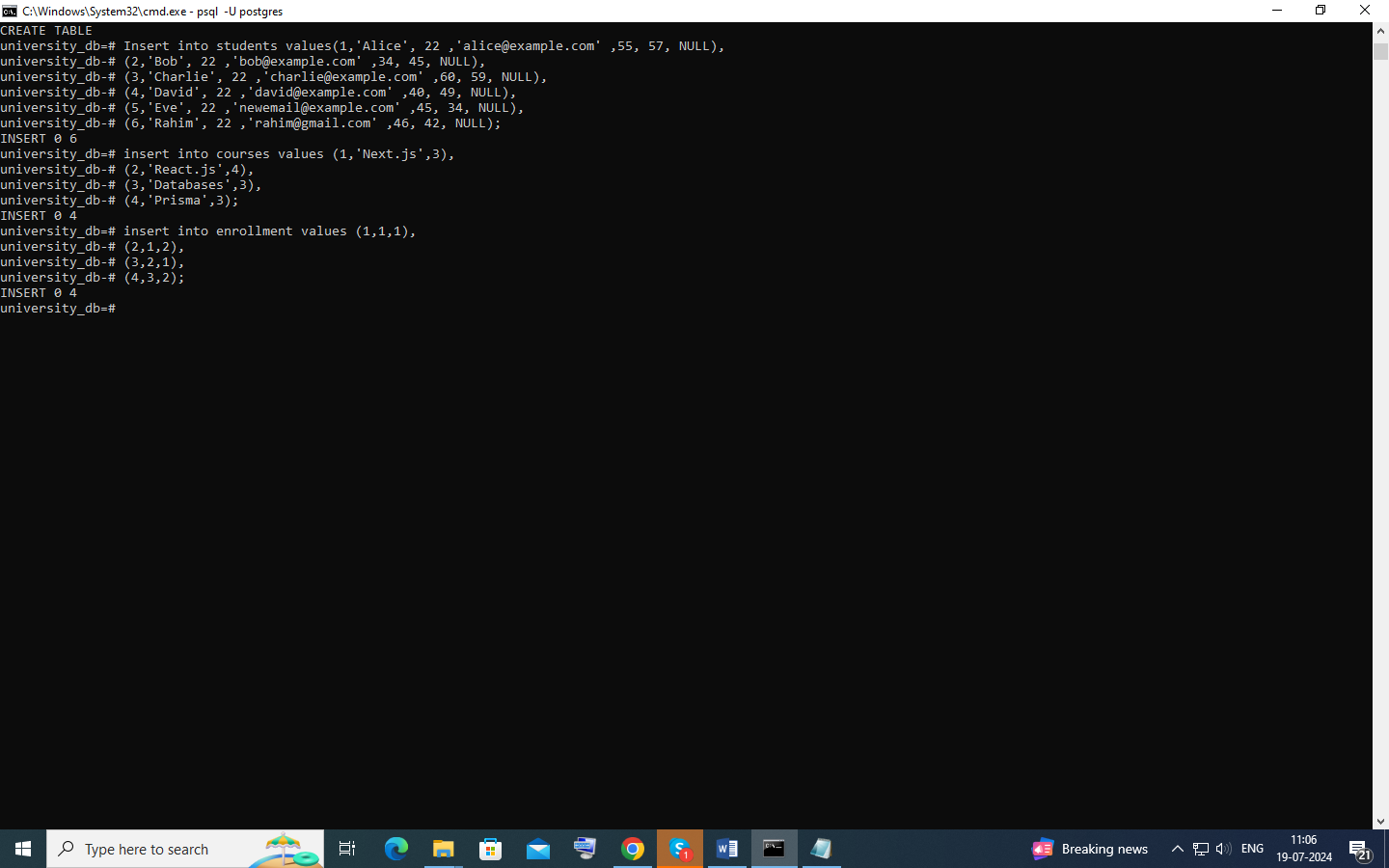
(4,'Prisma',3);

insert into enrollment values (1,1,1),

(2,1,2),

(3,2,1),

(4,3,2);



SQL QUERIES :

Query 1:

insert into students values(7,'sanjai j',22,'sanjai2k1j@gmail.com',80,100,NULL);

// Inserting value into table students

Query 2:

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'));

// using inner queries to get couse id and match with enrollment table to get student id and query the students name

Query 3:

update students

set status = 'Awarded'

where student\_id = (

Select student\_id

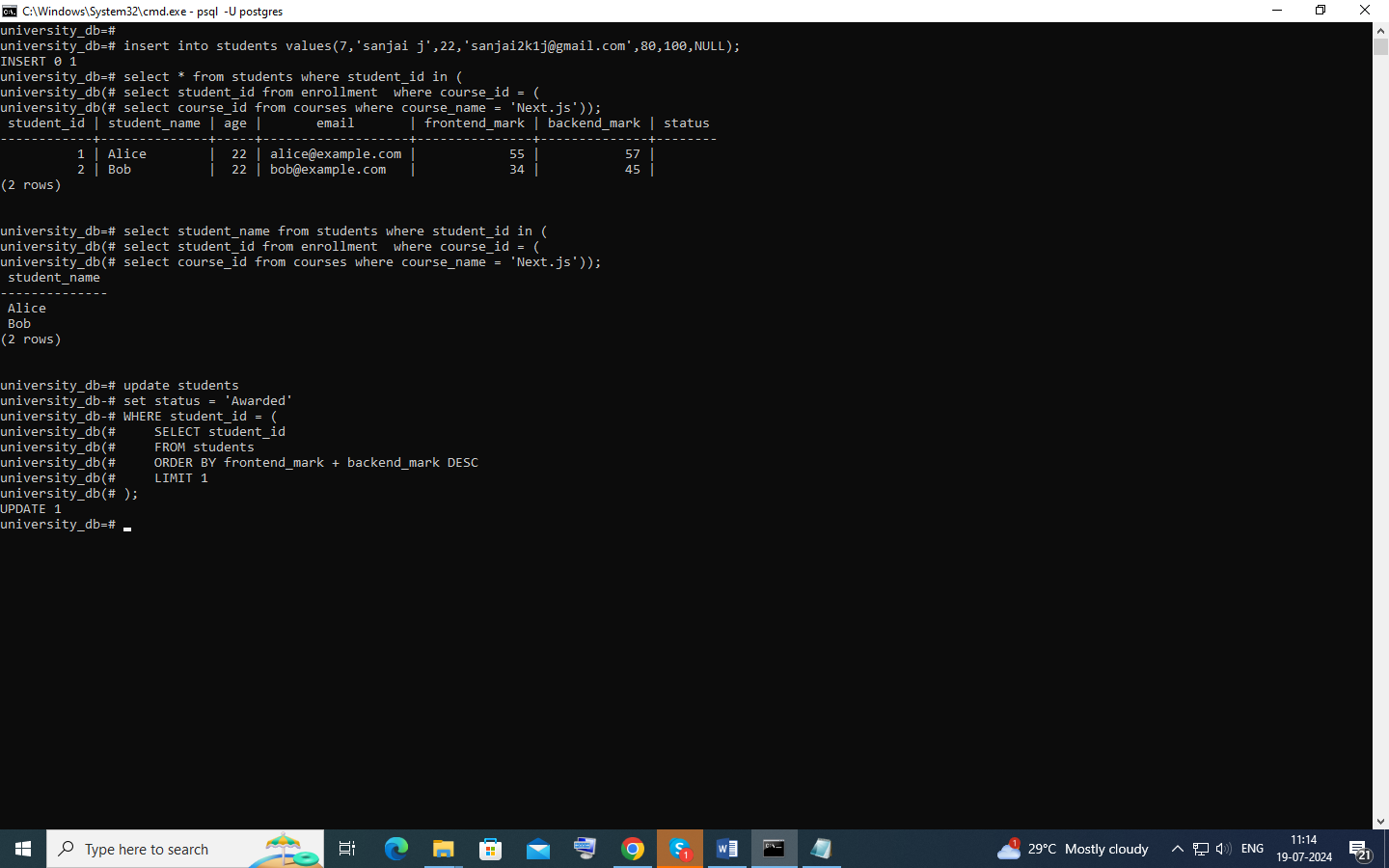
From students

Order by frontend\_mark + backend\_mark desc

Limit 1

);

// using inner query to get the student with high mark and updating



Query 4 :

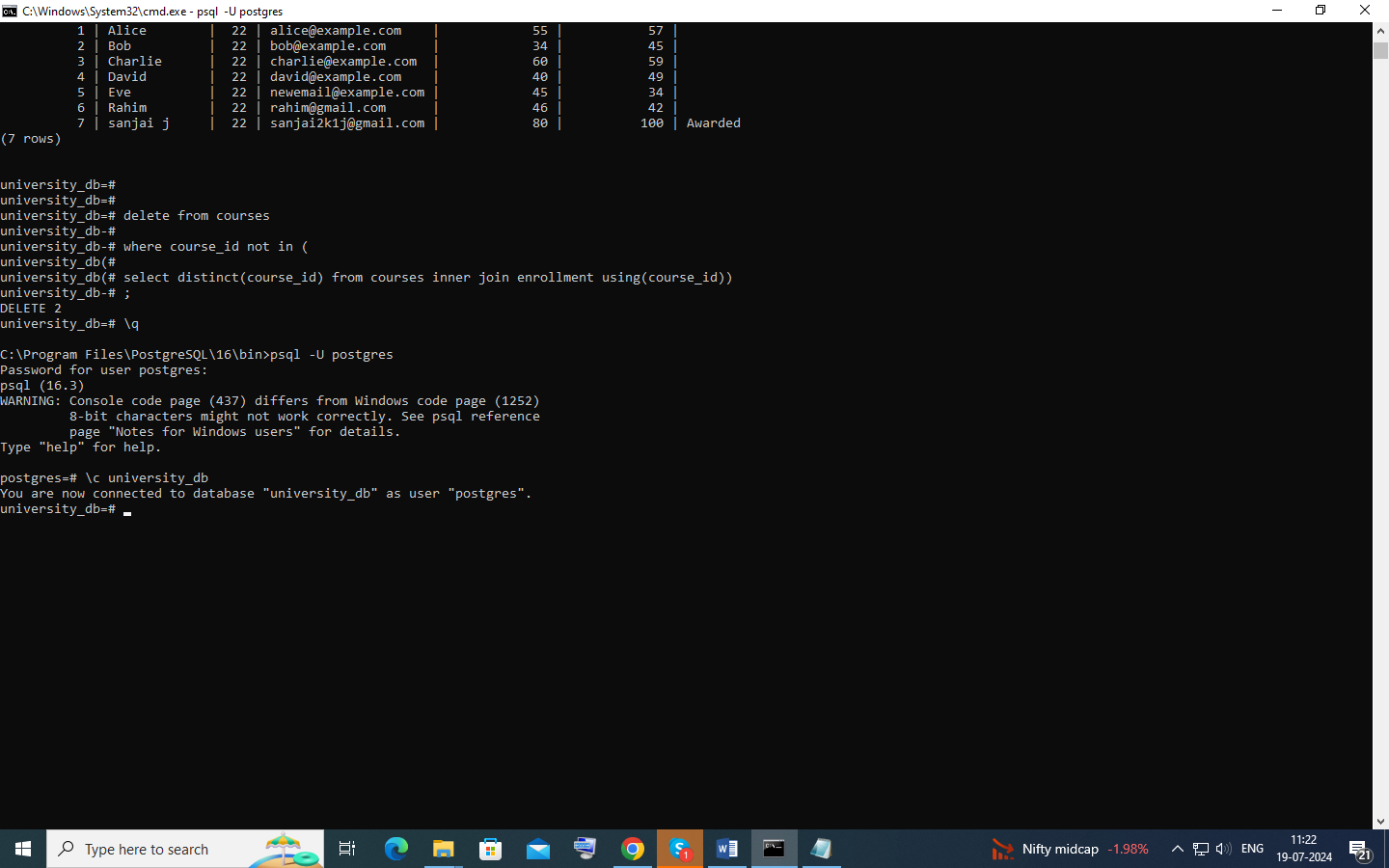
delete from courses

where course\_id not in (

select distinct(course\_id) from courses inner join enrollment using(course\_id))

;

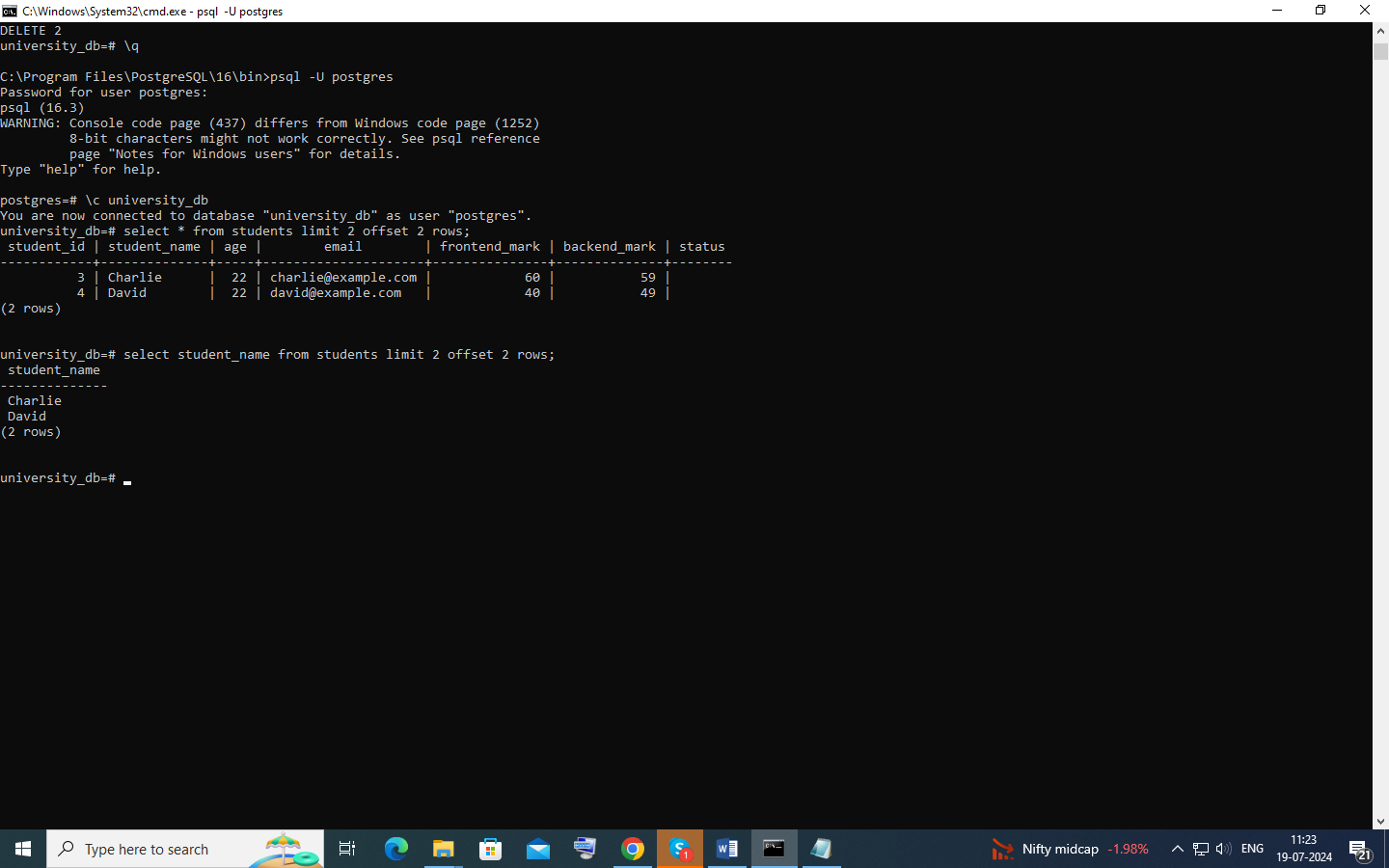
// using inner query to get courses id and deleting courses which doesn’t match



Query 5 :

select student\_name from students limit 2 offset 2 rows;

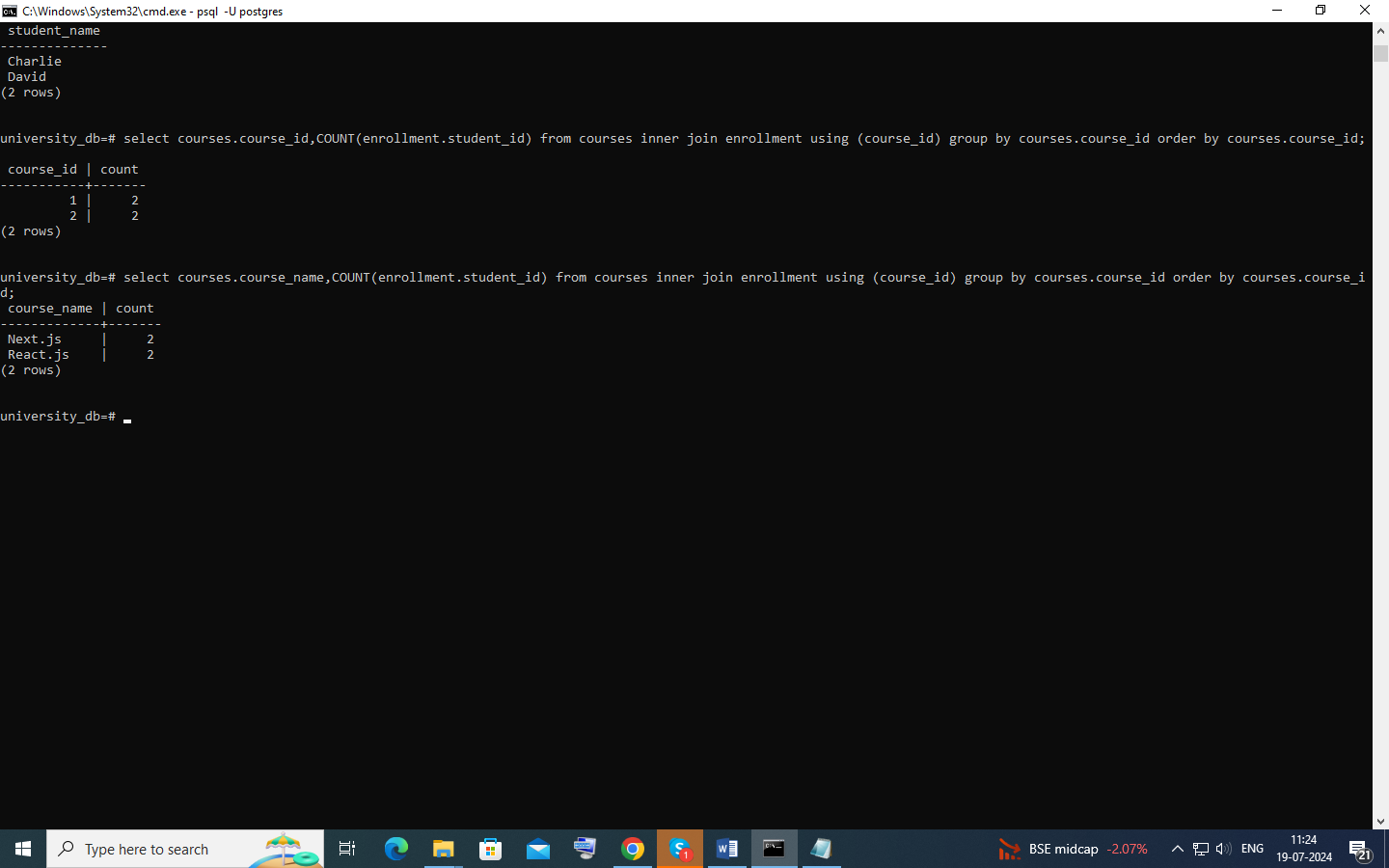
// limiting rows and using offset to skip rows



Ouery 6:

select courses.course\_name,count(enrollment.student\_id) from courses inner join enrollment using (course\_id) group by courses.course\_id order by courses.course\_id;

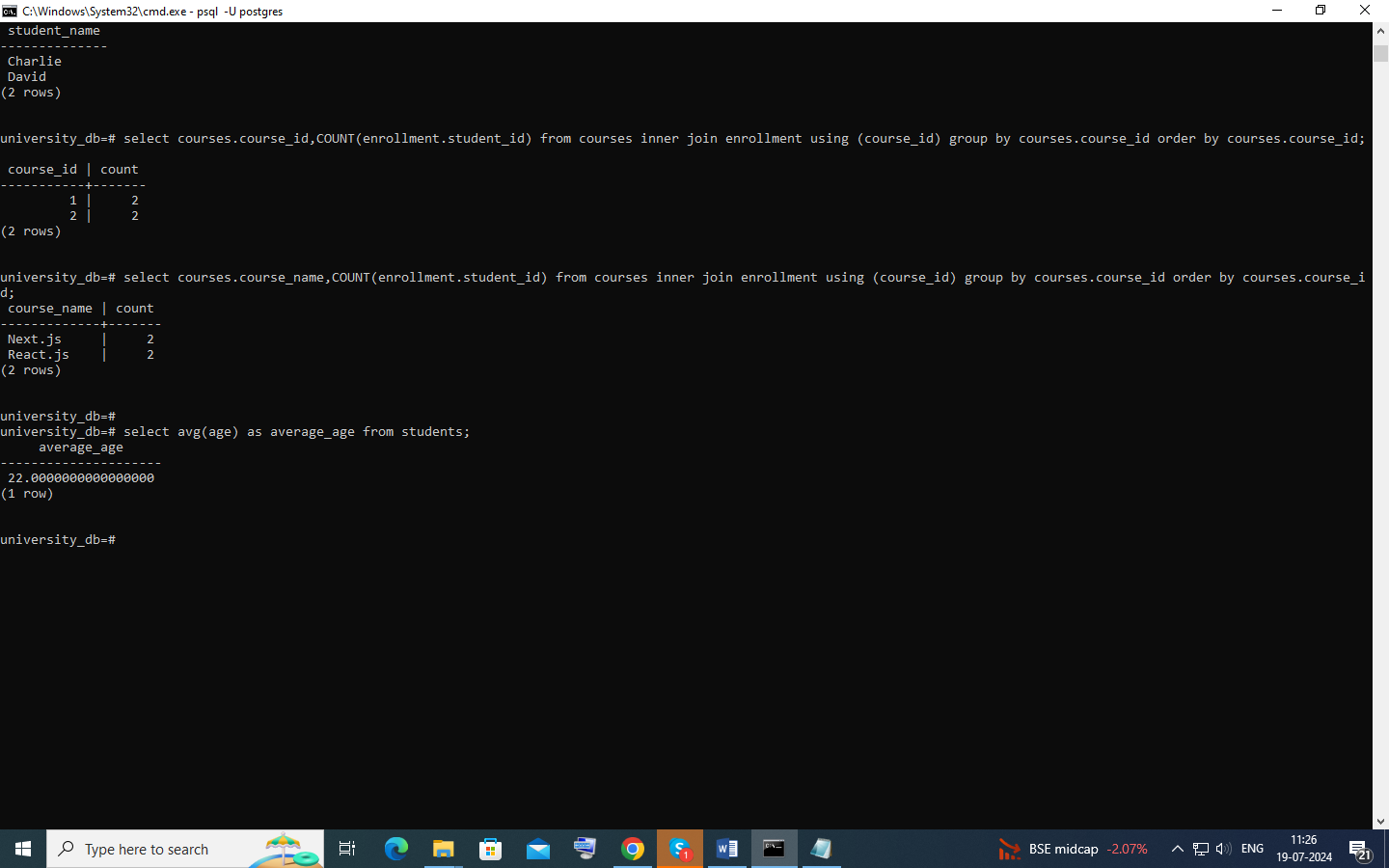
// using inner join to get coursed\_id and grouping and counting the enrollment\_id and ordering it



Query 7:

select avg(age) as average\_age from students;

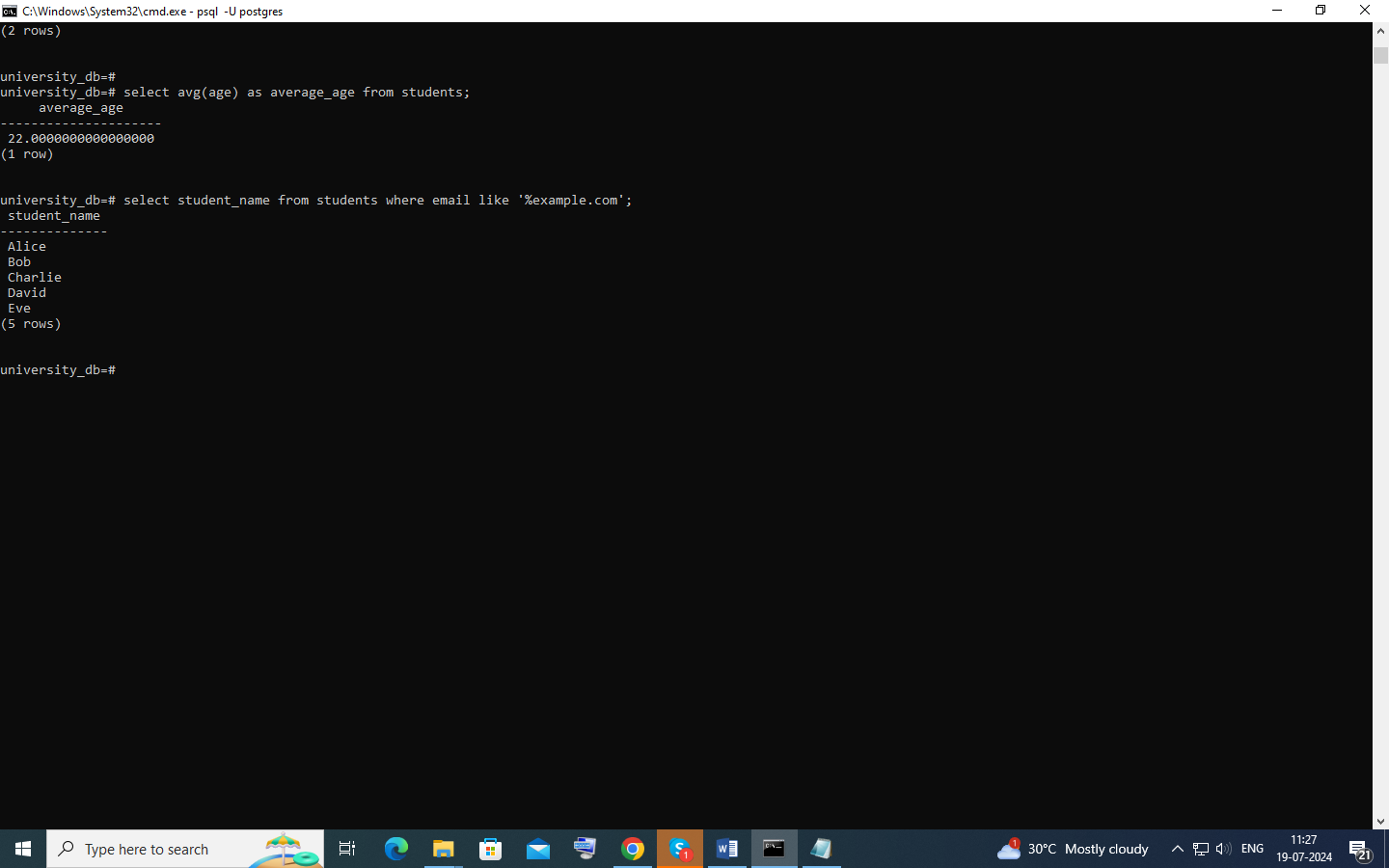
// using aggregrate function avg



Query 8 :

select student\_name from students where email like '%example.com';

//using wild cards and like operator



Concepts :

Explain the primary key and foreign key concepts in PostgreSQL.

Primary Key:

The primary key is like a unique ID for each row in a table. It's a special column (or a combination of columns) that ensures every row is distinct.

It ensures that there are no duplicate rows in the table.

Every table can have only one primary key.

Foregin –key:

A foreign key is a column or a group of columns in a table that points to the primary key of another table.

It establishes a link between two tables.

What is the difference between the VARCHAR and CHAR data types?

VARCHAR: Variable-length character string. Stores characters of varying lengths up to a maximum specified length.

CHAR: Fixed-length character string. Pads shorter strings with spaces to match the defined length. Example:email is VARCHAR because names vary in length.

Explain the purpose of the WHERE clause in a SELECT statement.

The WHERE clause filters rows based on a specified condition. It allows retrieval of rows that satisfy the condition. Example: WHERE email LIKE '%example.com' filters students whose email addresses contain 'example.com'.

What are the LIMIT and OFFSET clauses used for?

LIMIT: Specifies the maximum number of rows to return.

OFFSET: Specifies how many rows to skip before starting to return rows. Used for pagination. Example: LIMIT 2 OFFSET 2 retrieves 2 rows starting from the 3rd row.

How can you perform data modification using UPDATE statements?

UPDATE statement modifies existing records in a table based on specified conditions. Example: UPDATE students SET status = 'Awarded' WHERE ... updates the status of a student based on certain criteria

What is the significance of the JOIN operation, and how does it work in PostgreSQL?

JOIN: Combines rows from two or more tables based on related columns. Allows retrieval of related data from multiple tables in a single query. Example: LEFT JOIN enrollment ON c.course\_id = e.course\_id retrieves courses with the count of students enrolled.

Explain the GROUP BY clause and its role in aggregation operations.

GROUP BY: Groups rows that share a common value into summary rows. Used with aggregate functions like COUNT, SUM, AVG, etc., to perform operations on groups of data. Example: GROUP BY c.course\_name groups courses for counting enrolled students.

How can you calculate aggregate functions like COUNT, SUM, and AVG in PostgreSQL?

COUNT: Counts the number of rows returned by a query. o SUM: Calculates the sum of values in a column. o AVG: Calculates the average of values in a column. Example: SELECT AVG(age) AS average\_age FROM students calculates the average age of all students.

What is the purpose of an index in PostgreSQL, and how does it optimize query performance?

Index: Improves query performance by reducing the number of data pages PostgreSQL needs to read. Speeds up data retrieval operations. Example: CREATE INDEX idx\_student\_email ON students(email); creates an index on the email column for faster email-based searches.

Explain the concept of a PostgreSQL view and how it differs from a table.

View: Virtual table based on the result set of a SELECT query. Stores a query definition but not data itself. Simplifies complex queries, provides data security by limiting access to columns. Example: CREATE VIEW view\_students AS SELECT student\_name, age FROM students WHERE age > 21; creates a view of students older than 21