PostgreSQL Assignment

Database Setup:

* Create a fresh database titled **"university\_db"** or any other appropriate name.

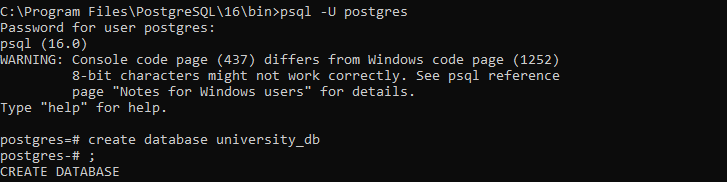
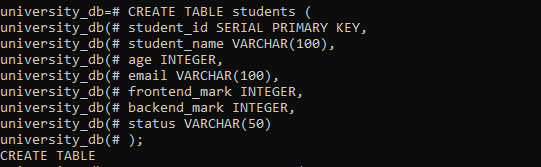


Table Creation:

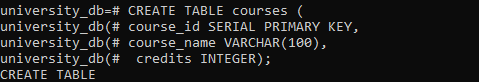
Create a **"students"** table with the following fields:

* student\_id (Primary Key): Integer, unique identifier for students.
* student\_name: String, representing the student's name.
* age: Integer, indicating the student's age.
* email: String, storing the student's email address.
* frontend\_mark: Integer, indicating the student's frontend assignment marks.
* backend\_mark: Integer, indicating the student's backend assignment marks.
* status: String, storing the student's result status.



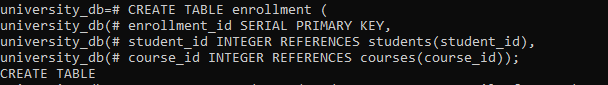
Create a **"courses"** table with the following fields:

* course\_id (Primary Key): Integer, unique identifier for courses.
* course\_name: String, indicating the course's name.
* credits: Integer, signifying the number of credits for the course.



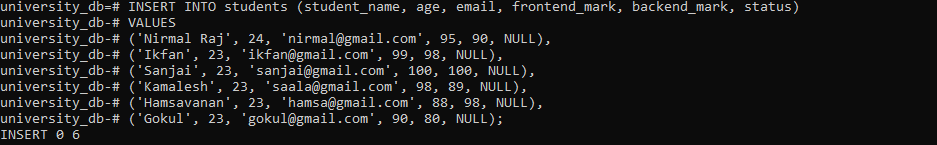
Create an **"enrollment"** table with the following fields:

* enrollment\_id (Primary Key): Integer, unique identifier for enrollments.
* student\_id (Foreign Key): Integer, referencing student\_id in "Students" table.
* course\_id (Foreign Key): Integer, referencing course\_id in "Courses" table.

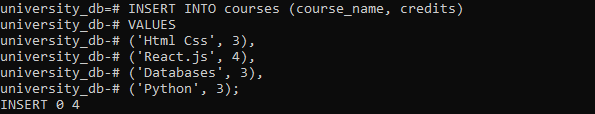


Sample Data

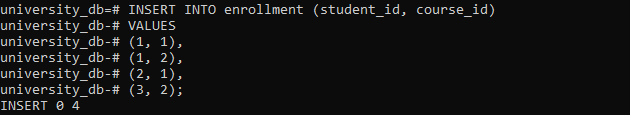
* Insert the following sample data into the **"students"** table:



* Insert the following sample data into the **"courses"** table:



* Insert the following sample data into the **"enrollment"** table:



Execute SQL queries to fulfill the ensuing tasks:

Query 1:

Insert a new student record with the following details:

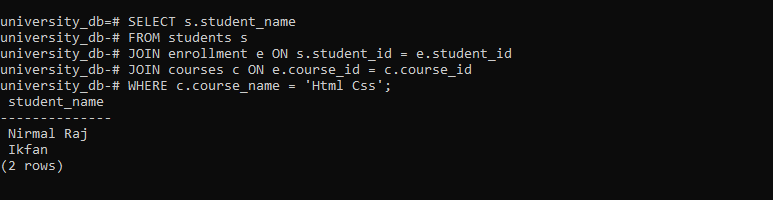
* Name: YourName
* Age: YourAge
* Email: YourEmail
* Frontend-Mark: YourMark
* Backend-Mark: YourMark
* Status: NULL



Query 2:

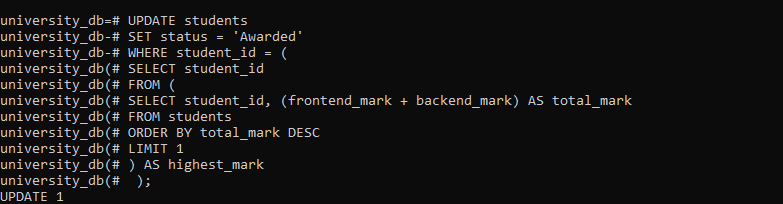
Retrieve the names of all students who are enrolled in the course titled Html Css.js'.

**Sample Output:**



Query 3:

Update the status of the student with the highest total (frontend\_mark + backend\_mark) mark to 'Awarded'



Query 4:

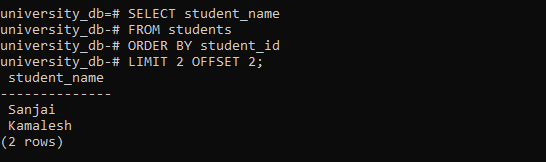
Delete all courses that have no students enrolled.



Query 5:

Retrieve the names of students using a limit of 2, starting from the 3rd student.

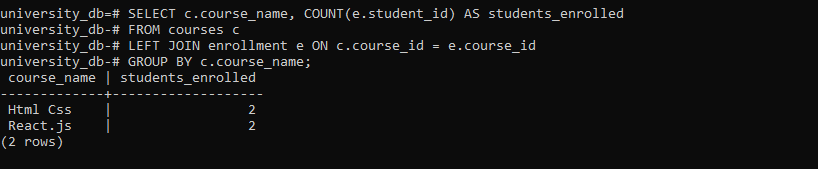
**Sample Output:**



Query 6:

Retrieve the course names and the number of students enrolled in each course.

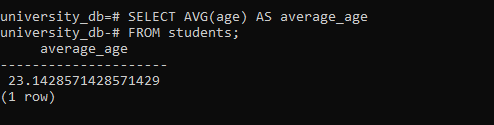
**Sample Output:**



Query 7:

Calculate and display the average age of all students.

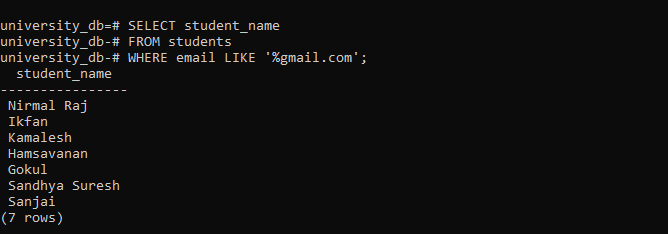
**Sample Output:**



Query 8:

Retrieve the names of students whose email addresses contain ‘gmail.com'.

**Sample Output:**



**Based on the above table data explain the concept along with  the example for below items**

1. Explain the primary key and foreign key concepts in PostgreSQL.
2. What is the difference between the VARCHAR and CHAR data types?
3. Explain the purpose of the WHERE clause in a SELECT statement.
4. What are the LIMIT and OFFSET clauses used for?
5. How can you perform data modification using UPDATE statements?
6. What is the significance of the JOIN operation, and how does it work in PostgreSQL?
7. Explain the GROUP BY clause and its role in aggregation operations.
8. How can you calculate aggregate functions like COUNT, SUM, and AVG in PostgreSQL?
9. What is the purpose of an index in PostgreSQL, and how does it optimize query performance?
10. Explain the concept of a PostgreSQL view and how it differs from a table.

1. **Primary Key and Foreign Key in PostgreSQL**:

* **Primary Key**: A primary key is a column or a set of columns that uniquely identifies each row in a table. It enforces entity integrity and ensures that each record is uniquely identifiable.
* **Foreign Key**: A foreign key is a column or a set of columns in a table that refers to the primary key of another table. It establishes a link between two tables, enforcing referential integrity and ensuring that data remains consistent between related tables.

2 **Difference between VARCHAR and CHAR data types**:

* **VARCHAR**: Variable-length character string. It stores strings of varying lengths, up to a specified maximum.
* **CHAR**: Fixed-length character string. It stores strings of a fixed length, padding shorter strings with spaces.

3 **Purpose of the WHERE clause in a SELECT statement**:

* The **WHERE** clause filters records based on a condition, allowing you to retrieve only the rows that meet specific criteria from a table.

4 **LIMIT and OFFSET clauses**:

* **LIMIT**: Specifies the maximum number of rows to return in a result set.
* **OFFSET**: Specifies the number of rows to skip before starting to return rows.

5 **Performing data modification using UPDATE statements**:

* The **UPDATE** statement in PostgreSQL modifies existing records in a table based on a specified condition, allowing you to change the values of one or more columns.

6 **Significance of the JOIN operation in PostgreSQL**:

* **JOIN** combines rows from two or more tables based on a related column between them, allowing you to retrieve data that spans across multiple tables.

7 **GROUP BY clause and its role in aggregation operations**:

* The **GROUP BY** clause groups rows that have the same values into summary rows, and it is used with aggregate functions like SUM, COUNT, AVG, etc., to perform calculations on grouped data.

8 **Calculating aggregate functions in PostgreSQL**:

* Aggregate functions like **COUNT**, **SUM**, and **AVG** calculate values across a set of rows. For example:
  + **COUNT(column\_name)**: Counts the number of rows that match a specified condition.
  + **SUM(column\_name)**: Calculates the sum of numeric values in a column.
  + **AVG(column\_name)**: Calculates the average value of numeric values in a column.

9 **Purpose of an index in PostgreSQL**:

* An **index** in PostgreSQL is a database object that improves the speed of data retrieval operations on a table at the cost of additional storage space. It enhances query performance by allowing the database to locate rows quickly using indexed columns.

10 **PostgreSQL view and its difference from a table**:

* A **view** in PostgreSQL is a virtual table based on the result set of a SELECT query. It does not store data physically but provides a way to present data from one or more tables or views in a structured format. Unlike a table, a view does not hold actual data but acts as a stored query that can be queried like a table.