# PostgreSQL Assignment

### Assignment Description

In this assignment, you will work with PostgreSQL, a powerful open-source relational database management system. Your task involves creating 03 tables based on the provided sample data and then writing and executing queries to perform various database operations such as creating, reading, updating, and deleting data. Additionally, you will explore concepts like LIMIT and OFFSET, JOIN operations, GROUP BY, aggregation and LIKE.

## Instructions:

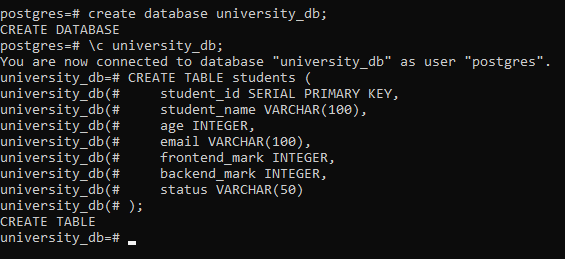
### Database Setup:

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

## Table Creation:

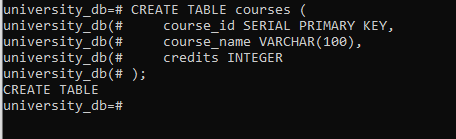
Create a **"students"** table with the following fields:

* 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.



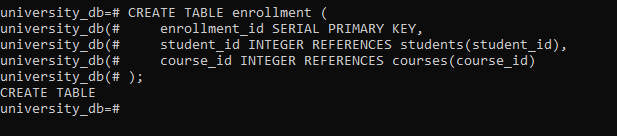
Create a **"courses"** table with the following fields:

* 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:

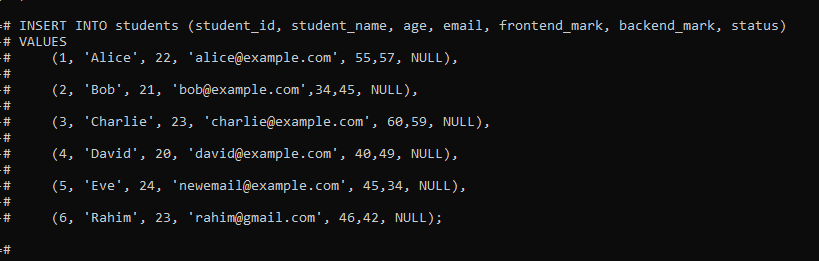
* enrollment\_id (Primary Key): Integer, unique identifier for enrollments.
* student\_id (Foreign Key): Integer, referencing student\_id in "Students" table.
* course\_id (Foreign Key): Integer, referencing course\_id in "Courses" table.



## Sample Data

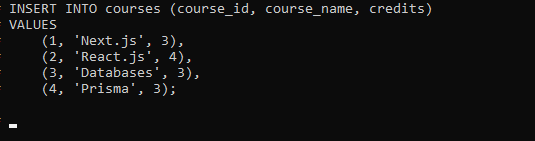
* Insert the following sample data into the **"students"** table:

| **student\_id** | **student\_name** | **age** | **email** | **frontend\_mark** | **backend\_mark** | **status** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Alice | 22 | [alice@example.com](mailto:alice@example.com) | 55 | 57 | NULL |
| 2 | Bob | 21 | [bob@example.com](mailto:bob@example.com) | 34 | 45 | NULL |
| 3 | Charlie | 23 | [charlie@example.com](mailto:charlie@example.com) | 60 | 59 | NULL |
| 4 | David | 20 | [david@example.com](mailto:david@example.com) | 40 | 49 | NULL |
| 5 | Eve | 24 | [newemail@example.com](mailto:newemail@example.com) | 45 | 34 | NULL |
| 6 | Rahim | 23 | [rahim@gmail.com](mailto:rahim@gmail.com) | 46 | 42 | NULL |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |



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

| **course\_id** | **course\_name** | **credits** |
| --- | --- | --- |
| 1 | Next.js | 3 |
| 2 | React.js | 4 |
| 3 | Databases | 3 |
| 4 | Prisma | 3 |



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

| **enrollment\_id** | **student\_id** | **course\_id** |
| --- | --- | --- |
| 1 | 1 | 1 |
| 2 | 1 | 2 |
| 3 | 2 | 1 |
| 4 | 3 | 2 |

## 

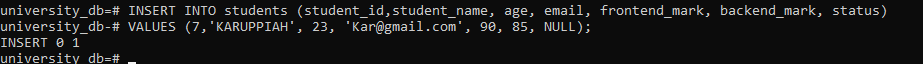
## Execute SQL queries to fulfill the ens

## uing 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:

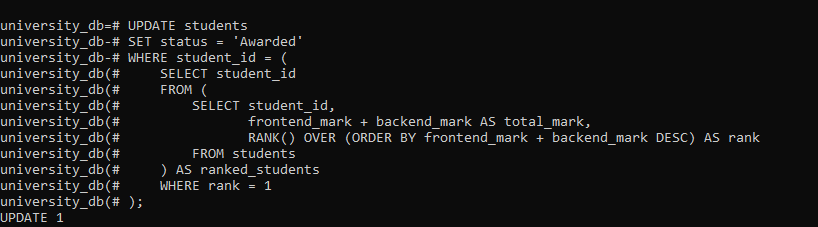
Retrieve the names of all students who are enrolled in the course titled 'Next.js'.

**Sample Output:**

| **student\_name** |
| --- |
| Alice |
| Bob |

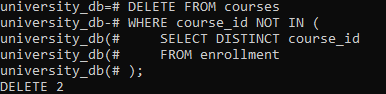
### Query 3:

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:**

| **student\_name** |
| --- |
| Charlie |
| David |
|  |

### Query 6:

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

**Sample Output:**

| **course\_name** | **students\_enrolled** |
| --- | --- |
| Next.js | 2 |
| React.js | 2 |

### 

### Query 7:

Calculate and display the average age of all students.

**Sample Output:**

| **average\_age** |
| --- |
| 22.2857142857142857 |
|  |

### Query 8:

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

**Sample Output:**

| **student\_name** |
| --- |
| Alice |
| Bob |
| Charlie |
| `David |
|  |
|  |

Prepare the SQL code for table creation, sample data insertion, and the seven queries in a text document or your preferred format. Include comments explaining each query's purpose and functionality. **Save your document as "PostgreSQL\_Assignment.sql" or any other appropriate name.**

**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.

**Primary Key:**

A primary key is a uniquely identifies each row in a table.

It cannot have NULL values and must be unique for each row.

**Foreign Key:**

A foreign key is a column that creates a relationship between two tables.

It references the primary key in another table to ensure data consistency.

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

**VARCHAR:** Variable-length string. Uses only as much space as needed.

**CHAR:** Fixed-length string. Always uses the specified amount of space, padding with spaces if necessary.

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

WHERE keyword is used for conditional statements on a table or in between tables for filter records.

1. What are the LIMIT and OFFSET clauses used for?

LIMIT is used to restricts the number of rows on a query.

OFFSET is used to skips number of rows before start.

1. How can you perform data modification using UPDATE statements?

The UPDATE statement changes existing records in a table.

UPDATE students

SET student\_name = sureshkumar

WHERE student\_id = 1;

1. 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 a similar column between those tables.

SELECT s.student\_name, c.course\_name

FROM students

JOIN enrollment ON s.student\_id = e.student\_id

JOIN courses ON e.course\_id = c.course\_id;

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

GROUP BY that have the same values in specified columns into summary rows.

Aggregate functions are COUNT, SUM, AVG, MIN,MAX.

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

SELECT COUNT(\*) FROM students;

SELECT SUM(backend\_mark) FROM students;

SELECT AVG(age) FROM students;

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

An index that helps to improves the speed of data retrieval from a database table.

It works like an index in a book, allowing quick look-ups when using query for the table to retrive the data from the table.

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

A view is a virtual table on the result of a SELECT query.

A view does not store data itself but displays data stored in other tables.