

EXP 8 : PostgreSQL

Aim : Set up a PostgreSQL database and create tables to store relational data. Perform basic CRUD operations using SQL queries.

Theory :

1. Setting Up PostgreSQL

- **PostgreSQL** is an open-source, object-relational database system.
- To get started, you install PostgreSQL on your computer or use a cloud-based service.
- After installation, you use tools like **pgAdmin** (GUI) or **psql** (CLI) to interact with your database.

2. Creating a Database

- A **database** is a container that stores organized data.
- You create a database to hold all your related tables, views, and relationships for a particular system (e.g., university management).
- Databases are isolated from each other and help manage different sets of data.

3. Creating Relational Tables

- A **table** stores data in rows and columns.
- **Relational tables** are connected using **keys**:
 - **Primary Key**: A unique identifier for each row in a table.
 - **Foreign Key**: A column that links to the primary key of another table, creating a relationship between tables.
- Example: A **students** table can reference a **teachers** table using a foreign key to represent the advisor-student relationship.

4. Performing CRUD Operations

CRUD stands for the four basic functions used to manage data in a database:

❖❖ Create (C)

- Inserting new data into a table.
- Used when adding new entries like students, teachers, etc.

❖❖ Read (R)

- Retrieving data from tables.
- Used to display records, filter results, and generate reports.

❖❖ Update (U)

- Modifying existing records in a table.
- Used to correct or change information like a student's course.

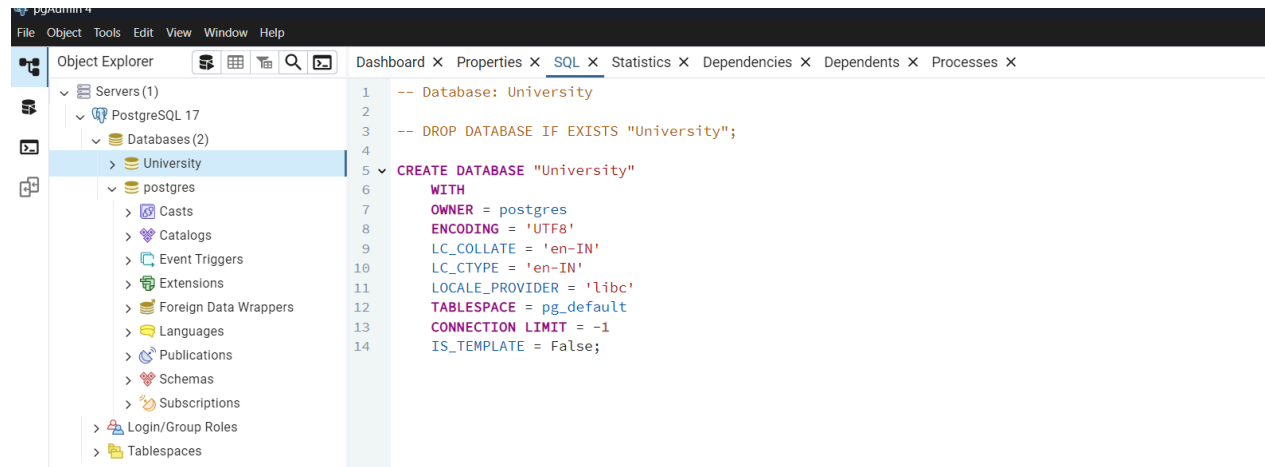
❖❖ Delete (D)

- Removing records from a table.
- Used to eliminate outdated or incorrect data.

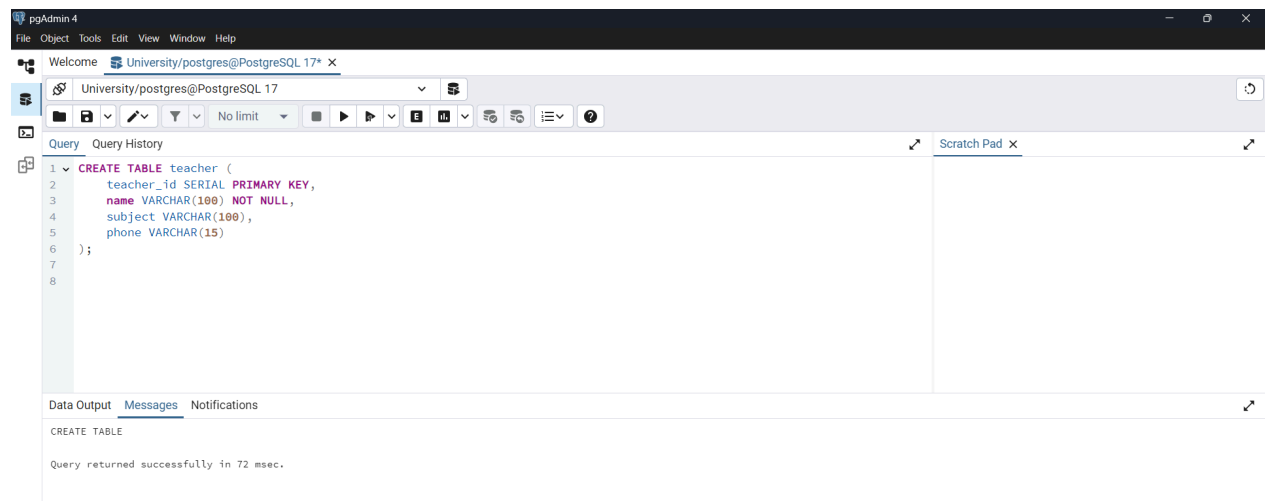
5. Benefits of Relational Design

- Ensures **data integrity** through relationships.
- Avoids **data duplication** using foreign keys and normalization.
- Supports **complex queries** across related tables using joins.

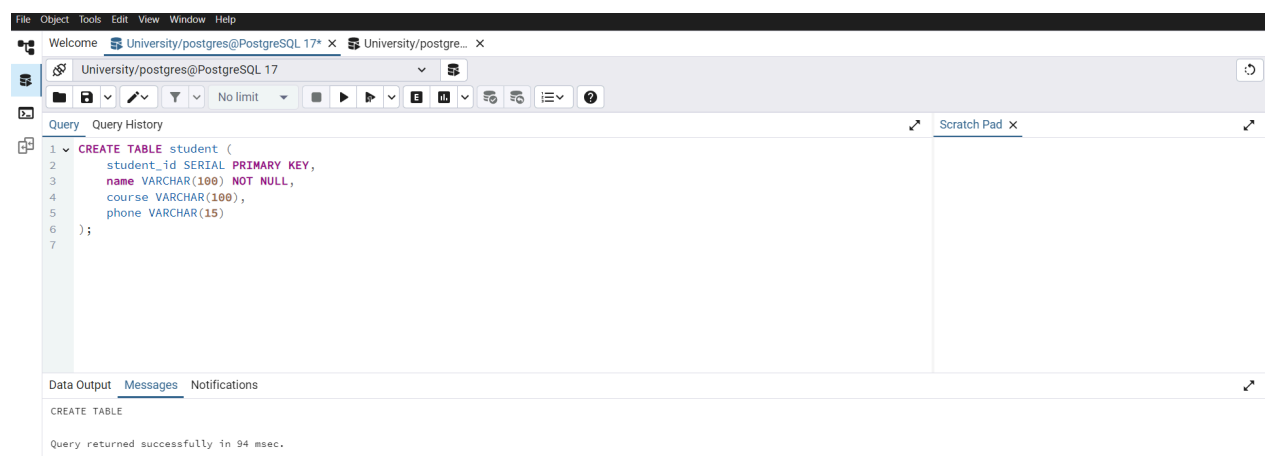
1. Creating an University Database in postgresql using pgadmin4



2. Creating teacher table



3. Creating student table



4. Inserting data in teachers table

University/postgres@PostgreSQL 17*

University/postgres@PostgreSQL 17

No limit

Query Query History

Scratch Pad

```
1 INSERT INTO teacher (name, subject, phone) VALUES
2 ('Anita Sharma', 'Mathematics', '9876543210'),
3 ('Rajiv Mehta', 'Physics', '9123456789'),
4 ('Kavita Rao', 'Chemistry', '9988776655'),
5 ('Amit Verma', 'Biology', '9871234567'),
6 ('Sneha Kulkarni', 'English', '9900112233');
7
```

Data Output Messages Notifications

INSERT 0 5

Query returned successfully in 70 msec.

5. Inserting data in student table

University/postgres@PostgreSQL 17*

University/postgres@PostgreSQL 17

No limit

Query Query History

Scratch Pad

```
1 INSERT INTO student (name, course) VALUES
2 ('Rahul Desai', 'Computer Science'),
3 ('Priya Nair', 'Mechanical Engineering'),
4 ('Arjun Patel', 'Electrical Engineering'),
5 ('Sneha Joshi', 'Civil Engineering'),
6 ('Kiran Mehta', 'Information Technology');
7
```

Data Output Messages Notifications

INSERT 0 5

Query returned successfully in 80 msec.

6. Reading data from student table

File Object Tools Edit View Window Help

University/postgres@PostgreSQL 17*

University/postgres@PostgreSQL 17

No limit

Query Query History

Scratch Pad

```
1 select * from student;
2
```

Data Output Messages Notifications

	student_id [PK] integer	name character varying (100)	course character varying (100)	phone character varying (15)
1	1	Rahul Desai	Computer Science	[null]
2	2	Priya Nair	Mechanical Engineering	[null]
3	3	Arjun Patel	Electrical Engineering	[null]
4	4	Sneha Joshi	Civil Engineering	[null]
5	5	Kiran Mehta	Information Technology	[null]

Showing rows: 1 to 5 Page No: 1

7.Updating student data

The screenshot shows a PostgreSQL client window titled "University/postgres@PostgreSQL 17* X". The query editor contains the following SQL statement:

```
1 UPDATE student
2 SET course = 'Data Science'
3 WHERE name = 'Priya Nair';
4
```

The "Data Output" tab is selected, displaying the message: "UPDATE 1" and "Query returned successfully in 78 msec."

The screenshot shows the same PostgreSQL client window. The query editor now contains the following SQL statement:

```
1 select * from student;
2
```

The "Data Output" tab is selected, displaying a table with 5 rows and 4 columns. The table structure is as follows:

	student_id [PK] integer	name character varying (100)	course character varying (100)	phone character varying (15)
1	1	Rahul Desai	Computer Science	[null]
2	3	Arjun Patel	Electrical Engineering	[null]
3	4	Sneha Joshi	Civil Engineering	[null]
4	5	Kiran Mehta	Information Technology	[null]
5	2	Priya Nair	Data Science	[null]

The interface also shows "Showing rows: 1 to 5" and "Page No: 1 of 1".

8.Deleting student data

The first screenshot shows a PostgreSQL IDE window with the query `delete from student where student_id = 3;` entered in the Query editor. The Data Output pane shows the message `DELETE 1` and `Query returned successfully in 74 msec.`

The second screenshot shows the same IDE window with the query `select * from student;` entered. The Data Output pane displays a table with 4 rows and 4 columns: `student_id` (integer), `name` (character varying (100)), `course` (character varying (100)), and `phone` (character varying (15)).

student_id	name	course	phone
1	Rahul Desai	Computer Science	[null]
2	Sneha Joshi	Civil Engineering	[null]
3	Kiran Mehta	Information Technology	[null]
4	Priya Nair	Data Science	[null]

Conclusion : In this experiment, we successfully explored the process of setting up a PostgreSQL database and managing relational data through structured tables and basic SQL operations. By understanding the relational model, we were able to design interconnected tables using primary and foreign keys, ensuring data consistency and integrity. The implementation of CRUD operations—Create, Read, Update, and Delete—demonstrated how data can be efficiently inserted, retrieved, modified, and removed within the system.