EXPERIMENT NO. 08

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

THEORY:

PostgreSQL Database Setup and CRUD Operations

PostgreSQL is a powerful, open-source object-relational database system with over 35 years of active development. It's known for reliability, feature robustness, and performance. Here's a brief introduction to the core concepts you'll need to work with PostgreSQL:

Key Concepts

1. **Relational Database**: PostgreSQL organizes data into tables with relationships between them, allowing for efficient data storage and retrieval.
2. **SQL (Structured Query Language)**: The standard language used to communicate with the database system.
3. **CRUD Operations**: The four basic functions of persistent storage:
    - Create (INSERT)
    - Read (SELECT)
    - Update (UPDATE)
    - Delete (DELETE)
4. **Tables and Schemas**: Tables store data in rows and columns. Schemas group related tables together.
5. **Primary and Foreign Keys**: Primary keys uniquely identify rows in a table. Foreign keys establish relationships between tables.
6. **Data Types**: PostgreSQL supports numerous data types including numeric, string, date/time, boolean, geometric, and custom types.

```
Server [localhost]: localhost
Database [postgres]: postgres
Port [5432]: 5432
Username [postgres]: postgres
Password for user postgres:

psql (17.4)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.
```

CREATE DATABASE

```
postgres=# \dt
Did not find any relations.
postgres=# CREATE TABLE patients (
postgres(#     patient_id SERIAL PRIMARY KEY,
postgres(#     name VARCHAR(100),
postgres(#     age INTEGER,
postgres(#     gender VARCHAR(10),
postgres(#     diagnosis VARCHAR(150),
postgres(#     doctor_name VARCHAR(100),
postgres(#     hospital_name VARCHAR(100)
postgres(# );
CREATE TABLE
postgres=# SELECT COUNT(*) FROM patients;
```

INSERT VALUES

```
postgres=# INSERT INTO patients (name, age, gender, diagnosis, doctor_name, hospital_name) VALUES
postgres-# ('Ravi Mehra', 45, 'Male', 'Hypertension', 'Dr. Shalini Gupta', 'AIIMS Delhi'),
postgres-# ('Pooja Nair', 30, 'Female', 'Diabetes Type 2', 'Dr. Kiran Rao', 'Apollo Hospital, Chennai'),
postgres-# ('Imran Sheikh', 60, 'Male', 'Cardiac Arrest', 'Dr. Abhay Kulkarni', 'Fortis Hospital, Mumbai'),
postgres-# ('Lakshmi Devi', 55, 'Female', 'Arthritis', 'Dr. Ramesh Iyer', 'CMC Vellore');
INSERT 0 4
```

VIEW RELATION

```
postgres=# SELECT * FROM patients;
 patient_id |     name     | age | gender |    diagnosis    |    doctor_name     |      hospital_name
------------+--------------+-----+--------+-----------------+--------------------+--------------------------
          1 | Ravi Mehra   |  45 | Male   | Hypertension    | Dr. Shalini Gupta  | AIIMS Delhi
          2 | Pooja Nair   |  30 | Female | Diabetes Type 2 | Dr. Kiran Rao      | Apollo Hospital, Chennai
          3 | Imran Sheikh |  60 | Male   | Cardiac Arrest  | Dr. Abhay Kulkarni | Fortis Hospital, Mumbai
          4 | Lakshmi Devi |  55 | Female | Arthritis       | Dr. Ramesh Iyer    | CMC Vellore
(4 rows)
```

UPDATE

```
postgres=# UPDATE patients
postgres-# SET diagnosis = 'Chronic Arthritis'
postgres-# WHERE name = 'Lakshmi Devi';
UPDATE 1
```

DELETE

```
postgres=# DELETE FROM patients
postgres-# WHERE name = 'Imran Sheikh';
DELETE 1
```

FINAL TABLE

```
postgres=# SELECT * FROM patients;
 patient_id |     name     | age | gender |     diagnosis     |    doctor_name    |      hospital_name
------------+--------------+-----+--------+-------------------+-------------------+--------------------------
          1 | Ravi Mehra   |  45 | Male   | Hypertension      | Dr. Shalini Gupta | AIIMS Delhi
          2 | Pooja Nair   |  30 | Female | Diabetes Type 2   | Dr. Kiran Rao     | Apollo Hospital, Chennai
          4 | Lakshmi Devi |  55 | Female | Chronic Arthritis | Dr. Ramesh Iyer   | CMC Vellore
(3 rows)
```

Conclusion

PostgreSQL stands as a robust, versatile relational database management system that offers powerful data handling capabilities through structured SQL operations. By understanding the fundamentals of database setup and CRUD operations (Create, Read, Update, Delete), users can effectively manage persistent data in well-organized tables with defined relationships. This foundation not only enables basic data manipulation but also provides a pathway to leverage PostgreSQL's advanced features such as transactions, stored procedures, triggers, and sophisticated query optimization techniques. Mastering these core concepts empowers developers to build scalable, efficient database solutions that can handle complex data requirements across various applications, from small projects to enterprise-level systems, while maintaining data integrity and performance.