Experiment 8

Aim: Set up a PostgreSQL database for Music Store Management System and create tables (Customer, Artists, Albums, Songs and Order) to store relational data. Perform basic CRUD operations using SQL queries

Theory:

PostgreSQL is an open-source, object-relational database management system (ORDBMS) that supports both SQL (structured query language) and advanced features like transactions, indexing, stored procedures, and JSON. It is widely used for web applications, data analytics, and enterprise applications due to its scalability, reliability, and extensibility.

Steps to Connect a Web Application with PostgreSQL

1. Install PostgreSQL

- Download and install PostgreSQL from the official website.
- During installation, set up a password for the postgres superuser

2. Start PostgreSQL Server

- Start the PostgreSQL server using:
  pg_ctl -D "C:\Program Files\PostgreSQL\14\data" start

3. Connect to PostgreSQL Database

- Open psql (PostgreSQL interactive terminal) and connect:
  \c database_name;

Node.js (with pg module)

```
const { Client } = require('pg');
const client = new Client({
   user: 'postgres',
   host: 'localhost',
   database: 'music_store',
   password: 'yourpassword',
   port: 5432,
});
client.connect();
console.log("Connected to PostgreSQL database");
```

Basic CRUD Operations in PostgreSQL

1. Create (Insert Data)

```
INSERT INTO Customers (name, email, phone, address)
VALUES ('Alice Johnson', 'alice@example.com', '1234567890', '456 Park Ave');
```

2. Read (Retrieve Data)

```
SELECT * FROM Customers;
SELECT title, release_year FROM Albums WHERE release_year > 2000;
```

3. Update Data

```
UPDATE Customers
SET phone = '9876543210'
WHERE customer_id = 1;
```

4. Delete Data

```
DELETE FROM Customers WHERE customer_id = 1;
```

Relational Database Systems and Data Manipulation Commands (DML)

Relational Database Systems (RDBMS)

- An RDBMS is a database that stores data in tables with structured relationships between them.
- PostgreSQL follows the ACID properties (Atomicity, Consistency, Isolation, Durability).
- Tables are connected using primary keys and foreign keys to maintain data integrity.

Data Manipulation Language (DML)

DML commands allow modification of data in tables.

INSERT – Adds data to a table.

```
INSERT INTO Artists (name, genre) VALUES ('Adele', 'Pop');
```

UPDATE – Modifies existing data.

```
UPDATE Songs SET duration = '3:45' WHERE title = 'Hello';
```

DELETE – Removes data from a table.

```
DELETE FROM Orders WHERE order_id = 5;
```

SELECT – Retrieves data from tables.

```
SELECT * FROM Albums;
```

Data Definition Language (DDL) Commands

DDL commands are used to define and manage database structures.

1. CREATE – Creates a new database object (table, schema, index).

```
CREATE TABLE Customers (
   customer_id SERIAL PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL
);
```

2. ALTER – Modifies an existing table.

```
ALTER TABLE Customers ADD COLUMN phone VARCHAR(15);
```

3. DROP – Deletes a database object permanently.

```
DROP TABLE Customers;
```

4. TRUNCATE – Deletes all data from a table but retains the structure.

```
TRUNCATE TABLE Orders;
```

MUSIC STORE MANAGEMENT SYSTEM

1) Database Creation

```
postgres=# create database music_store;
CREATE DATABASE
postgres=# \c music_store;
You are now connected to database "music_store" as user "postgres".
```

2) Customers Table

```
music_store=# create table Customers (
music_store(# customer_id serial primary key,
music_store(# name varchar(100) not null,
music_store(# email varchar(100) unique not null,
music_store(# phone varchar(10),
music_store(# address text
music_store(# );
CREATE TABLE
```

3) Artists Table

```
music_store=# create table Artists (
music_store(# artist_id serial primary key,
music_store(# name varchar(100) not null,
music_store(# genre varchar(50)
music_store(# );
CREATE TABLE
```

4) Albums Table

```
music_store=# CREATE TABLE Albums (
music_store(#     album_id SERIAL PRIMARY KEY,
music_store(#     title VARCHAR(100) NOT NULL,
music_store(#     artist_id INT REFERENCES Artists(artist_id) ON DELETE CASCADE,
music_store(#     release_year INT
music_store(# );
CREATE TABLE
```

5) Songs Table

```
music_store=# CREATE TABLE Songs (
music_store(#     song_id SERIAL PRIMARY KEY,
music_store(#     title VARCHAR(100) NOT NULL,
music_store(#     album_id INT REFERENCES Albums(album_id) ON DELETE CASCADE,
music_store(#     duration INTERVAL NOT NULL
music_store(# );
CREATE TABLE
```

6) Orders Table

```
music_store=# CREATE TABLE Orders (
music_store(#     order_id SERIAL PRIMARY KEY,
music_store(#     customer_id INT REFERENCES Customers(customer_id) ON DELETE CASCADE,
music_store(#     order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
music_store(#     total_amount DECIMAL(10,2) NOT NULL
music_store(# );
CREATE TABLE
```

7) Insert data into Tables

```
music_store=# INSERT INTO Customers (name, email, phone, address)
music_store-# VALUES ('Diksha', 'diksha@gmail.com', '1234567890', 'Mumbai');
INSERT 0 1
music_store=# INSERT INTO Artists (name, genre)
music_store-# VALUES ('The Beatles', 'Rock');
INSERT 0 1
music_store=# INSERT INTO Albums (title, artist_id, release_year)
music_store-# VALUES ('Abbey Road', 1, 1969);
INSERT 0 1
music_store=# INSERT INTO Songs (title, album_id, duration)
music_store-# VALUES ('Perfect', 1, '2 minutes 20 seconds');
INSERT 0 1
music_store=# INSERT INTO Orders (customer_id, total_amount)
music_store-# VALUES (1, 100);
INSERT 0 1
```

8) Select Query

```
music_store=# SELECT * FROM Customers;
 customer_id |  name  |      email       |   phone    | address
-------------+--------+------------------+------------+---------
           1 | Diksha | diksha@gmail.com | 9876543210 | Mumbai
(1 row)
```

9) Select Query with JOINS

```
music_store=# SELECT a.title, ar.name FROM Albums a JOIN Artists ar ON a.artist_id = ar.artist_id;
   title    |    name
------------+-------------
 Abbey Road | The Beatles
(1 row)
```

10) Update Operation

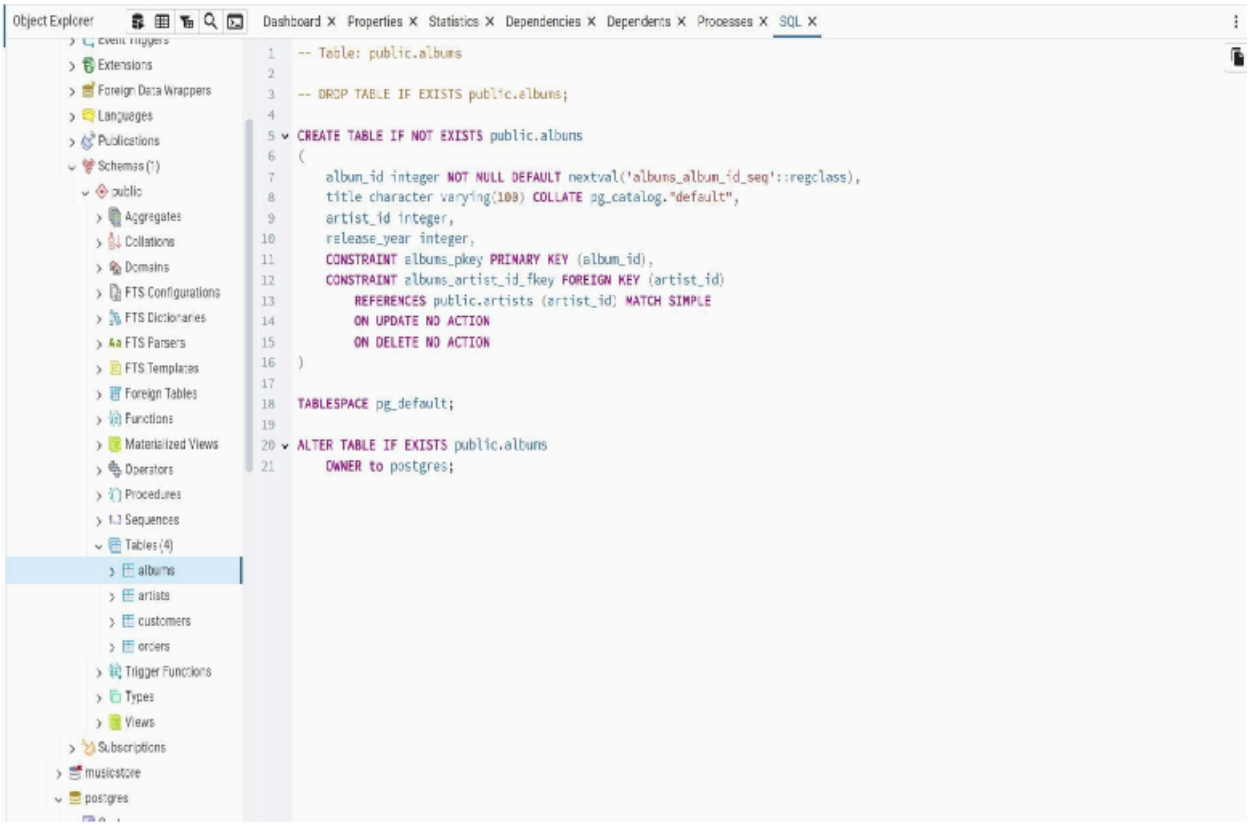
```
music_store=# UPDATE Customers
music_store-# SET phone = '9876543210'
music_store-# WHERE customer_id = 1;
UPDATE 1
```

11) Delete Operation

```
music_store=# DELETE FROM Orders WHERE order_id = 1;
DELETE 1
```

12) Drop table

```
music_store=# drop table orders
music_store-# ;
DROP TABLE
```



Conclusion:

PostgreSQL is a powerful RDBMS that supports DML (INSERT, UPDATE, DELETE, SELECT) and DDL (CREATE, ALTER, DROP, TRUNCATE) commands to manage data effectively. Connecting a web application to PostgreSQL requires installing the database, setting up a connection, and executing CRUD operations using SQL queries. This enables smooth database interactions for modern web applications.