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 a powerful, open-source object-relational database management system (RDBMS) with a strong reputation for reliability, feature robustness, and performance. It runs on all major operating systems and has been in active development for over 30 years.

Key Features of PostgreSQL:

- ACID compliance (Atomicity, Consistency, Isolation, Durability)
- Support for JSON and other NoSQL features
- Advanced indexing capabilities
- Robust security features
- Extensibility through custom functions and data types
- Excellent concurrency handling

Steps to Connect a Web Application to PostgreSQL

1. **Install PostgreSQL**: Download and install PostgreSQL on your server or development machine.
2. **Create a Database**: Use the PostgreSQL command line tool (psql) or a GUI like pgAdmin to create a new database:
   ```
   CREATE DATABASE your_database_name;
   ```
3. **Install Database Driver**: Add the appropriate PostgreSQL client library to your web application. The driver depends on your programming language:
   - Node.js: `pg` or `sequelize`
   - Python: `psycopg2` or `SQLAlchemy`
   - Java: JDBC driver or Spring Data JPA
   - PHP: `pdo_pgsql` extension
4. **Configure Connection**: Set up the database connection parameters:
   - Host (usually localhost or a remote server address)
   - Port (default is 5432)
   - Database name
   - Username
   - Password

Create Connection Code: Implement connection logic in your application. Here's a simplified example using Node.js with the `pg` library

Basic CRUD Operations

CRUD stands for Create, Read, Update, and Delete - the four basic operations for persistent storage:

1. **Create (INSERT)**:
   ```
   INSERT INTO users (name, email) VALUES ('John Doe',
   'john@example.com');
   ```

2. **Read (SELECT)**:
   ```
   SELECT * FROM users WHERE id = 1;
   ```

3. **Update (UPDATE)**:
   ```
   UPDATE users SET email = 'newemail@example.com' WHERE id =
   1;
   ```

4. **Delete (DELETE)**:
   ```
   DELETE FROM users WHERE id = 1;
   ```

Relational Database Systems and DML

Relational database systems organize data into tables (relations) with rows and columns. Each table represents an entity, and relationships between entities are established through keys.

Key Concepts in Relational Databases:

- **Tables**: Collections of related data organized in rows and columns
- **Primary Keys**: Unique identifiers for each row in a table
- **Foreign Keys**: Fields that reference primary keys in other tables
- **Constraints**: Rules enforced on data columns (NOT NULL, UNIQUE, etc.)
- **Indexes**: Data structures that improve the speed of data retrieval
- **Normalization**: Process of organizing data to reduce redundancy

Data Manipulation Language (DML)

DML consists of SQL commands used to manipulate data stored in the database:

INSERT: Adds new records to a table

```
INSERT INTO products (name, price, category_id) VALUES
('Laptop', 999.99, 2);
```

SELECT: Retrieves data from one or more tables

```
SELECT products.name, categories.name as category
FROM products
JOIN categories ON products.category_id = categories.id
WHERE products.price < 1000;
```

UPDATE: Modifies existing records

```
UPDATE products
SET price = price * 0.9
WHERE category_id = 2;
```

DELETE: Removes records from a table

```
DELETE FROM products WHERE id = 5;
```

MERGE (or UPSERT in PostgreSQL): Combines INSERT and UPDATE operations

```
INSERT INTO products (id, name, price)

VALUES (1, 'Updated Laptop', 899.99)
ON CONFLICT (id) DO UPDATE
SET name = EXCLUDED.name, price = EXCLUDED.price;
```

Data Definition Language (DDL) Commands

DDL commands are used to define and modify the structure of database objects:

1.CREATE: Creates new database objects

```
-- Create a new table
CREATE TABLE customers (
  id SERIAL PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(100) UNIQUE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
-- Create an index
CREATE INDEX idx_customer_email ON customers(email);
```

```
-- Create a view
CREATE VIEW active_customers AS
SELECT * FROM customers WHERE last_order_date > CURRENT_DATE -
INTERVAL '1 year';
```

2.ALTER: Modifies existing database objects

```
-- Add a column
ALTER TABLE customers ADD COLUMN phone VARCHAR(20);

-- Modify a column
ALTER TABLE customers ALTER COLUMN name TYPE VARCHAR(150);

-- Add a constraint

ALTER TABLE customers ADD CONSTRAINT check_email CHECK (email
LIKE '%@%.%');
```

3.DROP: Deletes database objects

```
-- Drop a table
DROP TABLE customers;

-- Drop an index
DROP INDEX idx_customer_email;

-- Drop a column

ALTER TABLE customers DROP COLUMN phone;
```

4.TRUNCATE: Removes all records from a table, but keeps the table structure

```
TRUNCATE TABLE customers;
```

5.COMMENT: Adds comments to database objects

```
COMMENT ON TABLE customers IS 'Table storing customer
information';
```

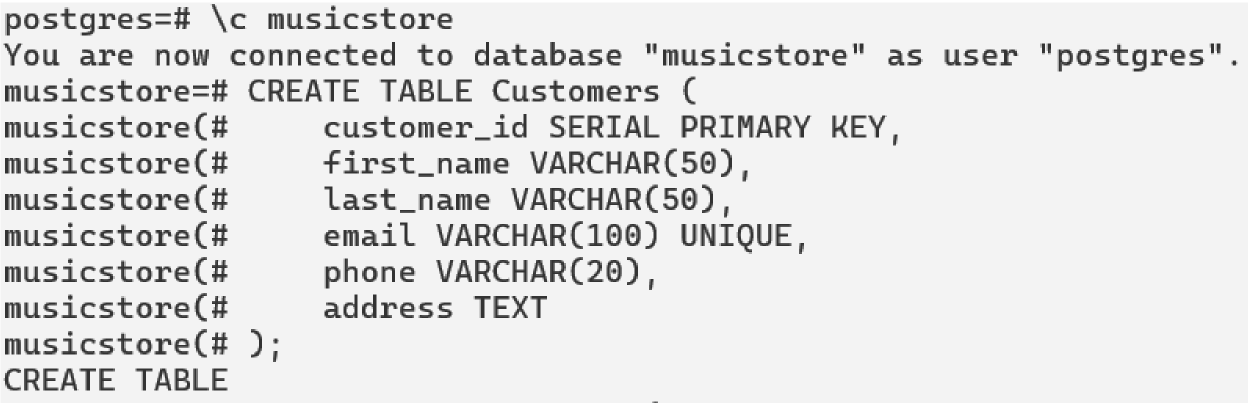
6.RENAME: Changes the name of database objects

```
ALTER TABLE customers RENAME TO clients;
```

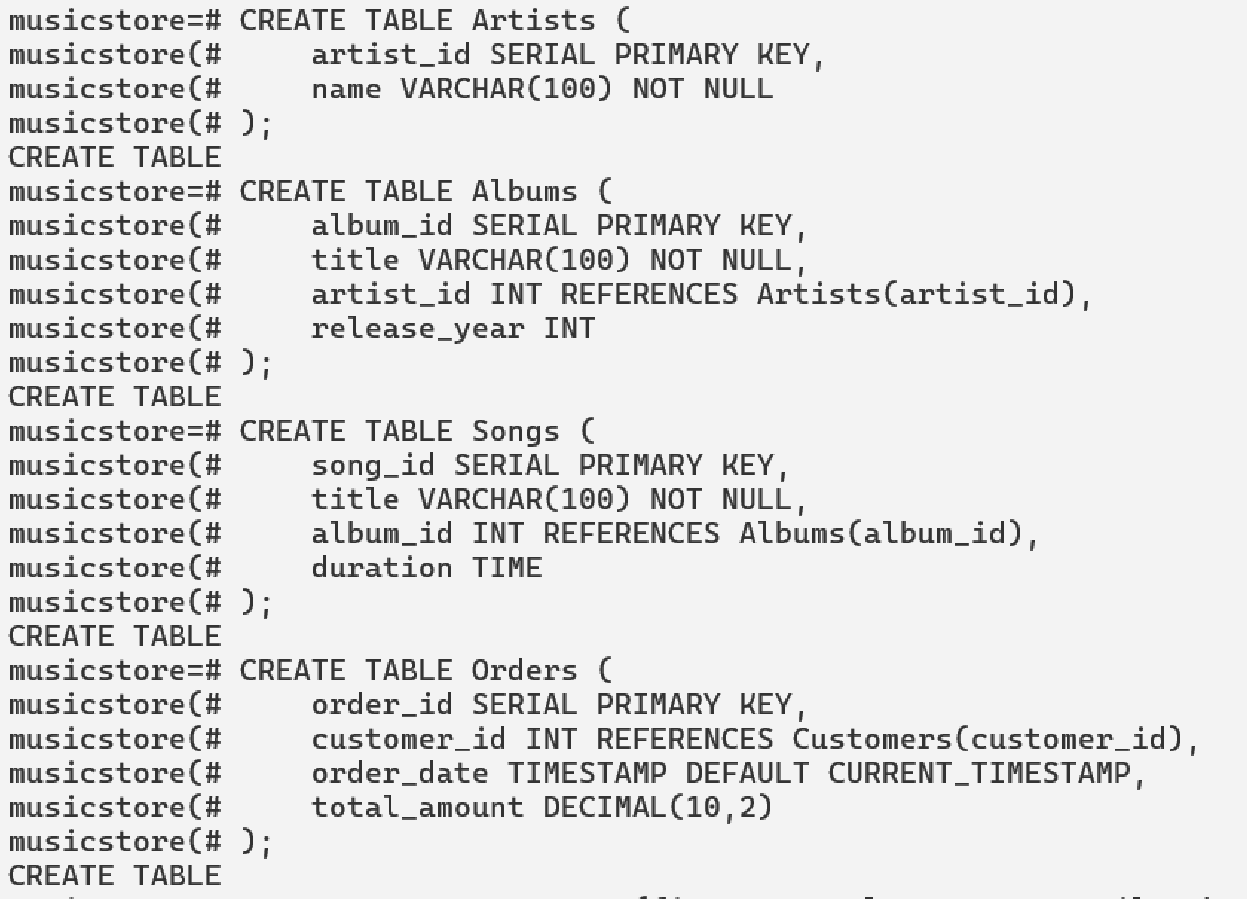
These DDL commands provide the foundation for creating and managing the structure of your database, while the DML commands allow you to work with the data within that structure.

MUSIC STORE:

TABLE CREATION:

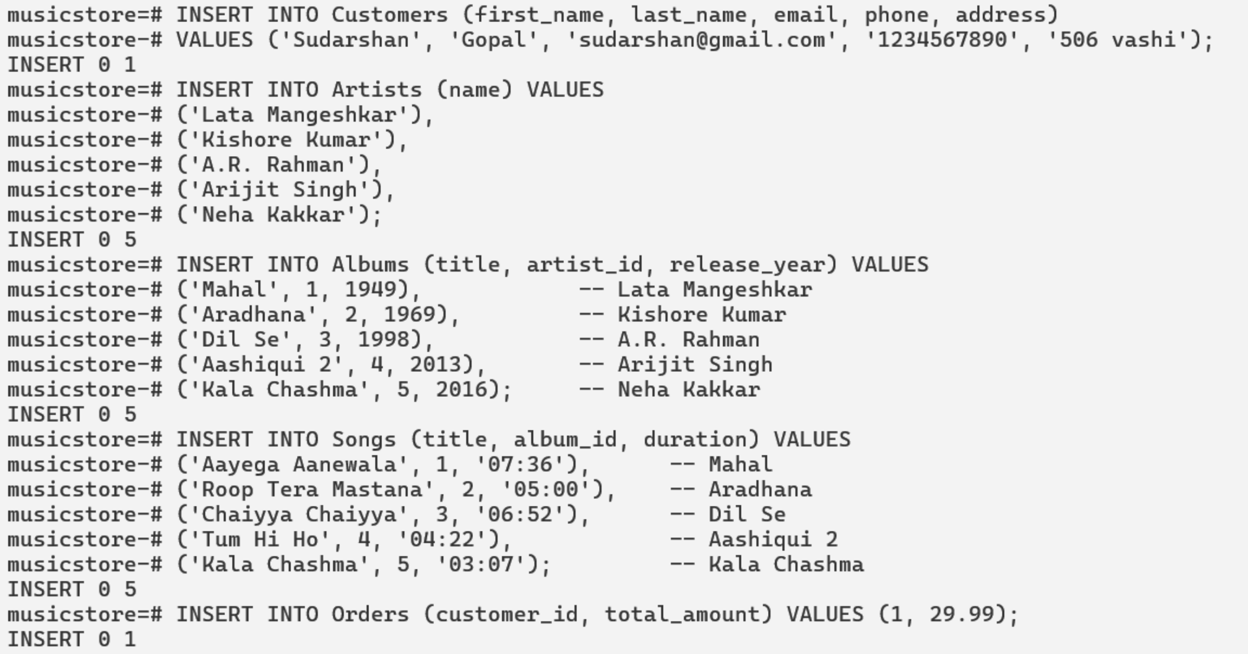


```
postgres=# \c musicstore
You are now connected to database "musicstore" as user "postgres".
musicstore=# CREATE TABLE Customers (
musicstore(#     customer_id SERIAL PRIMARY KEY,
musicstore(#     first_name VARCHAR(50),
musicstore(#     last_name VARCHAR(50),
musicstore(#     email VARCHAR(100) UNIQUE,
musicstore(#     phone VARCHAR(20),
musicstore(#     address TEXT
musicstore(# );
CREATE TABLE
```



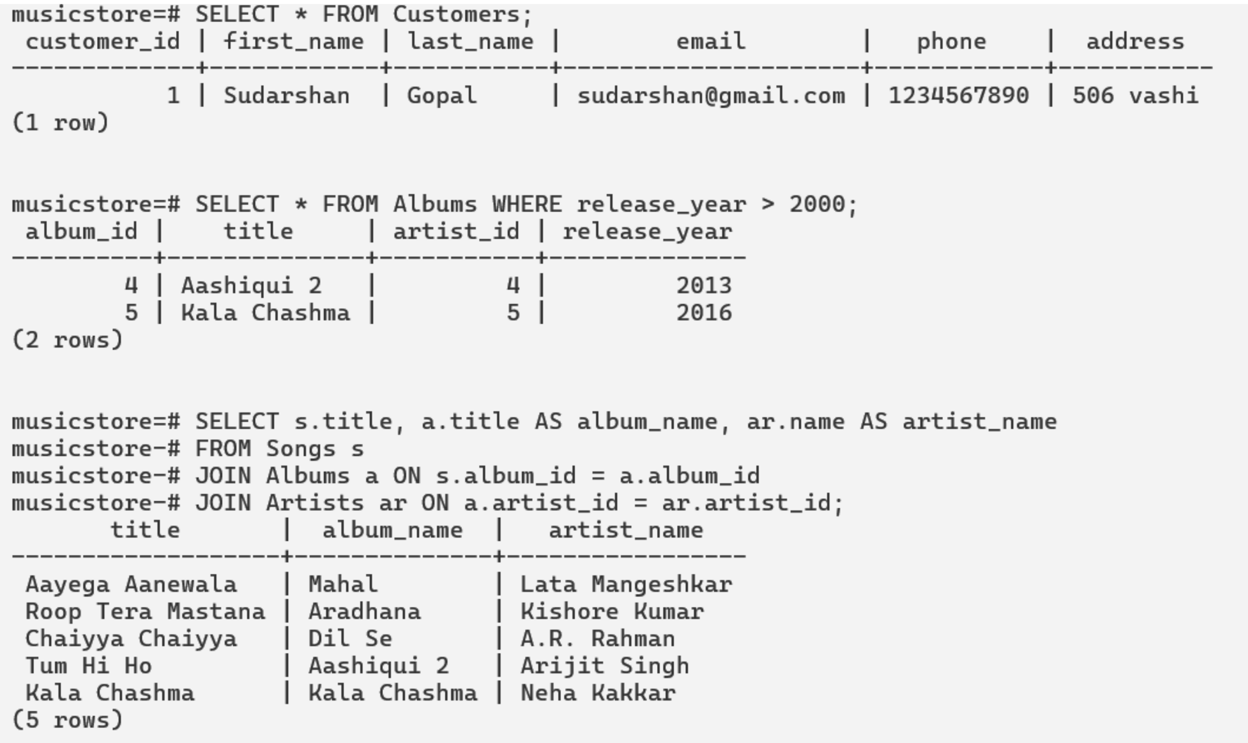
```
musicstore=# CREATE TABLE Artists (
musicstore(#     artist_id SERIAL PRIMARY KEY,
musicstore(#     name VARCHAR(100) NOT NULL
musicstore(# );
CREATE TABLE
musicstore=# CREATE TABLE Albums (
musicstore(#     album_id SERIAL PRIMARY KEY,
musicstore(#     title VARCHAR(100) NOT NULL,
musicstore(#     artist_id INT REFERENCES Artists(artist_id),
musicstore(#     release_year INT
musicstore(# );
CREATE TABLE
musicstore=# CREATE TABLE Songs (
musicstore(#     song_id SERIAL PRIMARY KEY,
musicstore(#     title VARCHAR(100) NOT NULL,
musicstore(#     album_id INT REFERENCES Albums(album_id),
musicstore(#     duration TIME
musicstore(# );
CREATE TABLE
musicstore=# CREATE TABLE Orders (
musicstore(#     order_id SERIAL PRIMARY KEY,
musicstore(#     customer_id INT REFERENCES Customers(customer_id),
musicstore(#     order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
musicstore(#     total_amount DECIMAL(10,2)
musicstore(# );
CREATE TABLE
```

INSERT INTO TABLES:

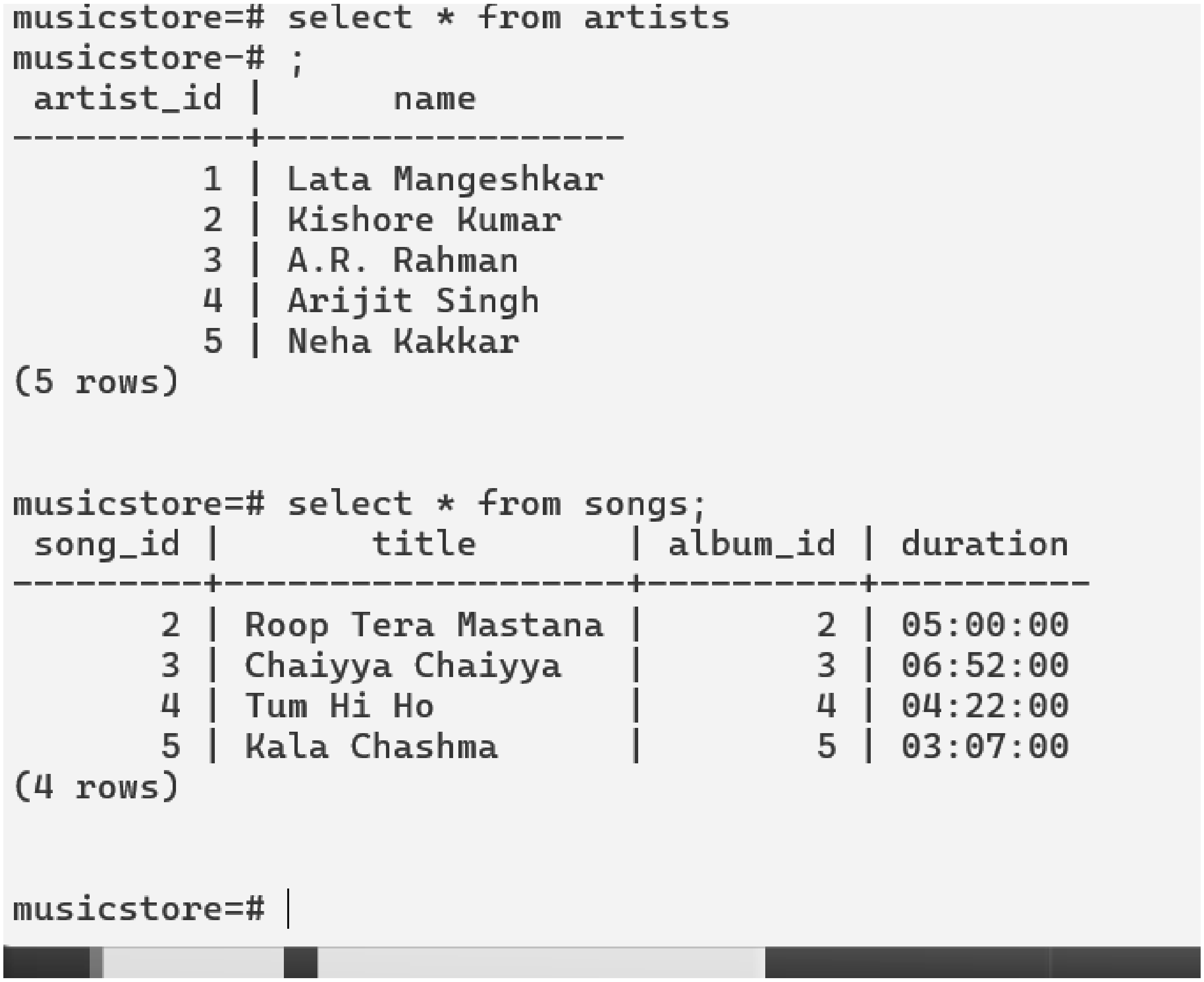


```
musicstore=# INSERT INTO Customers (first_name, last_name, email, phone, address)
musicstore-# VALUES ('Sudarshan', 'Gopal', 'sudarshan@gmail.com', '1234567890', '506 vashi');
INSERT 0 1
musicstore=# INSERT INTO Artists (name) VALUES
musicstore-# ('Lata Mangeshkar'),
musicstore-# ('Kishore Kumar'),
musicstore-# ('A.R. Rahman'),
musicstore-# ('Arijit Singh'),
musicstore-# ('Neha Kakkar');
INSERT 0 5
musicstore=# INSERT INTO Albums (title, artist_id, release_year) VALUES
musicstore-# ('Mahal', 1, 1949),            -- Lata Mangeshkar
musicstore-# ('Aradhana', 2, 1969),         -- Kishore Kumar
musicstore-# ('Dil Se', 3, 1998),           -- A.R. Rahman
musicstore-# ('Aashiqui 2', 4, 2013),       -- Arijit Singh
musicstore-# ('Kala Chashma', 5, 2016);     -- Neha Kakkar
INSERT 0 5
musicstore=# INSERT INTO Songs (title, album_id, duration) VALUES
musicstore-# ('Aayega Aanewala', 1, '07:36'),      -- Mahal
musicstore-# ('Roop Tera Mastana', 2, '05:00'),    -- Aradhana
musicstore-# ('Chaiyya Chaiyya', 3, '06:52'),      -- Dil Se
musicstore-# ('Tum Hi Ho', 4, '04:22'),            -- Aashiqui 2
musicstore-# ('Kala Chashma', 5, '03:07');         -- Kala Chashma
INSERT 0 5
musicstore=# INSERT INTO Orders (customer_id, total_amount) VALUES (1, 29.99);
INSERT 0 1
```

SELECT

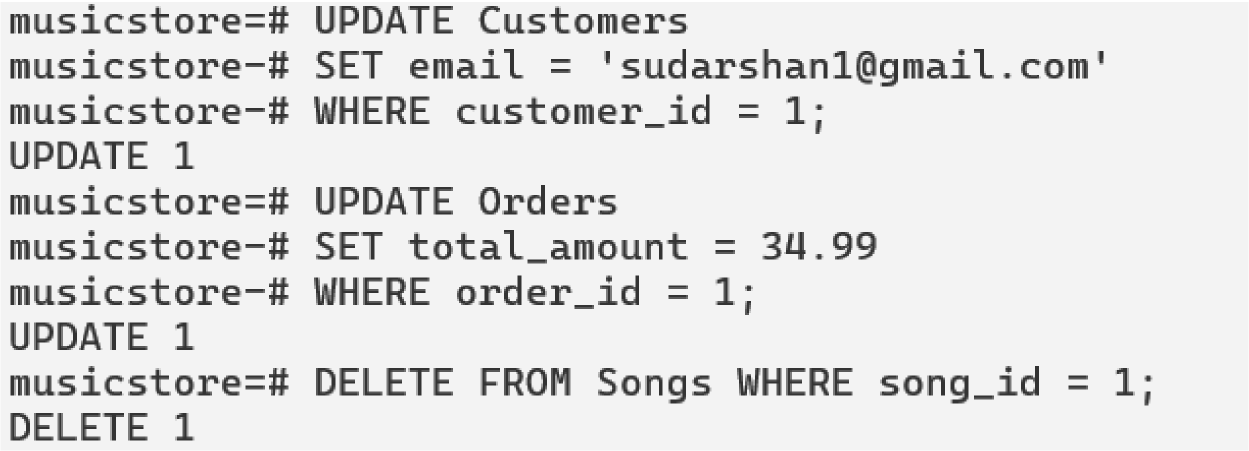


```
musicstore=# SELECT * FROM Customers;
 customer_id | first_name | last_name |        email        |   phone    |  address
-------------+------------+-----------+---------------------+------------+-----------
           1 | Sudarshan  | Gopal     | sudarshan@gmail.com | 1234567890 | 506 vashi
(1 row)


musicstore=# SELECT * FROM Albums WHERE release_year > 2000;
 album_id |    title     | artist_id | release_year
----------+--------------+-----------+--------------
        4 | Aashiqui 2   |         4 |         2013
        5 | Kala Chashma |         5 |         2016
(2 rows)


musicstore=# SELECT s.title, a.title AS album_name, ar.name AS artist_name
musicstore-# FROM Songs s
musicstore-# JOIN Albums a ON s.album_id = a.album_id
musicstore-# JOIN Artists ar ON a.artist_id = ar.artist_id;
       title       |  album_name  |   artist_name
-------------------+--------------+-----------------
 Aayega Aanewala   | Mahal        | Lata Mangeshkar
 Roop Tera Mastana | Aradhana     | Kishore Kumar
 Chaiyya Chaiyya   | Dil Se       | A.R. Rahman
 Tum Hi Ho         | Aashiqui 2   | Arijit Singh
 Kala Chashma      | Kala Chashma | Neha Kakkar
(5 rows)
```



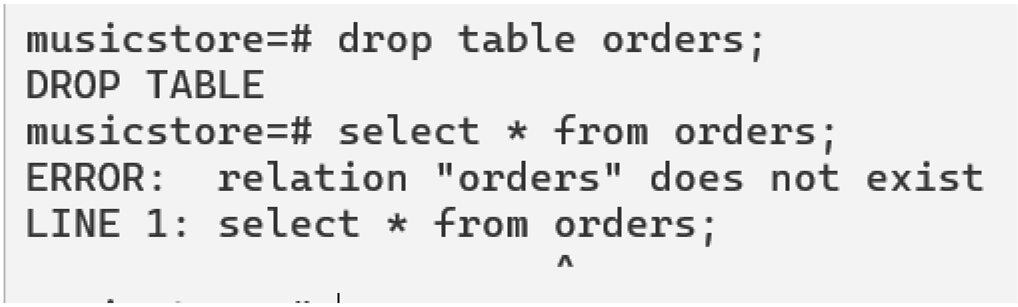
```
musicstore=# select * from artists
musicstore-# ;
 artist_id |      name
-----------+-----------------
         1 | Lata Mangeshkar
         2 | Kishore Kumar
         3 | A.R. Rahman
         4 | Arijit Singh
         5 | Neha Kakkar
(5 rows)


musicstore=# select * from songs;
 song_id |       title        | album_id | duration
---------+--------------------+----------+----------
       2 | Roop Tera Mastana  |        2 | 05:00:00
       3 | Chaiyya Chaiyya    |        3 | 06:52:00
       4 | Tum Hi Ho          |        4 | 04:22:00
       5 | Kala Chashma       |        5 | 03:07:00
(4 rows)


musicstore=# |
```

UPDATE AND DELETE

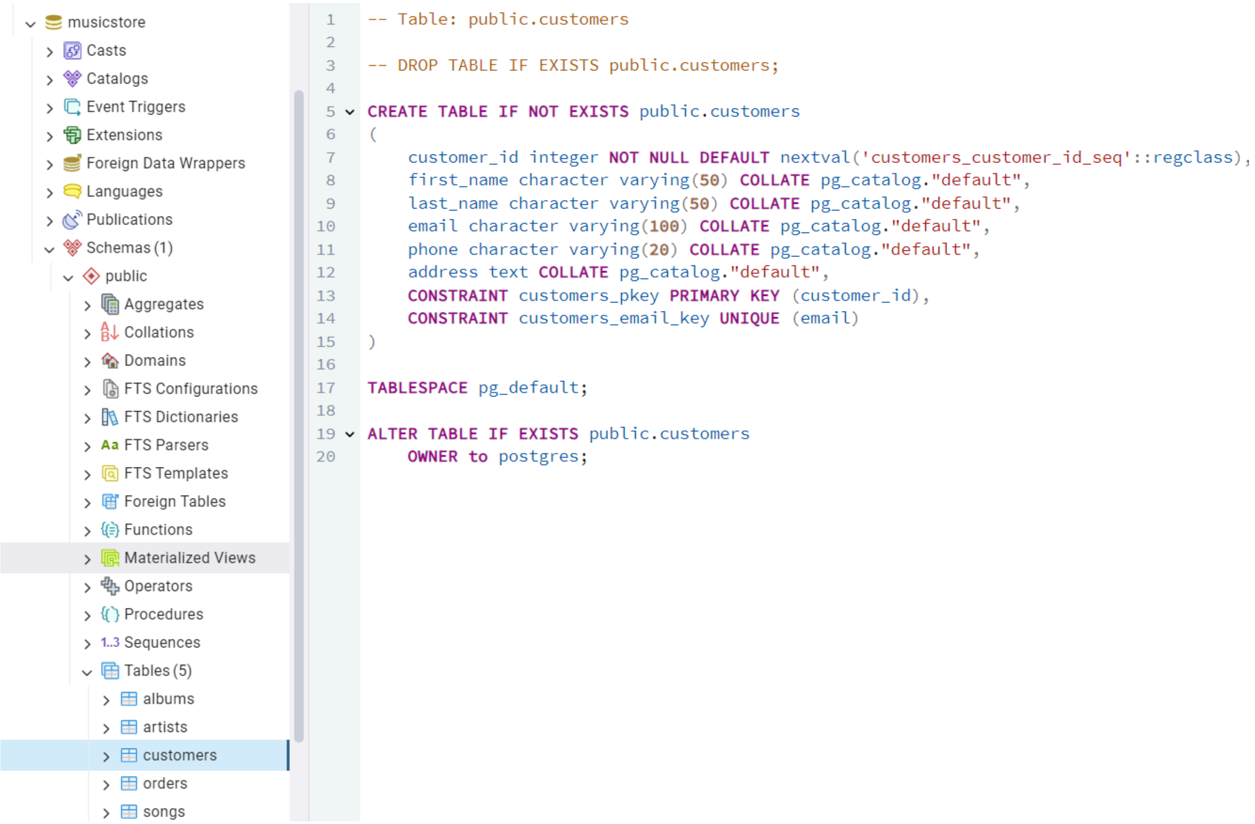


```
musicstore=# UPDATE Customers
musicstore-# SET email = 'sudarshan1@gmail.com'
musicstore-# WHERE customer_id = 1;
UPDATE 1
musicstore=# UPDATE Orders
musicstore-# SET total_amount = 34.99
musicstore-# WHERE order_id = 1;
UPDATE 1
musicstore=# DELETE FROM Songs WHERE song_id = 1;
DELETE 1
```

DROP TABLE



```
musicstore=# drop table orders;
DROP TABLE
musicstore=# select * from orders;
ERROR:  relation "orders" does not exist
LINE 1: select * from orders;
                      ^
```



```sql
-- Table: public.customers

-- DROP TABLE IF EXISTS public.customers;

CREATE TABLE IF NOT EXISTS public.customers
(
    customer_id integer NOT NULL DEFAULT nextval('customers_customer_id_seq'::regclass),
    first_name character varying(50) COLLATE pg_catalog."default",
    last_name character varying(50) COLLATE pg_catalog."default",
    email character varying(100) COLLATE pg_catalog."default",
    phone character varying(20) COLLATE pg_catalog."default",
    address text COLLATE pg_catalog."default",
    CONSTRAINT customers_pkey PRIMARY KEY (customer_id),
    CONSTRAINT customers_email_key UNIQUE (email)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.customers
    OWNER to postgres;
```

Conclusion:

By completing this project, we successfully designed and implemented a Music Store Management System using PostgreSQL. The database was structured with relational tables for Customers, Artists, Albums, Songs, and Orders, ensuring efficient data organization and retrieval.