1.1

AIM

To understand and implement the following using PostgreSQL:

1. Create and populate **Authors** and **Books** tables using DDL and DML commands.
2. Establish a **foreign key relationship** between the tables.
3. Perform an **INNER JOIN** to retrieve book titles along with their authors' names and countries.

OBJECTIVE

- To gain practical knowledge of **table creation**, **data insertion**, and **data retrieval** in PostgreSQL.
- To implement **referential integrity** using foreign keys.
- To apply **INNER JOIN** queries for combining data from multiple tables based on a common attribute.
- To understand **relational data modeling** in the context of an author-book relationship.

PROCEDURE / ALGORITHM

1. **Create Table** Authors
    - Fields: author_id (Primary Key), author_name, country
2. **Create Table** Books
    - Fields: book_id (Primary Key), title, author_id (Foreign Key referencing Authors)
3. **Insert Sample Records**
    - Insert relevant author data into the Authors table.
    - Insert corresponding books referencing the correct author_id.
4. **Use INNER JOIN Query**
    - Fetch title, author_name, and country from joined Books and Authors tables using INNER JOIN on author_id.

PROBLEM STATEMENT

Write a PostgreSQL query to retrieve the **titles of all books**, along with the **corresponding author's name** and their **country**, using INNER JOIN. The tables involved are:

- Authors(author_id, author_name, country)
- Books(book_id, title, author_id)

The output should include:

- title
- author_name
- country

QUERY

```
create table Authors( author_id INT Primary Key, name VARCHAR(50), country VARCHAR(50));


create table Books(book_id INT Primary Key, title VARCHAR(100), author_id INT, Foreign key (author_id) references Authors(author_id));


insert into Authors(author_id, name, country) values (1, 'George Orwell', 'UK'), (2, 'Haruki Murakami', 'Japan'), (3, 'Jane Austen', 'UK');


insert into Books(book_id, title, author_id) Values(101, '1984', 1), (102, 'Animal Farm', 1), (103, 'Norwegian Wood', 2), (104, 'Pride and Prejudice', 3), (105, 'Emma', 3);


FROM Books B INNER JOIN Authors A ON B.author_id = A.author_id;
```

OUTPUT

	book_title character varying (100)	author_name character varying (50)	author_country character varying (50)
1	1984	George Orwell	UK
2	Animal Farm	George Orwell	UK
3	Norwegian Wood	Haruki Murakami	Japan
4	Pride and Prejudice	Jane Austen	UK
5	Emma	Jane Austen	UK

1.2

AIM

To design and implement a normalized academic schema in PostgreSQL using Departments and Courses tables (normalized up to 3NF), insert sample data, retrieve department names offering more than two courses using a subquery, and apply access control by granting SELECT permission on the Courses table to a specific user.

OBJECTIVE

This lab demonstrates the principles of:

- **Database Normalization** up to Third Normal Form (3NF) to eliminate redundancy and maintain data integrity.
- **Foreign Key Relationships** between parent (Departments) and child (Courses) entities.
- **Subqueries** for conditional data retrieval using GROUP BY and HAVING.
- **Data Control Language (DCL)** for permission management in PostgreSQL.

Key theoretical concepts involved include:

- **1NF**: Ensures atomicity of data.
- **2NF**: Removes partial dependencies.
- **3NF**: Eliminates transitive dependencies.
- **Subqueries**: Used for data filtering based on aggregated values.
- **Access Control**: Restricting database operations using GRANT/REVOKE.

ALGORITHM

Part A

1. Open PostgreSQL terminal or pgAdmin.

2. Create Departments table with primary key on dept_id.

3. Create Courses table with a foreign key reference to Departments.dept_id.

4. Ensure department names are unique by adding a UNIQUE constraint on dept_name.

PART B

1. Insert 5 department records into Departments.

2. Insert 10 course records into Courses, each course mapped to a valid dept_id.

PROBLEM STATEMENT

You are tasked with designing a normalized academic database schema in PostgreSQL consisting of two tables: Departments and Courses, where each course is assigned to a single department. Normalize the structure up to 3NF, ensuring no department name is repeated**.**

Once the schema is designed:

- Insert a minimum of 5 departments and 10 courses such that each department offers at least 2 courses.
- Use a subquery to retrieve names of departments offering more than two courses.
- Use Data Control Language (DCL) to grant only SELECT permission on the Courses table to a user called viewer_user.

QUERY

CREATE TABLE departments dept_id INT PRIMARY KEY, dept_name VARCHAR(50) UNIQUE NOT NULL);

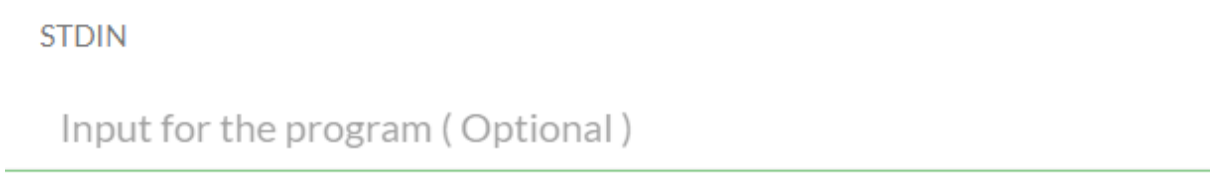
CREATE TABLE courses (course_id INT PRIMARY KEY, course_name VARCHAR(100) NOT NULL, dept_id INT, FOREIGN KEY (dept_id) REFERENCES departments(dept_id));

```
INSERT INTO departments(dept_id, dept_name) VALUES(1, 'Computer Science'),(2, 'Electrical'),(3, 'Mechanical'), (4, 'Civil'), (5, 'Electronics');


INSERT INTO courses(course_id, course_name, dept_id) VALUES (101, 'DBMS', 1), (102, 'Operating System', 1), (103, 'Power Systems', 2), (104, 'Digital Circuits', 2), (105, 'Thermodynamics', 3), (106, 'Fluid Mechanics', 3), (107, 'Structural Engineering', 4), (108, 'Surveying', 4), (109, 'Embedded Systems', 5), (110, 'VLSI Design', 5), (111, 'Advanced DBMS', 1);  -- Adding 3rd course to CS


SELECT dept_name FROM departments WHERE dept_id IN (SELECT dept_id FROM courses GROUP BY dept_id HAVING COUNT(course_id) > 2);
```

OUTPUT



Output:

```
CREATE TABLE
CREATE TABLE
INSERT 0 5
INSERT 0 11
    dept_name
------------------
 Computer Science
(1 row)
```

1.3

AIM

To perform the following operations using SQL:

1. Create Students, Courses, and Enrollments tables using DDL commands.
2. Insert sample records into each table using DML commands.
3. Retrieve **student name**, **course title**, and **grade** using **INNER JOIN**.

OBJECTIVE

The objective of this exercise is to:

- Understand how to define and create relational database tables using SQL Data Definition Language (DDL).
- Learn how to insert data using SQL Data Manipulation Language (DML).
- Practice using **INNER JOIN** to combine data from multiple tables based on relational keys.
- Understand real-world use cases of **Entity Relationship (ER)** mapping in databases.

This problem reflects a **many-to-many relationship** between students and courses via an intermediary table Enrollments.

PROCEDURE / ALGORITHM

Step 1: Start your SQL shell or database client (like MySQL, PostgreSQL, SQLite, etc.)
Step 2: Create the three tables: Students, Courses, and Enrollments with appropriate keys and data types.
Step 3: Insert valid sample data into each table.
Step 4: Write and execute an SQL SELECT query using INNER JOIN to fetch relevant combined data.

ROBLEM STATEMENT

Design a database system with three tables — Students, Courses, and Enrollments. Each student can enroll in multiple courses, and each course can have multiple students.

You must:

1. Use DDL statements to create the required tables.
2. Use DML to populate them with sample records.
3. Write a query using INNER JOIN to display the **student name**, **course title**, and **grade** for each enrollment.

Code

```
CREATE TABLE Students (student_id INT PRIMARY KEY,   name VARCHAR(50));
```

```
CREATE TABLE Courses (course_id INT PRIMARY KEY, title VARCHAR(100));


CREATE TABLE Enrollments (enroll_id INT PRIMARY KEY, student_id INT,
course_id INT, grade VARCHAR(5), FOREIGN KEY (student_id) REFERENCES
Students(student_id), FOREIGN KEY (course_id) REFERENCES
Courses(course_id));


INSERT INTO Students (student_id, name) VALUES  (1, 'Ashish'), (2, 'Naman');


INSERT INTO Courses (course_id, title) VALUES  (101, 'DBMS'), (102, 'Operating
Systems'), (103, 'Computer Networks');


INSERT INTO Enrollments (enroll_id, student_id, course_id, grade) VALUES   (1, 1,
101, 'A'), (2, 1, 102, 'B+'), (3, 1, 103, 'A'), (4, 2, 101, 'B');


SELECT s.name AS student_name,  c.title AS course_title, e.grade FROM
Enrollments e JOIN Students s ON e.student_id = s.student_id JOIN  Courses c ON
e.course_id = c.course_id;
```

OUTPUT

Output:

```
CREATE TABLE
CREATE TABLE
CREATE TABLE
INSERT 0 2
INSERT 0 3
INSERT 0 4
 student_name |   course_title    | grade
--------------+-------------------+-------
 Ashish       | DBMS              | A
 Ashish       | Operating Systems | B+
 Ashish       | Computer Networks | A
 Naman        | DBMS              | B
(4 rows)
```