WORKSHEET 1

Student Name: Supriya Dutta **UID: 23BCS13291**

Branch: CSE(3rd Year) **Section/Group: Krg-1-A**

Semester: 5th **Date of Performance: 24/07/25**

Subject Name: ADBMS **Subject Code: 23CSP-333**

1. **AIM:**
   Easy-Level Problem
   Problem Title: Author-Book Relationship Using Joins and Basic SQL Operations
   Procedure (Step-by-Step):
   a) Design two tables — one for storing author details and the other for book details.
   b) Ensure a foreign key relationship from the book to its respective author.
   c) Insert at least three records in each table.
   d) Perform an INNER JOIN to link each book with its author using the common author ID.
   e) Select the book title, author name, and author's country.
      Sample Output Description:
      When the join is performed, we get a list where each book title is shown along with its author's name and their country.

   Medium-Level Problem
   Problem Title: Department-Course Subquery and Access Control
   Procedure (Step-by-Step):
   a) Design normalized tables for departments and the courses they offer, maintaining a foreign key relationship.
   b) Insert five departments and at least ten courses across those departments.
   c) Use a subquery to count the number of courses under each department.
   d) Filter and retrieve only those departments that offer more than two courses.
   e) Grant SELECT-only access on the courses table to a specific user.
      Sample Output Description:
      The result shows the names of departments which are associated with more than two courses in the system.

2. **Tools Used :** SQL Server Management Studio

DBMS SCRIPT:

--Q1: Easy level

```
CREATE TABLE Authors (
   AuthorID INT PRIMARY KEY,
   AuthorName VARCHAR(100),
   Country VARCHAR(100)
);

CREATE TABLE Books (
   BookID INT PRIMARY KEY,
   Title VARCHAR(100),
   AuthorID INT,
   FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)
);

INSERT INTO Authors (AuthorID, AuthorName, Country)
VALUES
(1, 'J.K. Rowling', 'United Kingdom'),
(2, 'George R.R. Martin', 'United States'),
(3, 'Haruki Murakami', 'Japan');

INSERT INTO Books (BookID, Title, AuthorID)
VALUES
(101, 'Harry Potter', 1),
(102, 'Game of Thrones', 2),
(103, 'Norwegian Wood', 3);

SELECT
   B.Title AS BookTitle,
   A.AuthorName,
   A.Country
FROM
   Books B
INNER JOIN
   Authors A ON B.AuthorID = A.AuthorID;
```

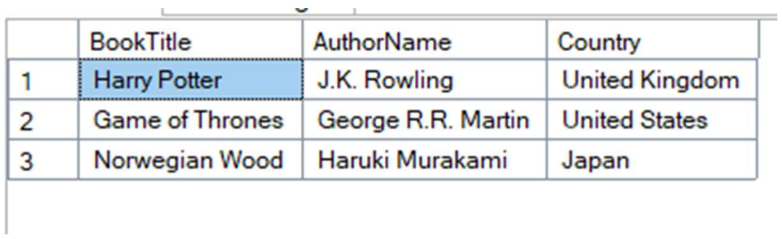
--Q2 Medium level

```
CREATE TABLE Departments (
   DeptID INT PRIMARY KEY,
   DeptName VARCHAR(100) NOT NULL
);
```

```
CREATE TABLE Courses (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR(100) NOT NULL,
    DeptID INT,
    FOREIGN KEY (DeptID) REFERENCES Departments(DeptID)
);

INSERT INTO Departments (DeptID, DeptName) VALUES
(1, 'Computer Science'),
(2, 'Mechanical Engineering'),
(3, 'Electrical Engineering'),
(4, 'Mathematics'),
(5, 'Physics');

INSERT INTO Courses (CourseID, CourseName, DeptID) VALUES
(101, 'Data Structures', 1),
(102, 'Algorithms', 1),
(103, 'Operating Systems', 1),
(104, 'Thermodynamics', 2),
(105, 'Fluid Mechanics', 2),
(106, 'Circuits', 3),
(107, 'Signals and Systems', 3),
(108, 'Linear Algebra', 4),
(109, 'Quantum Mechanics', 5),
(110, 'Classical Mechanics', 5),
(111, 'Compiler Design', 1);

SELECT DeptName
FROM Departments
WHERE DeptID IN (
    SELECT DeptID
    FROM Courses
    GROUP BY DeptID
    HAVING COUNT(*) > 2
);
```

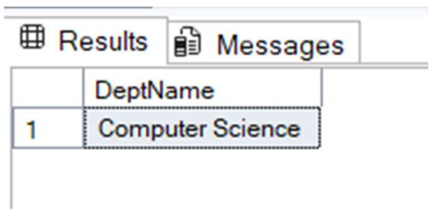
3. OUTPUT:

--Easy Level:



	BookTitle	AuthorName	Country
1	Harry Potter	J.K. Rowling	United Kingdom
2	Game of Thrones	George R.R. Martin	United States
3	Norwegian Wood	Haruki Murakami	Japan

--Medium Level:



	DeptName
1	Computer Science

4. Learning Outcomes:

.• Learn to define and create relational database tables using the CREATE TABLE statement.

• Understand how to use data types such as INT and VARCHAR effectively.

• Acquire practical skills in setting up a **primary key** to uniquely identify records.

• Learn to create and enforce **foreign key** relationships to maintain data integrity between related tables (e.g., Books → Authors).

• Develop the ability to use **INNER JOIN** to combine data from multiple tables based on a shared key (e.g., author_id).

• Understand how to design normalized relational tables with **foreign key constraints** for real-world entities like departments and courses.

• Gain experience inserting multiple records into related tables using the INSERT INTO statement.

• Learn to apply **subqueries** with GROUP BY and HAVING for data aggregation and conditional filtering.

• Apply filtering logic to retrieve records from a parent table based on subquery results from a related child table.