Experiment 1.1

Student Name: Naresh Kumar **UID:** 23BCS13655

Branch: B.E-C.S.E **Section/Group:** 23BCS_KRG-2B

Semester: 5th **Date of Performance:** 28 July, 2025

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

EASY - LEVEL

1. **Problem Title:** Author-Book Relationship Using Joins and Basic SQL Operations.
2. **Procedure (Step-by-Step):** Design two tables — one for storing author details and the other for book details.
   a. Ensure a foreign key relationship from the book to its respective author.
   b. Insert at least three records in each table.
   c. Perform an INNER JOIN to link each book with its author using the common author ID.
   d. Select the book title, author name, and author's country.
3. **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.
4. **SQL Commands:**
   a. Create the database and use it:

```
CREATE DATABASE VaibhavADBMS;
USE VaibhavADBMS;
```

   b. Create tables Author_Info and Book_List:

```
CREATE TABLE Author_Info (
    Author_id INT PRIMARY KEY,
    Author_name VARCHAR(MAX),
    Country VARCHAR(MAX)
);

CREATE TABLE Book_List (
    Book_id INT PRIMARY KEY,
    Book_title VARCHAR(MAX),
    AuthorId INT,
    FOREIGN KEY (AuthorId) REFERENCES Author_Info(Author_id)
);
```

c. Insert the values in the tables:

```
INSERT INTO Author_Info (Author_id, Author_name, Country) VALUES
(1, 'Ravinder Singh', 'India'),
(2, 'Paulo Coelho', 'Brazil'),
(3, 'Arundhati Roy', 'India'),
(4, 'Dan Brown', 'United States'),
(5, 'Khaled Hosseini', 'Afghanistan'),
(6, 'Elif Shafak', 'Turkey');

INSERT INTO Book_List (Book_id, Book_title, AuthorId) VALUES
(201, 'I Too Had a Love Story', 1),
(202, 'The Alchemist', 2),
(203, 'The God of Small Things', 3),
(204, 'The Da Vinci Code', 4),
(205, 'The Kite Runner', 5),
(206, 'The Forty Rules of Love', 6);
```

d. Selecting the book title, author name, and author's country:

```
SELECT b.Book_title AS 'Book Name', a.Author_name AS [Author Name], a.Country AS [Country]
FROM Book_List AS b
INNER JOIN Author_Info AS a
ON b.AuthorId = a.Author_id;
```

5. **Output:**

Results | Messages

	Book Name	Author Name	Country
1	I Too Had a Love Story	Ravinder Singh	India
2	The Alchemist	Paulo Coelho	Brazil
3	The God of Small Things	Arundhati Roy	India
4	The Da Vinci Code	Dan Brown	United States
5	The Kite Runner	Khaled Hosseini	Afghanistan
6	The Forty Rules of Love	Elif Shafak	Turkey

6. **Learning Outcome:**

a. I learned how to design and manage relational databases effectively using SQL.

b. I understood the use of primary and foreign key constraints to establish relationships between tables.

c. I practiced inserting multiple records efficiently to populate tables with relevant data.

d. I gained hands-on experience with INNER JOIN operations to retrieve meaningful combined data from related tables.

MEDIUM - LEVEL

1. **Problem Title:** Course Subquery and Access Control 2.
2. **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.
3. **Sample Output Description:** The result shows the names of departments which are associated with more than two courses in the system.
4. **SQL Commands:**
    a. Create the tables.

```
CREATE TABLE Dept_Info (
    DeptID INT PRIMARY KEY,
    DeptName VARCHAR(100)
);

CREATE TABLE Subject (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR(100),
    DeptID INT,
    FOREIGN KEY (DeptID) REFERENCES Dept_Info(DeptID)
);
```

    b. Insert the values.

```
INSERT INTO Dept_Info VALUES
(1, 'Electronics'),
(2, 'Mechanical'),
(3, 'Civil Engineering'),
(4, 'AI & Robotics'),
(5, 'Humanities');

INSERT INTO Subject VALUES
(301, 'Digital Circuits', 1),
(302, 'Microcontrollers', 1),
(303, 'Thermodynamics', 2),
(304, 'Fluid Mechanics', 2),
(305, 'Structural Analysis', 3),
(306, 'Construction Planning', 3),
(307, 'Machine Learning', 4),
(308, 'Neural Networks', 4),
(309, 'Philosophy', 5),
(310, 'Signal Processing', 1),
(311, 'Embedded Systems', 1),
(312, 'Geotechnical Engineering', 3),
(313, 'Mechanics of Materials', 2);
```

c. Use a subquery to count the number of courses under each department.

```
SELECT D.DeptName,
(SELECT COUNT(*) FROM Subject C WHERE C.DeptID = D.DeptID) AS CourseCount
FROM Dept_Info D;
```

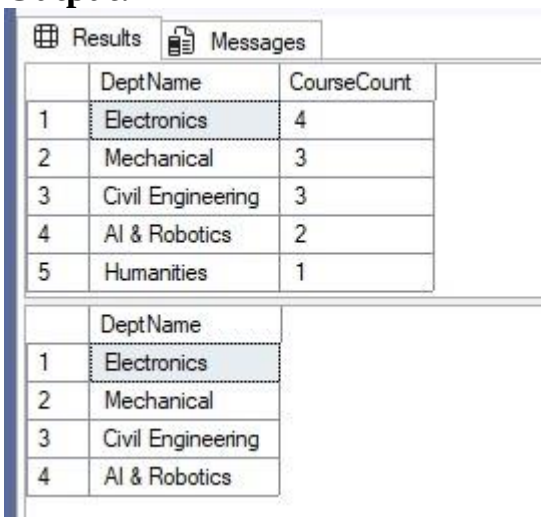
d. Filter and retrieve only those departments that offer more than two courses.

```
SELECT D.DeptName
FROM Dept_Info D
WHERE (SELECT COUNT(*) FROM Subject C WHERE C.DeptID = D.DeptID) >= 2;
```

e. Grant SELECT-only access on the courses table to a specific user.

```
CREATE LOGIN vaibhav_login WITH PASSWORD = 'Vaibhav@123';
CREATE USER vaibhav_user FOR LOGIN vaibhav_login;
EXECUTE AS USER = 'vaibhav_user';
GRANT SELECT ON Subject TO vaibhav_user;
```

5. **Output:**



Results | Messages

	DeptName	CourseCount
1	Electronics	4
2	Mechanical	3
3	Civil Engineering	3
4	AI & Robotics	2
5	Humanities	1

	DeptName
1	Electronics
2	Mechanical
3	Civil Engineering
4	AI & Robotics

6. **Learning Outcomes:**
   a. Learned to design normalized database schemas using primary and foreign keys to ensure referential integrity between related entities.
   b. Gained proficiency in inserting, updating, and managing structured data across interconnected relational tables.
   c. Mastered the use of correlated subqueries to dynamically count and retrieve related records within parent-child relationships.
   d. Applied scalar subqueries within SELECT and WHERE clauses to compute and filter aggregated results in a row-wise context.
   e. Acquired practical experience in implementing user-level access control by using GRANT for SELECT-only permissions and EXECUTE AS with REVERT to securely manage user impersonation.