EXPERIMENT - 1

Student Name: Dikshay Sharma **UID: 23BCS11096**
Branch: BE-CSE **Section/Group: KRG 1-A**
Semester: 5th **Date of Performance: 25/07/2025**
Subject Name: ADBMS **Subject Code: 23CSP-333**

1. **AIM: Ques 1** :- Author-Book Relationship Using Joins and Basic SQL Operations. Design two tables — one for storing author details and the other for book details.
2. Ensure a foreign key relationship from the book to its respective author.
3. Insert at least three records in each table.
4. Perform an INNER JOIN to link each book with its author using the common author ID.
5. Select the book title, author name, and author's country.

2. TOOLS USED:- MS SSMS & Microsoft SQL Server

3. SQL CODE:

```
CREATE TABLE TBL_AUTHOR (
    AUTHOR_ID INT PRIMARY KEY,
    AUTHOR_NAME VARCHAR(30)
);

CREATE TABLE TBL_BOOK (
        BOOK_ID INT PRIMARY KEY,
        BOOK_TITLE VARCHAR(50),
        AUTHOR_ID INT,
        FOREIGN KEY (AUTHOR_ID) REFERENCES TBL_AUTHOR(AUTHOR_ID)
    );

    INSERT INTO TBL_AUTHOR (AUTHOR_ID, AUTHOR_NAME) VALUES
    (1, 'C.J. Date'),
```

```
    (2, 'Silberschatz'),
    (3, 'A. Tanenbaum');

    INSERT INTO TBL_BOOK (BOOK_ID, BOOK_TITLE, AUTHOR_ID) VALUES
    (101, 'Database Systems', 1),
    (102, 'Operating Systems', 2),
    (103, 'Computer Networks', 3),
    (104, 'Advanced Databases', 1),
    (105, 'Modern OS', 2);

    SELECT
        book.BOOK_TITLE AS Title,
        author.AUTHOR_NAME AS Author
    FROM
        TBL_BOOK AS book
    INNER JOIN
        TBL_AUTHOR AS author ON book.AUTHOR_ID = author.AUTHOR_ID
    ORDER BY
        Author, Title;
```

4. OUTPUT:

	Title	Author
1	Computer Networks	A. Tanenbaum
2	Advanced Databases	C.J. Date
3	Database Systems	C.J. Date
4	Modern OS	Silberschatz
5	Operating Systems	Silberschatz

5. **Ques 2: -**Department-Course Subquery and Access Control.

1. Design normalized tables for departments and the courses they offer, maintaining a foreign key relationship.
2. Insert five departments and at least ten courses across those departments.
3. Use a subquery to count the number of courses under each department.
4. Filter and retrieve only those departments that offer more than two courses.
5. Grant SELECT-only access on the courses table to a specific user.
6. **SQL CODE:-**

```
CREATE TABLE University_Branches (
   branch_code INT PRIMARY KEY,
   branch_title VARCHAR(100) NOT NULL
);

CREATE TABLE Class_Listings (
   class_id INT PRIMARY KEY,
   class_subject VARCHAR(100) NOT NULL,
   branch_code INT,
   FOREIGN KEY (branch_code) REFERENCES University_Branches(branch_code)
);

INSERT INTO University_Branches (branch_code, branch_title) VALUES
(10, 'Computer Science'),
(20, 'Mechanical Engineering'),
(30, 'Electrical Engineering'),
(40, 'Civil Engineering'),
(50, 'Mathematics');

INSERT INTO Class_Listings (class_id, class_subject, branch_code) VALUES
(501, 'Data Structures', 10),
(502, 'Operating Systems', 10),
(503, 'Machine Learning', 10),
(504, 'Thermodynamics', 20),
(505, 'Fluid Mechanics', 20),
```

```
(506, 'Circuits and Systems', 30),
(507, 'Control Systems', 30),
(508, 'Structural Analysis', 40),
(509, 'Linear Algebra', 50),
(510, 'Calculus', 50),
(511, 'Probability Theory', 50);

SELECT
   branch.branch_title,
   COUNT(listing.class_id) AS number_of_classes
FROM
   University_Branches AS branch
LEFT JOIN
   Class_Listings AS listing ON branch.branch_code = listing.branch_code
GROUP BY
   branch.branch_title
ORDER BY
   branch.branch_title;

SELECT
   branch.branch_title,
   COUNT(listing.class_id) AS class_count
FROM
   University_Branches AS branch
JOIN
   Class_Listings AS listing ON branch.branch_code = listing.branch_code
GROUP BY
   branch.branch_title
HAVING
   COUNT(listing.class_id) > 2
ORDER BY
   branch.branch_title;
```

7. OUTPUT

Results | Messages

	branch_title	number_of_classes
1	Civil Engineering	1
2	Computer Science	3
3	Electrical Engineering	2
4	Mathematics	3
5	Mechanical Engineering	2

	branch_title	class_count
1	Computer Science	3
2	Mathematics	3