**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.

1. Ensure a foreign key relationship from the book to its respective author.
2. Insert at least three records in each table.
3. Perform an INNER JOIN to link each book with its author using the common author ID.
4. Select the book title, author name, and author’s country.
5. **TOOLS USED:-** MS SSMS & Microsoft SQL Server
6. **SQL CODE:**
7. CREATE TABLE TBL\_AUTHOR (
8. AUTHOR\_ID INT PRIMARY KEY,
9. AUTHOR\_NAME VARCHAR(30)
10. );
11. CREATE TABLE TBL\_BOOK (
12. BOOK\_ID INT PRIMARY KEY,
13. BOOK\_TITLE VARCHAR(50),
14. AUTHOR\_ID INT,
15. *FOREIGN KEY* (AUTHOR\_ID) *REFERENCES* TBL\_AUTHOR(AUTHOR\_ID)
16. );
17. INSERT INTO TBL\_AUTHOR (AUTHOR\_ID, AUTHOR\_NAME) VALUES
18. (1, 'C.J. Date'),
19. (2, 'Silberschatz'),
20. (3, 'A. Tanenbaum');
21. INSERT INTO TBL\_BOOK (BOOK\_ID, BOOK\_TITLE, AUTHOR\_ID) VALUES
22. (101, 'Database Systems', 1),
23. (102, 'Operating Systems', 2),
24. (103, 'Computer Networks', 3),
25. (104, 'Advanced Databases', 1),
26. (105, 'Modern OS', 2);
27. SELECT
28. book.*BOOK\_TITLE* AS Title,
29. author.*AUTHOR\_NAME* AS Author
30. FROM
31. TBL\_BOOK AS book
32. INNER JOIN
33. TBL\_AUTHOR AS author ON book.*AUTHOR\_ID* **=** author.*AUTHOR\_ID*
34. ORDER BY
35. Author, Title;

1. **OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Ques 2: -**Department-Course Subquery and Access Control.
2. Design normalized tables for departments and the courses they offer, maintaining a foreign key relationship.
3. Insert five departments and at least ten courses across those departments.
4. Use a subquery to count the number of courses under each department.
5. Filter and retrieve only those departments that offer more than two courses.
6. Grant SELECT-only access on the courses table to a specific user.
7. **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**

A screenshot of a computer

AI-generated content may be incorrect.