# EXPERIMENT - 1

**Student Name: Nagesh Kumar** **UID: 23BCS10608**
**Branch: BE-CSE** **Section/Group: KRG2-A**
**Semester: 5th** **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(30),
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 * FROM TBL_BOOK;
SELECT * FROM TBL_AUTHOR;


SELECT B.BOOK_TITLE , A.AUTHOR_NAME
FROM TBL_BOOK AS B
INNER JOIN
TBL_AUTHOR AS A
ON
B.AUTHOR_ID = A.AUTHOR_ID;
```

## 4. OUTPUT:

| | BOOK_TITLE | AUTHOR_NAME |
|---|---|---|
| 1 | Database Systems | C.J. Date |
| 2 | Operating Systems | Silberschatz |
| 3 | Computer Networks | A. Tanenbaum |
| 4 | Advanced Databases | C.J. Date |
| 5 | Modern OS | 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:-**

**Step 1**:

```
CREATE TABLE Departments (
department_id INT PRIMARY KEY,
department_name VARCHAR(100) NOT NULL
);

CREATE TABLE Courses ( course_id INT
PRIMARY KEY, course_name
VARCHAR(100) NOT NULL,
department_id INT,
FOREIGN KEY (department_id) REFERENCES Departments(department_id)
);
```

**Step 2**:

```
INSERT INTO Departments (department_id, department_name) VALUES
(1, 'Computer Science'),
(2, 'Mechanical Engineering'),
(3, 'Electrical Engineering'),
```

```
(4, 'Civil Engineering'),
(5, 'Mathematics');
```

**Step 3**

```
INSERT INTO Courses (course_id, course_name, department_id) VALUES
(101, 'Data Structures', 1),
(102, 'Operating Systems', 1),
(103, 'Machine Learning', 1),
(104, 'Thermodynamics', 2),

(105, 'Fluid Mechanics', 2),
(106, 'Circuits and Systems', 3),
(107, 'Control Systems', 3),
(108, 'Structural Analysis', 4),
(109, 'Linear Algebra', 5),
(110, 'Calculus', 5),
(111, 'Probability Theory', 5);
```

**Step 4**

```
 SELECT
    department_name,
    (SELECT COUNT(*)
     FROM Courses c
     WHERE c.department_id = d.department_id) AS course_count FROM
Departments d;
```

**Step 5**

```
 SELECT
    department_name,
    (SELECT COUNT(*)
     FROM Courses c
    WHERE c.department_id = d.department_id) AS course_count
FROM Departments d
WHERE (SELECT COUNT(*)
       FROM Courses c
       WHERE c.department_id = d.department_id) > 2;
```

## 7. OUTPUT

| | department_name | course_count |
|---|---|---|
| 1 | Computer Science | 3 |
| 2 | Mathematics | 3 |