Experiment 1

Student Name: Akshit Gautam | **UID:** 23BAI71449

Branch: BE-AIT-CSE | **Section/Group:** 23AIT_KRG-1

Semester: 5th | **Date of Performance:** 23 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):**
   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.
3. **SQL Commands:**
   a. Create Tables Author and Books using the create table command. The attributes of the two are as following:

```
--Creating table 1 for storing the information of each author
CREATE TABLE TBL_AUTHOR
(
    AUTHOR_ID INT PRIMARY KEY,--Declaring author_id as primary
    AUTHOR_NAME VARCHAR(MAX),
    COUNTRY VARCHAR(MAX)
)
--Creating table number 2 for storing the information of books
CREATE TABLE TBL_BOOKS
(
    BOOK_ID INT PRIMARY KEY,
    BOOK_TITLE VARCHAR(MAX),
    AUTHORID INT
    FOREIGN KEY (AUTHORID) REFERENCES TBL_AUTHOR(AUTHOR_ID)
)
```

b. Insert values into the tables using insert into command:

```
-- Inserting the values into both the tables
INSERT INTO TBL_AUTHOR (AUTHOR_ID, AUTHOR_NAME, COUNTRY) VALUES
(1, 'Leo Tolstoy', 'Russia'),
(2, 'Jane Austen', 'United Kingdom'),
(3, 'R.K. Narayan', 'India'),
(4, 'Ernest Hemingway', 'United States'),
(5, 'Isabel Allende', 'Chile'),
(6, 'Franz Kafka', 'Austria-Hungary');

INSERT INTO TBL_BOOKS (BOOK_ID, BOOK_TITLE, AUTHORID) VALUES
(101, 'War and Peace', 1),
(102, 'Pride and Prejudice', 2),
(103, 'Malgudi Days', 3),
(104, 'The Old Man and the Sea', 4),
(105, 'The House of the Spirits', 5),
(106, 'The Trial', 3);
```

c. Using inner join to get desired output having the book title, author name, and author's country:

```
-- Inner join to get book name, author, and country as specified
SELECT B.BOOK_TITLE AS [BOOK NAME], A.AUTHOR_NAME, A.COUNTRY
FROM TBL_BOOKS AS B
INNER JOIN TBL_AUTHOR AS A
ON B.AUTHORID = A.AUTHOR_ID;
```

4. **Output:**

	BOOK NAME	AUTHOR_NAME	COUNTRY
1	War and Peace	Leo Tolstoy	Russia
2	Pride and Prejudice	Jane Austen	United Kingdom
3	Malgudi Days	R.K. Narayan	India
4	The Old Man and the Sea	Ernest Hemingway	United States
5	The House of the Spirits	Isabel Allende	Chile
6	The Trial	R.K. Narayan	India

Output of Inner Join on tables Author and Books

5. **Learning Outcome:**
   a. I learnt how to create and manage relational databases using SQL.
   b. I learnt how to define primary and foreign key constraints to link tables.
   c. I learnt how to insert multiple records into SQL tables efficiently.
   d. I learnt how to use INNER JOIN to retrieve combined data from related tables.

MEDIUM - LEVEL

1. **Problem Title:** Department Course Subquery and Access Control
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.

    Sample Output Description: The result shows the names of departments which are associated with more than two courses in the system.
3. **SQL Commands:**
    a. Create the tables Department and Course. The attributes of the two are as following:

```sql
-- Creating department table
CREATE TABLE Department (
    DeptID INT PRIMARY KEY,
    DeptName VARCHAR(100)
);

-- Createing course table
CREATE TABLE Course (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR(100),
    DeptID INT,
    FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
);
```

    b. Insert the values into the tables.

```sql
-- Inserting values into departments
INSERT INTO Department VALUES
(1, 'Computer Science'),
(2, 'Physics'),
(3, 'Mathematics'),
(4, 'Chemistry'),
(5, 'Biology');
```

```sql
-- Inserting values into courses
INSERT INTO Course VALUES
(101, 'Data Structures', 1),
(102, 'Operating Systems', 1),
(103, 'Quantum Mechanics', 2),
(104, 'Electromagnetism', 2),
(105, 'Linear Algebra', 3),
(106, 'Calculus', 3),
(107, 'Organic Chemistry', 4),
(108, 'Physical Chemistry', 4),
(109, 'Genetics', 5),
(110, 'Molecular Biology', 5),
(111, 'ADBMS', 1),
(112, 'FUll Stack', 1),
(113, 'Discrete Mahematics', 3),
(114, 'Inorganic Chemistry', 4);
```

c. Filtering the data based of our requirement using subqueries, group by clause and WHERE condition:

```sql
--Filtering departments with more than two courses
SELECT DeptName AS [Department Name], CourseCount AS [Number of Courses]
FROM (
    SELECT D.DeptName, COUNT(C.CourseID) AS CourseCount
    FROM Department AS D
    JOIN Course AS C
    ON D.DeptID = C.DeptID
    GROUP BY D.DeptName
) AS DeptCourseCount
WHERE CourseCount > 2;
```

d. Grant SELECT-only access on the courses table to a specific user Test_User:

```sql
CREATE LOGIN TEST_LOGIN WITH PASSWORD = 'TEst@123';
USE ADBMS;
CREATE USER TEST_USER FOR LOGIN TEST_LOGIN;
GRANT SELECT ON Course TO TEST_USER;
```

4. **Output:**

	Department Name	Number of Courses
1	Chemistry	3
2	Computer Science	4
3	Mathematics	3

Output of Department with courses>2

5. **Learning Outcomes:**

a. Learned to design normalized database schemas using primary and foreign keys to maintain referential integrity between related entities.
b. Developed proficiency in inserting and managing structured data across relational tables.
c. Mastered the use of **correlated subqueries** to dynamically count related records for each row in a parent table.
d. Applied **scalar subqueries** within SELECT and WHERE clauses to filter and compute aggregated results per row context.
e. Gained practical experience in implementing **user-level access control**, using GRANT to assign SELECT-only privileges and EXECUTE AS with REVERT to switch and restore user contexts securely.