

Experiment 1.1

**Student Name:** Priyanka **UID:** 23BAI70303

**Branch:** BE-AIT-CSE **Section/Group:** 23AIT\_KRG-G1\_A

**Semester:** 5th **Date of Performance:** 27 July, 2025

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

# Aim:

## EASY LEVEL PROBLEM:

To create author and book tables linked by a foreign key, insert sample data, and use an INNER JOIN to display each book’s title with its author’s name and country, demonstrating basic SQL joins and relational design.

## MEDIUM LEVEL PROBLEM:

Create normalized tables for departments and courses linked by a foreign key, insert sample data, use a subquery to count and filter departments offering more than two courses, and grant SELECT-only access to a specific user on the courses table, demonstrating subqueries, filtering, and access control in SQL.

# Objective:

Design related tables for authors-books and departments-courses with foreign keys; insert sample data; use INNER JOIN to link books with authors, subqueries to find departments with over two courses, and grant SELECT-only access to a user—demonstrating core SQL concepts of relational design, data retrieval, and access control.

# Theory:

This exercise involves foundational concepts of relational databases and SQL operations. Relational databases organize data into tables (called relations), where each table consists of rows (records or tuples) and columns (attributes or fields).

Tables are linked by keys: a primary key uniquely identifies each record in a table, and a foreign key establishes a relationship between tables by referring to a primary key in

another table. This structure supports efficient data storage, retrieval, and ensures data integrity through referential constraints.

SQL (Structured Query Language) is the standard language used to create, manipulate, and query relational databases. Key SQL operations used here include:

* + **Table creation and data insertion:** Defining tables with appropriate columns and constraints (like foreign keys), then inserting sample data into them.
  + **INNER JOIN:** A fundamental join operation that links rows between two tables based on a matching key, allowing combined information to be retrieved (e.g., books linked with their authors).
  + **Subqueries with aggregation:** Using nested queries to compute summary data (such as counting courses per department) and filtering results based on aggregate conditions.
  + **Access control:** Managing database security by granting specific privileges (e.g., SELECT-only permission) to users, limiting their ability to modify data.

Together, these concepts demonstrate relational database design principles, how to model relationships using keys, retrieve meaningful combined data across tables, analyze data subsets through subqueries, and implement basic security measures in an SQL environment. This approach enables organized, consistent, and secure management of interconnected data.

This theory provides the conceptual foundation behind creating author-book and department-course database schemas, performing joins and subqueries for data retrieval, and applying access restrictions in practice.

# Procedure:

## Design Tables:

* + Create an Author table to store author details (e.g., **AUTOR\_ID**, **AUTOR\_AME**, Country).
  + Create a Book table to store book details (e.g., **BOOK\_ID**, Title, **AUTOR\_ID**), with a foreign key **AUTOR\_ID** referencing the Author table
  + Create a department table to store department details (e.g., **DEflARTMET\_ID**, **DEflARTMET\_ID**).
  + Create a Course table to store course details (e.g., **COURSE\_ID**, **COURSE\_AME**, **DEflARTMET\_ID**), with a foreign key **DEflARTMET\_ID** referencing the Department table.

## Insert Sample Data:

* + Insert at least three records into the Author table.
  + Insert at least three records into the Book table linking to authors through

**AUTOR\_ID**.

* + Insert five departments into the Department table.
  + Insert at least ten courses into the Course table, distributed among the departments via **DEflARTMET\_ID**.

## Perform SQL Operations:

* + Use an INNER JOIN query to retrieve and display each book’s title, corresponding author’s name, and author’s country by joining the Book and Author tables on **AUTOR\_ID**.
  + Use a subquery with aggregation (COUNT) on the Course table grouped by **DEflARTMET\_ID** to find the number of courses per department.
  + Filter the departments to retrieve only those having more than two courses based on the subquery result.

## Apply Access Control:

* + Grant SELECT permission on the Course table to a specific user to restrict data access to read-only.

# Code:

-- CREATE DATABASE AIT\_1A -- Already exists, no need to run again

USE AIT\_1A

GO

-- Only insert if author table has no data

IF NOT EXISTS (SELECT 1 FROM TBL\_AUTHOR)

BEGIN

INSERT INTO TBL\_AUTHOR (AUTHOR\_ID, AUTHOR\_NAME, COUNTRY) VALUES

(1, 'J.K. Rowling', 'United Kingdom'),

(2, 'George R.R. Martin', 'United States'),

(3, 'Haruki Murakami', 'Japan'),

(4, 'Isabel Allende', 'Chile'),

(5, 'Chinua Achebe', 'Nigeria'),

(6, 'Gabriel Garcia Marquez', 'Colombia'),

(7, 'Toni Morrison', 'United States'),

(8, 'Leo Tolstoy', 'Russia'),

(9, 'Jane Austen', 'United Kingdom'),

(10, 'Mark Twain', 'United States');

END

GO

IF NOT EXISTS (SELECT 1 FROM TBL\_BOOKS)

BEGIN

INSERT INTO TBL\_BOOKS (BOOK\_ID, BOOK\_TITLE, AUTHORID) VALUES

(1, 'Harry Potter and the Sorcerer''s Stone', 1),

(2, 'A Game of Thrones', 2),

(3, 'Norwegian Wood', 3),

(4, 'The House of the Spirits', 4),

(5, 'Things Fall Apart', 5),

(6, 'One Hundred Years of Solitude', 6),

(7, 'Beloved', 7),

(8, 'War and Peace', 8),

(9, 'Pride and Prejudice', 9),

(10, 'Adventures of Huckleberry Finn', 10);

END

GO

-- Show results (no harm re-running)

SELECT B.BOOK\_TITLE AS [Book Title], A.AUTHOR\_NAME AS [Author Name], A.COUNTRY AS [Country]

FROM TBL\_BOOKS AS B

INNER JOIN TBL\_AUTHOR AS A ON B.AUTHORID = A.AUTHOR\_ID

GO

-- Medium Level

-- Only insert if department table has no data

IF NOT EXISTS (SELECT 1 FROM TBL\_DEPARTMENT)

BEGIN

INSERT INTO TBL\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT\_NAME) VALUES

(1, 'Computer Science'),

(2, 'Mathematics'),

(3, 'Physics'),

(4, 'Chemistry'),

(5, 'English Literature');

END

GO

IF NOT EXISTS (SELECT 1 FROM TBL\_COURSE)

BEGIN

INSERT INTO TBL\_COURSE (COURSE\_ID, COURSE\_NAME, DEPARTMENT\_ID) VALUES

(1, 'Data Structures', 1),

(2, 'Operating Systems', 1),

(3, 'Algorithms', 1),

(4, 'Calculus', 2),

(5, 'Linear Algebra', 2),

(6, 'Quantum Mechanics', 3),

(7, 'Electromagnetism', 3),

(8, 'Organic Chemistry', 4),

(9, 'Physical Chemistry', 4),

(10, 'Shakespearean Literature', 5),

(11, 'Modern Poetry', 5);

END

GO

-- Count query (safe)

SELECT COUNT(COURSE\_NAME) AS Total, DEPARTMENT\_NAME AS [Department Name]

FROM TBL\_COURSE

INNER JOIN TBL\_DEPARTMENT ON TBL\_COURSE.DEPARTMENT\_ID = TBL\_DEPARTMENT.DEPARTMENT\_ID

GROUP BY TBL\_DEPARTMENT.DEPARTMENT\_NAME

GO

-- Subquery for departments with more than 2 courses

SELECT DEPARTMENT\_NAME

FROM TBL\_DEPARTMENT

WHERE DEPARTMENT\_ID IN (

SELECT DEPARTMENT\_ID

FROM TBL\_COURSE

GROUP BY DEPARTMENT\_ID

HAVING COUNT(\*) > 2

)

GO

-- Create login and user if not exist

IF NOT EXISTS (SELECT \* FROM sys.server\_principals WHERE name = 'TEST\_LOGIN\_PRIYANKA')

BEGIN

CREATE LOGIN TEST\_LOGIN\_PRIYANKA WITH PASSWORD = 'TESTLOGIN@123PRIYANKA';

END

GO

IF NOT EXISTS (SELECT \* FROM sys.database\_principals WHERE name = 'TEST\_LOGIN\_PRIYANKA')

BEGIN

CREATE USER TEST\_LOGIN\_PRIYANKA FOR LOGIN TEST\_LOGIN\_PRIYANKA;

END

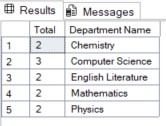
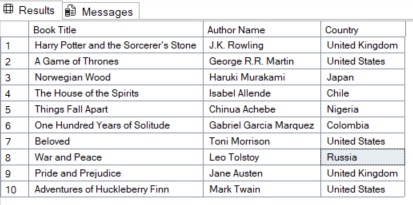
GO

-- Grant SELECT if not already granted (this will not error even if already granted)

GRANT SELECT ON TBL\_COURSE TO TEST\_LOGIN\_PRIYANKA;

GO

# Output:

****

1. **Learning Outcomes:**
2. Understand how to **design a relational schema** for a real-world system.
3. Practice **creating and linking tables** using SQL.
4. Use **JOINs to query multi-table data** meaningfully.
5. Implement **data access control** using GRANT/REVOKE.