

Experiment 1.1

**Student Name:** Shreshta **UID:** 23BAI70375

**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:

Create related tables for authors and books, and departments and courses, using foreign keys to establish relationships. Insert sample data, apply an INNER JOIN to connect books with their respective authors, and use subqueries to identify departments offering more than two courses. Additionally, provide SELECT-only permissions to a user on the courses table, showcasing key SQL concepts like relational schema design, data querying, and access control.

# Theory:

This exercise covers essential concepts of relational databases and core SQL operations. Relational databases store data in structured tables (called relations), where each table consists of rows (records) and columns (fields). Tables are connected using keys: a **primary key** uniquely identifies each record, while a **foreign key** links records across tables by referencing a primary key in another table. This design ensures efficient storage, retrieval, and maintains data integrity through referential constraints.

**SQL (Structured Query Language)** is the standard language for creating, managing, and querying relational databases. Key operations used in this exercise 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.

These concepts demonstrate the fundamentals of relational database design—modeling relationships using keys, retrieving combined data across tables, analyzing specific subsets with subqueries, and applying basic security practices. This forms the theoretical basis for building and querying author-book and department-course schemas while enforcing data access rules.

# Procedure:

## Design Tables:

* Create an **Author** table with columns for AUTHOR\_ID, AUTHOR\_NAME, and Country.
* Create a **Book** table with columns for BOOK\_ID, Title, and AUTHOR\_ID, where AUTHOR\_ID is a foreign key linking to the Author table.
* Create a **department** table with fields like DEPARTMENT\_ID and DEPARTMENT\_NAME.
* Create a **Course** table with COURSE\_ID, COURSE\_NAME, and DEPARTMENT\_ID, with DEPARTMENT\_ID as a foreign key referencing the Department table.

## Insert Sample Data:

* Insert at least **3 authors** into the Author table.
* Add at least **3 books**, each connected to an author using AUTHOR\_ID.
* Insert **5 departments** into the Department table.
* Add at least **10 courses**, making sure they are assigned to various departments through DEPARTMENT\_ID.

## Perform SQL Operations:

* Use an **INNER JOIN** to combine the Book and Author tables and display each book’s title along with the author's name and country.
* Run a **subquery** to count how many courses each department offers, using GROUP BY on DEPARTMENT\_ID.
* **Filter** the results to show only those departments that offer **more than two courses**.

## 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

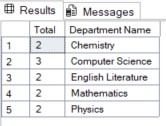
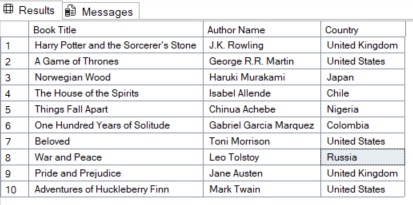
GO

-- 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.