

University Institute of Engineering Department of Computer Science & Engineering

EXPERIMENT: 1

NAME: PAYAL SINGH UID: 23BCS13626

BRANCH: BE-CSE SECTION/GROUP: KRG-1-A

SEMESTER: 5TH SUBJECT CODE: 23CSP-339

SUBJECT NAME: ADBMS

1. Aim Of The Practical:

[EASY] Author-Book Relationship Using Joins and Basic SQL Operations

- 1. 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.

[MEDIUM] 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.
- 2. Tools Used: SQL Server Management Studio

3. Code:

First Question

```
CREATE TABLE Authors (
  AuthorID INT PRIMARY KEY,
  AuthorName VARCHAR(100),
  Country VARCHAR(100)
);
CREATE TABLE Books (
  BookID INT PRIMARY KEY,
  Title VARCHAR(100),
  AuthorID INT,
  FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)
);
INSERT INTO Authors (AuthorID, AuthorName, Country) VALUES
(1, 'J.K. Rowling', 'United Kingdom'),
(2, 'George R.R. Martin', 'United States'),
(3, 'Haruki Murakami', 'Japan');
INSERT INTO Books (BookID, Title, AuthorID) VALUES
(101, 'Harry Potter', 1),
(102, 'Game of Thrones', 2),
(103, 'Norwegian Wood', 3);
SELECT B. Title AS Book Title, A. Author Name, A. Country
FROM Books B
INNER JOIN Authors A
   ON B.AuthorID = A.AuthorID;
```

Second Question

CREATE TABLE Departments(DeptID INT PRIMARY KEY, DeptName VARCHAR(100) NOT NULL);

CREATE TABLE Courses (CourseID INT PRIMARY KEY, CourseName VARCHAR(100) NOT NULL, DeptID INT,

FOREIGN KEY (DeptID) REFERENCES Departments(DeptID));

INSERT INTO Departments (DeptID, DeptName) VALUES

- (1, 'Computer Science'),
- (2, 'Mechanical Engineering'),
- (3, 'Electrical Engineering'),
- (4, 'Mathematics'),
- (5, 'Physics');

INSERT INTO Courses (CourseID, CourseName, DeptID) VALUES

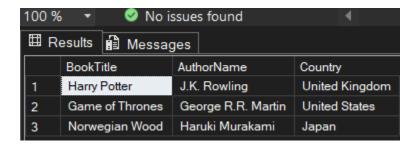
- (101, 'Data Structures', 1),
- (102, 'Algorithms', 1),
- (103, 'Operating Systems', 1),
- (104, 'Thermodynamics', 2),
- (105, 'Fluid Mechanics', 2),
- (106, 'Circuits', 3),
- (107, 'Signals and Systems', 3),
- (108, 'Linear Algebra', 4),
- (109, 'Quantum Mechanics', 5),
- (110, 'Classical Mechanics', 5),
- (111, 'Compiler Design', 1);

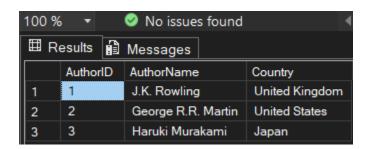
SELECT DeptName

FROM Departments

WHERE DeptID IN (SELECT DeptID FROM Courses GROUP BY DeptID HAVING COUNT(*) > 2);

4. Output: [EASY]



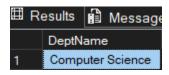




[MEDIUM]

🖽 Results 🛍 Messages				
	DeptID	DeptName		
1	1	Computer Science		
2	2	Mechanical Engineering		
3	3	Electrical Engineering		
4	4	Mathematics		
5	5	Physics		

⊞ Results Messages				
	CourselD	CourseName	DeptID	
1	101	Data Structures	1	
2	102	Algorithms	1	
3	103	Operating Systems	1	
4	104	Thermodynamics	2	
5	105	Fluid Mechanics	2	
6	106	Circuits	3	
7	107	Signals and Systems	3	
8	108	Linear Algebra	4	
9	109	Quantum Mechanics	5	
10	110	Classical Mechanics	5	
11	111	Compiler Design	1	



5. Learning Outcomes:

- Learn how to define and create relational database tables using CREATE TABLE syntax. Understand the use of data types like INT and VARCHAR.
- Gain practical knowledge of establishing a primary key for uniquely identifying records.
- Understand how to create and enforce foreign key relationships to maintain data integrity between related tables (Books → Authors).
- Develop the ability to use INNER JOIN to combine data from multiple tables based on a common key (e.g. author id).
- Understand how to design normalized relational tables with foreign key constraints for real-world entities like departments and courses.
- Gain proficiency in inserting multiple records into related tables using the INSERT INTO statement.
- Learn how to use subqueries with GROUP BY and HAVING to aggregate data and apply conditional logic.
- Apply filtering logic to retrieve records from a parent table based on results from a subquery on a related child table.