University Institute of Engineering Department of

Computer Science & Engineering

EXPERIMENT : 1

NAME : ISHITA NISHANT **UID: 23BCS11354**

BRANCH : BE-CSE **SECTION/GROUP : KRG_2A**

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 :

```
----------EASY----------

CREATE TABLE AuthorDetails (
        AuthorID INT PRIMARY KEY,
        AuthorName VARCHAR(50),
        Country VARCHAR(50)
);
```

```
CREATE TABLE BookDetails (
  BookID INT PRIMARY KEY,
  BookTitle VARCHAR(200),
  AuthorID INT,
 FOREIGN KEY (AuthorID) REFERENCES
 AuthorDetails(AuthorID));


SELECT * FROM AuthorDetails;
SELECT * FROM BookDetails;


SELECT B.BookTitle, A.AuthorName, A.Country
FROM AuthorDetails AS A
INNER JOIN
BookDetails AS B
ON A.AuthorID=B.AuthorID;
----------------------------MEDIUM----------------------------
-
CREATE TABLE Department (
  DeptID INT PRIMARY KEY,
  DeptName VARCHAR(100) NOT NULL
);
CREATE TABLE Course (
  CourseID INT PRIMARY KEY,
  CourseName VARCHAR(150) NOT NULL,
  DeptID INT,
  FOREIGN KEY (DeptID) REFERENCES
Department(DeptID)
);
INSERT INTO Department VALUES
(1, 'Computer Science'),
(2, 'Mathematics'),
```

```
(3, 'Physics'),
(4, 'Chemistry'),
(5, 'Biology');

SELECT * FROM Department;

INSERT INTO Course VALUES
(101, 'Data Structures', 1),
(102, 'Operating Systems', 1),
(103, 'Algorithms', 1),
(104, 'Calculus I', 2),
(105, 'Linear Algebra', 2),
(106, 'Quantum Mechanics', 3),
(107, 'Classical Mechanics', 3),
(108, 'Modern Poetry', 4),
(109, 'Cell Biology', 5),
(110, 'Genetics', 5);

SELECT * FROM Course;

SELECT DeptName
FROM Department
WHERE DeptID IN (
  SELECT DeptID
  FROM Course
  GROUP BY DeptID
  HAVING COUNT(CourseID) > 2
);
```

4. **Output :**

[EASY]

AuthorID	AuthorName	Country
1	Ishita	India
2	Shreya	Bharat
3	Priyanshu	India
NULL	NULL	NULL

BookID	BookTitle	AuthorID
1	Mastering SQL in 30 Days	1
2	Data Structures Made Easy	2
3	Journey Through AI	3
NULL	NULL	NULL

BookTitle	AuthorName	Country
Mastering SQL in 30 Days	Ishita	India
Data Structures Made Easy	Shreya	Bharat
Journey Through AI	Priyanshu	India

[MEDIUM]

DeptID	DeptName
1	Computer Science
2	Mathematics
3	Physics
4	Chemistry
5	Biology
NULL	NULL

	CourseID	CourseName	DeptID
▶	101	Data Structures	1
	102	Operating Systems	1
	103	Algorithms	1
	104	Calculus I	2
	105	Linear Algebra	2
	106	Quantum Mechanics	3
	107	Classical Mechanics	3
	108	Modern Poetry	4
	109	Cell Biology	5
	110	Genetics	5
*	NULL	NULL	NULL

	DeptName
▶	Computer Science

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.