University Institute of Engineering

Department of Computer Science & Engineering

Experiment: 1

Student Name: Ayush Kohli **UID: 23BCS11238**

Branch: Computer Science & Engineering **Section/Group:** KRG-3B

Semester: 5th **Subject Code:** 23CSP-339

Subject Name: ADBMS

1. Aim of the practical:

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.

Sample Output Description: When the join is performed, we get a list where each book title is shown along with its s author's name and their country.

2. Tool Used: SQL Server Management Studio.

3. CODE:

```
CREATE TABLE Authors
    ( author_id
    INT PRIMARY
    KEY, name
    VARCHAR(100),
    country VARCHAR(100)
);

CREATE TABLE Books (
    book_id INT
    PRIMARY KEY,
    title
    VARCHAR(150),
    author_id INT,
    FOREIGN KEY (author_id) REFERENCES Authors(author_id)
);
```

```
INSERT INTO Books (book_id, title, author_id) VALUES
(101, 'x', 1),
(102, 'Y', 2),
(103, 'Z', 3);
```

```
SELECT
  B.title AS Book_Title,
  A.name AS Author_Name,
  A.country AS Author_Country
FROM
  Books B
INNER JOIN
  Authors A ON B.author_id = A.author_id;
```

4. 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 table (Books → Authors).
- Develop the ability to use INNER JOIN to combine data from multiple tables based on a common key (e.g., author_id).

University Institute of Engineering

Department of Computer Science & Engineering

Experiment: 1.2

Student Name: Ayush kohli **UID: 23BCS11238**

Branch: Computer Science & Engineering **Section/Group:** KRG-3B

Semester: 5th **Subject Code:** 23CSP-339

Subject Name: ADBMS

1. Aim of the practical:

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.

Sample Output Description:

The result shows the names of departments which are associated with more than two courses in the system.

2. Tool Used: SQL Server Management Studio.

3. CODE:

```
CREATE TABLE
    Departments (
    dept_id INT
    PRIMARY KEY,
    dept_name VARCHAR(100) NOT NULL
);


CREATE TABLE
    Courses (
    course_id INT
    PRIMARY KEY,
    course_name VARCHAR(150)
    NOT NULL, dept_id INT,
    FOREIGN KEY (dept_id) REFERENCES Departments(dept_id)
);
```

```
, 'English'),
, 'Biology');

SERT INTO Courses (course_id, course_name, dept_id) VALUES
01, 'Data Structures', 1),
02, 'Operating Systems', 1),
03, 'Algorithms', 1),
04, 'Calculus I', 2),
05, 'Linear Algebra', 2),
06, 'Quantum Mechanics', 3),
07, 'Classical Mechanics', 3),
08, 'Modern Poetry', 4),
09, 'Cell Biology', 5),
10, 'Genetics', 5);

LECT dept_name
OM Departments
ERE dept_id IN (
  SELECT dept_id
  FROM Courses
  GROUP BY dept_id
  HAVING COUNT(course_id) > 2
)
```

4. LEARNING OUTCOMES:-

- 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.