University Institute of Engineering

Department of Computer Science & Engineering

EXPERIMENT : 1

NAME : GUNJAN KAMBOJ	**UID: 23BCS13605**
BRANCH : BE-CSE	**SECTION/GROUP : KRG 2 A**
SEMESTER : 5TH	**SUBJECT CODE : 23CSP-333**
SUBJECT NAME : ADBMS	

1. Aim Of The Practical :

1.1 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.

1.2

1. Create a table Employees to store employee details with columns: EmpID, EName, Department, and ManagerID.
2. Add a self-referencing foreign key constraint on ManagerID that references EmpID in the same table.
3. Insert at least six records into the table, where some employees have managers and some do not.
4. Perform a **LEFT OUTER JOIN** to link each employee with their respective manager using ManagerID and EmpID.
5. Select the employee's name and the corresponding manager's name.

2. Tools Used : SQL Server Management Studio

3. Code :

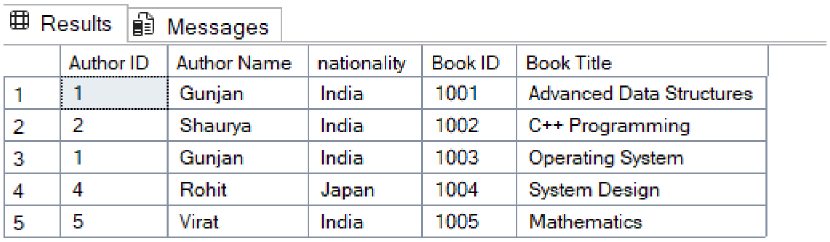
1.1

```
CREATE TABLE Author (
  author_id INT PRIMARY KEY,
  author_name VARCHAR(30),
  nationality VARCHAR(30)
);

CREATE TABLE Book (
  book_id INT PRIMARY KEY,
  title VARCHAR(50),
  author_id INT,
  FOREIGN KEY (author_id) REFERENCES Author(author_id)
);


INSERT INTO Author (author_id, author_name, nationality)
VALUES
  (1, 'Gunjan', 'India'),
  (2, 'Shaurya', 'India'),
  (3, 'Manan', 'India'),
  (4, 'Rohit', 'Japan'),
  (5, 'Virat', 'India');

INSERT INTO Book (book_id, title, author_id)
VALUES
  (1001, 'Advanced Data Structures', 1),
  (1002, 'C++ Programming', 2),
  (1003, 'Operating System', 1),
  (1004, 'System Design', 4),
  (1005, 'Mathematics', 5);

SELECT * FROM Author;
SELECT * FROM Book;

SELECT
  A.author_id AS [Author ID],
  A.author_name AS [Author Name],
  A.nationality,
  B.book_id AS [Book ID],
 B.title AS [Book Title]
FROM Author A
    JOIN Book B ON A.author_id = B.author_id;
```

1.2

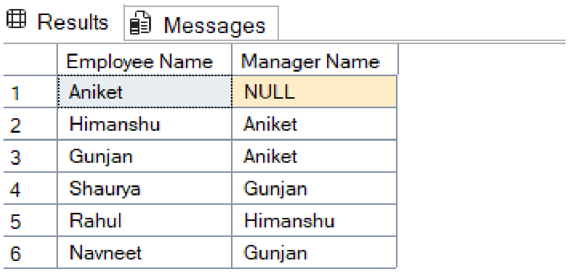
```
CREATE TABLE Employees (
  EmpID INT PRIMARY KEY,
  EName VARCHAR(50),
  Department VARCHAR(50),
  ManagerID INT
);

ALTER TABLE Employees
ADD CONSTRAINT fKey FOREIGN KEY (ManagerID) REFERENCES Employees(EmpID);

INSERT INTO Employees VALUES
  (1, 'Aniket', 'EE', NULL),
  (2, 'Himanshu', 'ECE', 1),
  (3, 'Gunjan', 'CSE', 1),
  (4, 'Shaurya', 'CSE', 3),
  (5, 'Rahul', 'ME', 2),
  (6, 'Navneet', 'ECE', 3);

SELECT A.EName AS [Employee Name],
    B.EName AS [Manager Name]
FROM Employees AS A
LEFT OUTER JOIN Employees AS B
      ON A.ManagerID = B.EmpID;
```

N Output : 1.1



	Author ID	Author Name	nationality	Book ID	Book Title
1	1	Gunjan	India	1001	Advanced Data Structures
2	2	Shaurya	India	1002	C++ Programming
3	1	Gunjan	India	1003	Operating System
4	4	Rohit	Japan	1004	System Design
5	5	Virat	India	1005	Mathematics

1.2



	Employee Name	Manager Name
1	Aniket	NULL
2	Himanshu	Aniket
3	Gunjan	Aniket
4	Shaurya	Gunjan
5	Rahul	Himanshu
6	Navneet	Gunjan

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 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.