Experiment 2

Student Name: Shashwat Mishra **UID: 23BCS10866**
Branch: CSE **Section/Group: KRG 3-A**
Semester: 5th **Date ofPerformance:24/07/2025**
Subject Name: ADBMS **Subject Code: 23CSP-333**

1. **Aim**: To demonstrate the use of self-joins and conditional joins in SQL for managing hierarchical employee relationships and performing conditional lookups using LEFT JOIN and IFNULL across two related tables**.** a. Employee-Manager Hierarchy Using Self-Join
   b. Conditional Join Between Financial Tables
2. **Objective:**

   The objective of this SQL code is to develop a practical understanding of intermediate relational database concepts through two realistic scenarios. The first part demonstrates employee-management relationships using a self-join on a single table, showing how hierarchical data can be modeled and queried effectively. The second part showcases conditional data retrieval using LEFT JOIN and IFNULL to extract NPV values from time-based financial records, even when some data is missing. Together, these examples emphasize efficient data modeling, referential integrity, and robust querying practices.

3. **DBMS script and output:**

Solution-(a)

```
-- Medium Level Problem
  -- Creating Employee Table with self-reference
  CREATE TABLE STAFF (
     StaffID INTEGER PRIMARY KEY,
     StaffName TEXT NOT NULL,
     Division TEXT NOT NULL,
     SupervisorID INTEGER,
     FOREIGN KEY (SupervisorID) REFERENCES STAFF(StaffID)
  );

  -- Inserting data into STAFF table
  INSERT INTO STAFF (StaffID, StaffName, Division, SupervisorID) VALUES
```

```
(10, 'Riya', 'Operations', NULL),
(11, 'Kunal', 'Sales', 10),
(12, 'Meera', 'Tech', 10),
(13, 'Raghav', 'Sales', 11),
(14, 'Ishita', 'Tech', 12),
(15, 'Varun', 'Operations', 10);

-- Query to show employees and their respective managers
SELECT
    S.StaffName AS EmployeeName,
    S.Division AS EmployeeDivision,
    M.StaffName AS ManagerName,
    M.Division AS ManagerDivision
FROM
    STAFF S
JOIN
    STAFF M
ON
    S.SupervisorID = M.StaffID;
```

	EmployeeName	EmployeeDivision	ManagerName	ManagerDivision
1	Kunal	Sales	Riya	Operations
2	Meera	Tech	Riya	Operations
3	Raghav	Sales	Kunal	Sales
4	Ishita	Tech	Meera	Tech
5	Varun	Operations	Riya	Operations

Solution-(b)

```
-- Hard Level Problem
-- Creating Investment Data Table

CREATE TABLE YEARLY_INVESTMENT (
    InvestorID INT,
```

```
    InvYear INT,
    Profit INT
);

-- Creating Query Table
CREATE TABLE INVEST_QUERY (
    InvestorID INT,
    InvYear INT
);

-- Inserting data into YEARLY_INVESTMENT
INSERT INTO YEARLY_INVESTMENT (InvestorID, InvYear, Profit)
VALUES
(101, 2021, 500),
(107, 2023, 75),
(113, 2022, 90),
(101, 2022, 620),
(102, 2010, 310),
(103, 2011, 25),
(111, 2023, 88),
(107, 2022, 0);

-- Inserting queries
INSERT INTO INVEST_QUERY (InvestorID, InvYear)
VALUES
(101, 2022),
```

```
(102, 2010),
(103, 2011),
(107, 2021),
(107, 2022),
(107, 2023),
(113, 2022);

-- Query to return profit based on investor and year
SELECT
    Q.InvestorID,
    Q.InvYear,
    IF NULL(Y.Profit, 0) AS ProfitEarned
FROM
    INVEST_QUERY Q
LEFT JOIN
    YEARLY_INVESTMENT Y
ON
    Q.InvestorID = Y.InvestorID AND Q.InvYear = Y.InvYear;
```

ID	YEAR	NPV_Value
1	2019	113
2	2008	121
3	2009	12
7	2018	0
7	2019	0
7	2020	30
13	2019	40

4. Learning Outcomes (What I have Learnt):

- To demonstrate the use of **self-joins** in handling hierarchical relationships like employee–manager mappings within a single table.

- To understand how to define **foreign key constraints** to maintain referential integrity between entities.
- To use **LEFT JOINs with NULL handling (IFNULL)** for matching records across related datasets and managing missing data gracefully.
- To apply **real-world data modeling** scenarios, such as employee reporting structure and year-wise data lookups.
- To enhance proficiency in **querying and combining data** from multiple tables using conditional logic and joins.