EXPERIMENT- 02

Student Name: Ruchi Sharma	**UID: 23BCS10713**
Branch: BE-CSE	**Section/Group: KRG 1(B)**
Semester: 05	**Date of Performance: 29/07/25**
Subject Name: ADBMS	**Subject Code: 23CSP-333**

Organizational Hierarchy Explorer (Medium)

1. Aim:

You are a Database Engineer at TalentTree Inc., an enterprise HR analytics platform that stores employee data, including their reporting relationships. The company maintains a centralized Employee relation that holds:
Each employee's ID, name, department, and manager ID (who is also an employee in the same table).

Your task is to generate a report that maps employees to their respective managers, showing:

- The employee's name and department
- Their manager's name and department (if applicable)

This will help the HR department visualize the internal reporting hierarchy.

2. Objective:

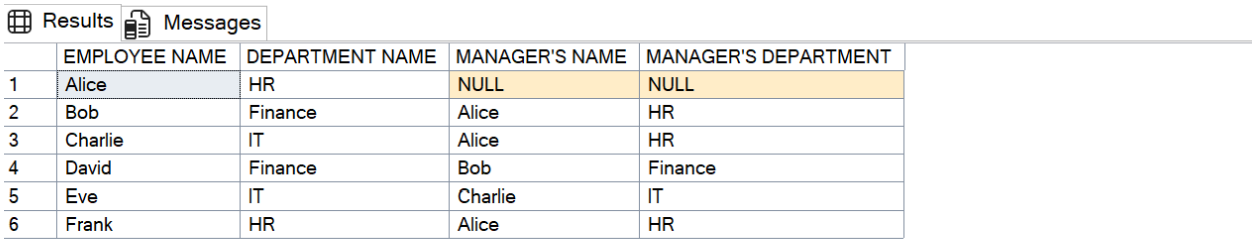
- To understand and apply self-join operations on a single table.
- To represent hierarchical relationships (employee-manager) within the same entity.
- To visualize organizational structure using SQL by mapping employees with their respective managers.

3. DBMS script:

```
CREATE TABLE EMPLOYEE (
   EMP_ID INT PRIMARY KEY,
   ENAME VARCHAR(10),
   DEPT VARCHAR(10),
   MANAGERID INT,
   FOREIGN KEY (MANAGERID) REFERENCES EMPLOYEE(EMP_ID)
);

INSERT INTO EMPLOYEE VALUES
```

```
(1, 'Alice', 'HR', NULL),
(2, 'Bob', 'Finance', 1),
(3, 'Charlie', 'IT', 1),
(4, 'David', 'Finance', 2),
(5, 'Eve', 'IT', 3),
(6, 'Frank', 'HR', 1);

SELECT
    E1.ENAME AS [EMPLOYEE NAME],
    E1.DEPT AS [DEPARTMENT NAME],
    E2.ENAME AS [MANAGER'S NAME],
    E2.DEPT AS [MANAGER'S DEPARTMENT]
FROM
    EMPLOYEE AS E1
LEFT JOIN
    EMPLOYEE AS E2
ON
    E1.MANAGERID = E2.EMP_ID;
```

4. Output:



	EMPLOYEE NAME	DEPARTMENT NAME	MANAGER'S NAME	MANAGER'S DEPARTMENT
1	Alice	HR	NULL	NULL
2	Bob	Finance	Alice	HR
3	Charlie	IT	Alice	HR
4	David	Finance	Bob	Finance
5	Eve	IT	Charlie	IT
6	Frank	HR	Alice	HR

Financial Forecast Matching with Fallback Strategy (Hard)

1. Aim:

You are a Data Engineer at FinSight Corp, a company that models Net Present Value (NPV) projections for investment decisions. Your system maintains two key datasets:

Year_tbl: Actual recorded NPV's of various financial instruments over different years:

ID: Unique Financial instrument identifier.

YEAR: Year of record

NPV: Net Present Value in that year

Queries_tbl: A list of instrument-year pairs for which stakeholders are requesting NPV values:

ID: Financial instrument identifier

YEAR: Year of interest.

Find the NPV of each query from the Queries table. Return the output order by ID and Year in the sorted form.

However, not all ID-YEAR combinations in the Queries table are present in the Year_tbl. If an NPV is missing for a requested combination, assume it to be 0 to maintain a consistent financial report.

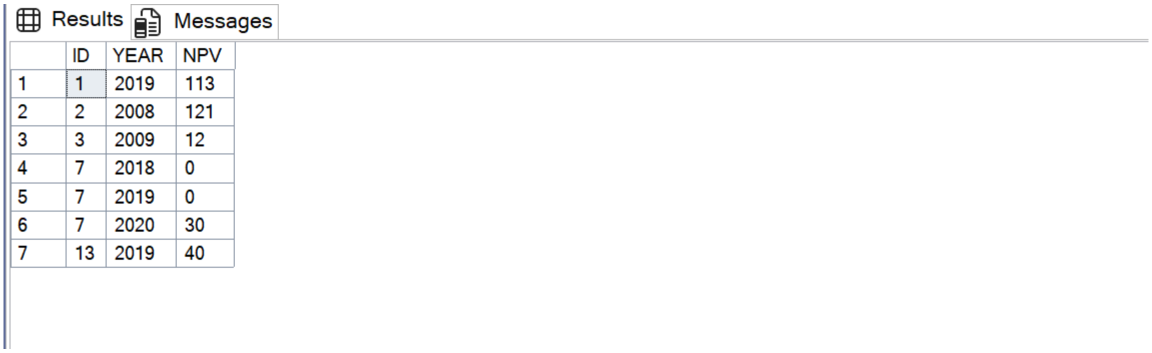
2. Objective:

- To retrieve financial data by performing joins across multiple datasets.
- To handle missing data scenarios using fallback strategies like ISNULL() in SQL.
- To understand and apply LEFT JOIN operations for data reconciliation.
- To ensure accurate and complete reporting of Net Present Values (NPV) even when data is unavailable.
- To return results in a sorted and standardized format, facilitating better decision-making in financial forecasting.

3. DBMS script:

```
CREATE TABLE Year_tbl (
    ID INT,
    YEAR INT,
    NPV INT
);

CREATE TABLE Queries (
    ID INT,
    YEAR INT
);

INSERT INTO Year_tbl (ID, YEAR, NPV)
VALUES
(1, 2018, 100),
(7, 2020, 30),
(13, 2019, 40),
(1, 2019, 113),
(2, 2008, 121),
(3, 2009, 12),
(11, 2020, 99),
(7, 2019, 0);

INSERT INTO Queries (ID, YEAR)
VALUES
(1, 2019),
(2, 2008),
(3, 2009),
(7, 2018),
(7, 2019),
(7, 2020),
(13, 2019);

SELECT Q.ID, Q.YEAR, ISNULL(Y.NPV, 0) AS NPV
FROM Queries AS Q
LEFT JOIN Year_tbl AS Y
ON Q.ID = Y.ID AND Q.YEAR = Y.YEAR
ORDER BY Q.ID, Q.YEAR;
```

4. Output:



	ID	YEAR	NPV
1	1	2019	113
2	2	2008	121
3	3	2009	12
4	7	2018	0
5	7	2019	0
6	7	2020	30
7	13	2019	40