DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

Experiment -2

Student Name: Divyansh Manhas	**UID:** 23BAI70239
Branch: BE-AIT-CSE	**Section/Group:** 23AIT_KRG-G1_A
Semester: 5th	**Date of Performance:** 30 July, 2025
Subject Name: ADBMS	**Subject Code:** 23CSP-333

1. Aim:

MEDIUM LEVEL PROBLEM:

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.
.

HARD LEVEL PROBLEM:

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:

1. **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

2. **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:

As a Data Engineer, you will generate reports across HR and finance domains. Your tasks include mapping employees to managers for an organizational hierarchy report and retrieving Net Present Value (NPV) figures for stakeholder queries. You must handle missing data, such as defaulting an NPV to 0, and ensure all final reports are properly sorted and structured.

3. Theory:

- This **Joins (LEFT JOIN & SELF JOIN):**
- **LEFT JOIN** is used to ensure all records from a primary table (like Queries or Employee) are included in the output, even if there's no match in the second table. This is how you kept all requested NPV queries and all employees in your reports.
- **SELF JOIN** is a pattern for querying hierarchical data within a single table, which you used to link employees to their managers.

- **Keys (PRIMARY & FOREIGN):**
- **PRIMARY KEY** ensures every row has a unique identifier (EmpID).
- **FOREIGN KEY** enforces data integrity by making sure a ManagerID refers to an actual, existing employee.

- **Handling Missing Data (NULL & ISNULL):**
- NULL is a marker for absent information. The LEFT JOIN creates NULLs when an NPV value is not found.
- **ISNULL()** is a function that applies a business rule by replacing these NULLs with a default value, like 0.

4. Procedure:

- Define the Employee Hierarchy Schema:

- Populate the Employee Table:

- Generate the Employee-Manager Report:

- Define the Financial Data Schema:

- Populate the Financial Data and Query Tables:

- Generate the NPV Calculation Report:

5. Code Medium:

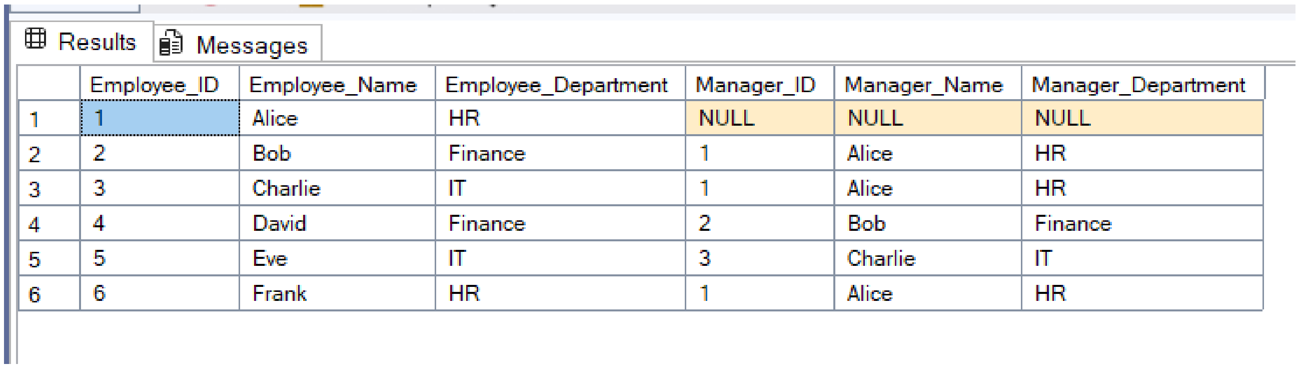
```
-- Medium
-- JOIN EMPLOY

USE Exp2;

-- SELF JOIN

CREATE TABLE Employee(
	EmpID INT PRIMARY KEY,
	EmpName VARCHAR(30) NOT NULL,
	Department VARCHAR(30) NOT NULL,
	ManagerID INT NULL
);

ALTER TABLE Employee
ADD CONSTRAINT FK_Employee FOREIGN KEY (ManagerID) REFERENCES EMPLOYEE(EmpID);

INSERT INTO Employee (EmpID, EmpName, Department, ManagerID)
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.EmpID as [Employee_ID], E1.EmpName as [Employee_Name], E1.Department as
[Employee_Department],
	E2.EmpID as [Manager_ID], E2.EmpName as [Manager_Name], E2.Department as
[Manager_Department]
FROM
	Employee as E1
```

```
LEFT OUTER JOIN
       Employee as E2
ON
       E1.ManagerID = E2.EmpID;
```

6.Code Hard:

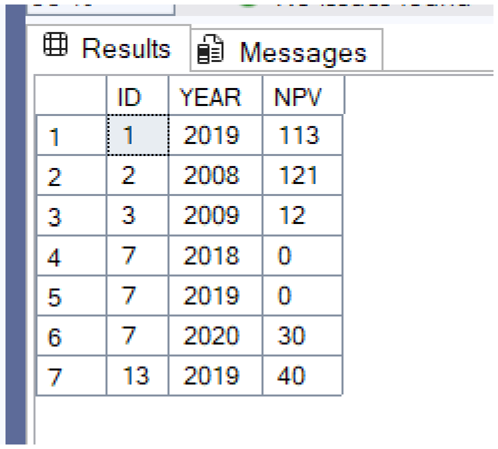
```
-- HARD
--(B) part

-- Create Year_tbl (holds actual NPV values)
CREATE TABLE Year_tbl (
    ID INT,
    YEAR INT,
    NPV INT
);

-- Create Queries table (requested values)
CREATE TABLE Queries (
    ID INT,
    YEAR INT
);

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


-- LEFT OUTER JOIN


SELECT
    Q.ID, Q.YEAR, ISNULL(Y.NPV, 0) as NPV
From
    Queries as Q
LEFT OUTER JOIN
    Year_tbl as Y
ON
    Q.ID = Y.ID AND Q.YEAR = Y.YEAR;
```

7. Output:



Results | Messages

	Employee_ID	Employee_Name	Employee_Department	Manager_ID	Manager_Name	Manager_Department
1	1	Alice	HR	NULL	NULL	NULL
2	2	Bob	Finance	1	Alice	HR
3	3	Charlie	IT	1	Alice	HR
4	4	David	Finance	2	Bob	Finance
5	5	Eve	IT	3	Charlie	IT
6	6	Frank	HR	1	Alice	HR



Results | Messages

	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

8. Learning Outcomes:

a. **Design and model complex data relationships**, including hierarchies using selfreferencing FOREIGN KEY constraints to ensure data integrity.

b. **Retrieve comprehensive datasets using advanced joins**, such as SELF JOIN for hierarchical queries and LEFT JOIN to create complete reports.

c. **Clean and structure query results for reliable reporting** by handling missing data with functions like ISNULL()