Experiment -2

Student Name: S S Sneha

Branch: BE-AIT-CSE

Semester: 5th

Subject Name: ADBMS

UID: 23BAI70259

Section/Group:23AIT_KRG-G1_B

Date of Performance: 29 July 2025

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 systemmaintains twokeydatasets:

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:

1. **Define the Employee Hierarchy Schema**:

   - The script first executes CREATE TABLE Employee to define the structure for storing employee data.
   - Immediately after, ALTER TABLE adds a self-referencing FOREIGN KEY constraint, linking the ManagerID column to the EmpID column.

2. **Populate the Employee Table**:

   - INSERT statements are used to add six employee records.

3. **Generate the Employee-Manager Report**:

   - The first SELECT query runs. It performs a SELF JOIN on the Employee table (using aliases E1 for the employee and E2 for the manager) to link each employee to their manager.
   - A LEFT JOIN is used to ensure all employees are included in the result, even 'Alice' who has no manager.
   - The query selects the name and department from both aliases to produce the final organizational chart report.

4. **Define the Financial Data Schema**:

   - The script proceeds to create two new tables for the financial task.
   - CREATE TABLE Year_tbl sets up the table to store historical NPV data, and CREATE TABLE Queries sets up the table to hold the list of stakeholder requests.

5. **Populate the Financial Data and Query Tables**:

   - INSERT statements are run to populate both Year_tbl with known financial records and Queries with the specific ID-YEAR pairs that need to be looked up.

6. **Generate the NPV Calculation Report**:

   - The final SELECT query runs to produce the NPV report. It works as follows:

     - It starts with the Queries table to ensure every request is addressed.
     - It uses a LEFT JOIN to look up the corresponding NPV from Year_tbl by matching both ID and YEAR.
     - It applies the ISNULL() function to replace any NULL values (for queries with no matching data) with 0.

5. Code:

```sql
CREATE TABLE Employee (
    EmpID INT PRIMARY KEY,
    EmpName VARCHAR(50) NOT NULL,
    Department VARCHAR(50) 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.EmpName as [EMPLOYEE NAME], E2.EmpName as [Manager Name],
E1.Department as [Employee Dept], E2.ManagerId as [Manager ID]
FROM Employee as E1
LEFT OUTER JOIN
Employee as E2
ON
E1.ManagerID = E2.EmpID

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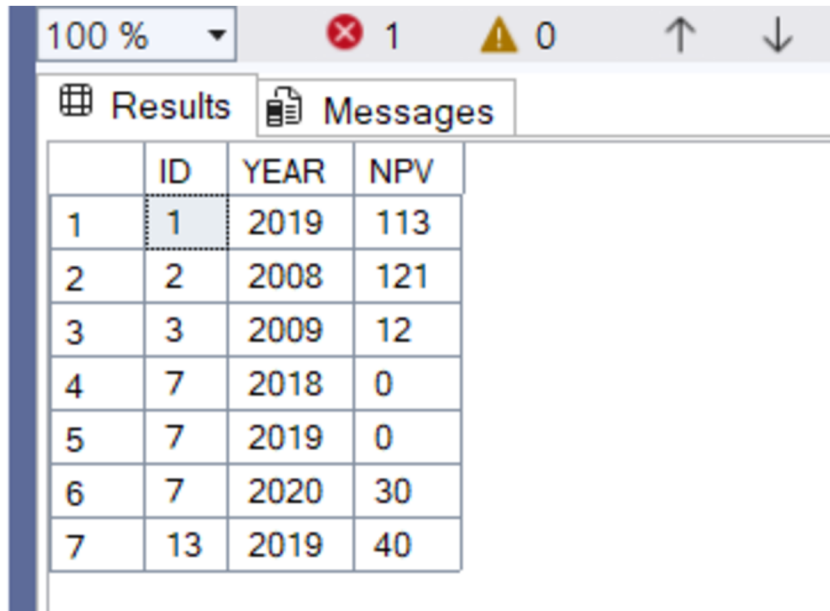
```sql
INSERT INTO Year_tbl (ID, YEAR, NPV)
VALUES
(1, 2018, 100),
(7, 2020, 30),
(13, 2019, 40),
(1, 2019, 113),
(2, 2008, 121),
(3, 2009, 12),
(11, 2020, 99),
(7, 2019, 0);

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

SELECT Q.ID  AS [ID], Q.YEAR AS [YEAR] , ISNULL(Y.NPV,0) AS [NPV]
FROM Queries AS Q
LEFT OUTER JOIN
Year_tbl AS Y
ON
Q.YEAR = Y.YEAR
AND
Q.ID = Y.ID
```
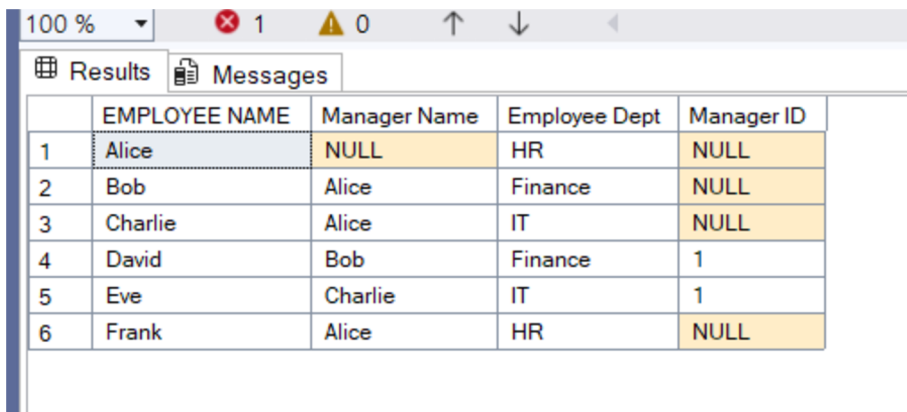
6. Output:

	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

	EMPLOYEE NAME	Manager Name	Employee Dept	Manager ID
1	Alice	NULL	HR	NULL
2	Bob	Alice	Finance	NULL
3	Charlie	Alice	IT	NULL
4	David	Bob	Finance	1
5	Eve	Charlie	IT	1
6	Frank	Alice	HR	NULL

7. Learning Outcomes:

- Design and model complex data relationships, including hierarchies using self- referencing FOREIGN KEY constraints to ensure data integrity
- Retrieve comprehensive datasets using advanced joins, such as SELF JOIN for hierarchical queries and LEFT JOIN to create complete reports.
- Clean and structure query results for reliable reporting by handling missing data with functions like ISNULL()