

**Experiment -2**

|  |  |
| --- | --- |
| **Student Name: Shreshta** | **UID:** 23BAI70375 |
| **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

1. **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.

1. **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.

1. **Theory:**

* **LEFT JOIN**: Ensures all records from the main table (like Employees or Queries) appear in the result, even if there's no match in the related table. This helps retain all employee or NPV query data in reports.
* **SELF JOIN**: Used to relate rows within the same table, such as connecting employees to their managers in a hierarchy.
* **PRIMARY KEY**: A unique identifier for each row (e.g., EmpID), ensuring no duplicates.
* **FOREIGN KEY**: Maintains referential integrity by ensuring values like ManagerID match a valid EmpID.
* **NULL**: Represents missing or unavailable data, such as when an NPV isn’t found.
* **ISNULL()**: Replaces NULL values with defaults (e.g., 0) to keep reports complete and meaningful.

1. **Procedure:**

1. **Create Employee Table**
   * CREATE TABLE Employee defines the structure for employee data.
   * ALTER TABLE adds a self-referencing FOREIGN KEY, linking ManagerID to EmpID to establish manager relationships.
2. **Insert Employee Data**
   * INSERT statements add six employee records into the Employee table.
3. **Generate Employee-Manager Report**
   * A SELF JOIN is performed on the Employee table using aliases (E1 for employee, E2 for manager).
   * A LEFT JOIN ensures all employees appear in the report, including those without a manager (e.g., Alice).
   * The SELECT retrieves names and departments for both employees and managers to form an organizational chart.
4. **Create Financial Tables**
   * CREATE TABLE Year\_tbl is used to store historical NPV values.
   * CREATE TABLE Queries holds stakeholder NPV lookup requests.
5. **Insert Financial Records**
   * INSERT statements add data into Year\_tbl (financial records) and Queries (lookup requests).
6. **Generate NPV Report**
   * The final SELECT starts from the Queries table to ensure all requests are included.
   * A LEFT JOIN matches NPV data from Year\_tbl based on ID and YEAR.
   * ISNULL() replaces missing NPV values with 0 to handle unmatched records cleanly.

**5. Code:**

--Medium Level Problem

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

--HARD LEVEL PROBLEM

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);

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

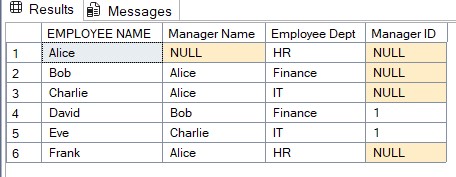
ON

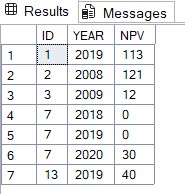
Q.YEAR = Y.YEAR

AND

Q.ID = Y.ID

1. **Output:**





1. **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()