



Experiment 1.2

Student Name: Tushar

UID: 23BAI70332

Branch: BE-AIT-CSE

Section/Group: 23AML-1 (B)

Semester: 5th

Date of Performance: 29 July 2025

Subject Name: ADBMS

Subject Code: 23CSP-333

1. Experiment Name: JOINS

2. Objective:

---Medium-Level Problem ---

Problem Title: Organizational Hierarchy Explorer

(Step-by-Step):

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:

- I. The employee's name and department
- II. Their manager's name and department (if applicable)
- III. This will help the HR department visualize the internal reporting hierarchy.

---Hard-Level Problem ---

Problem Title: Financial Forecast Matching with Fallback Strategy (Step-by-Step):

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.

3. Code:

----MEDIUM LEVEL PROBLEM CODE----

```
USE DB_DEMO
```

```
CREATE TABLE Employee (  
    EmployeeID INT PRIMARY KEY,  
    EmployeeName VARCHAR(100),  
    Department VARCHAR(100),  
    ManagerID INT -- References EmployeeID of another employee  
);
```

```
INSERT INTO Employee (EmployeeID, EmployeeName, 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  
    E.EmployeeName,  
    E.Department AS EmployeeDept,  
    M.EmployeeName AS ManagerName,  
    M.Department AS ManagerDept  
FROM  
    Employee E  
LEFT JOIN  
    Employee M ON E.ManagerID = M.EmployeeID;
```

----HARD LEVEL PROBLEM CODE----

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

```
CREATE TABLE Queries_tbl (  
    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_tbl (ID, YEAR) VALUES  
(1, 2019),  
(2, 2008),  
(3, 2009),  
(7, 2018),  
(7, 2019),  
(7, 2020),  
(13, 2019);
```

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

4. Output:

----MEDIUM level problem output----

	EmployeeID	EmployeeName	Department	ManagerID
1	1	Alice	HR	NULL
2	2	Bob	Finance	1
3	3	Charlie	IT	1
4	4	David	Finance	2
5	5	Eve	IT	3
6	6	Frank	HR	1

	EmployeeName	EmployeeDept	ManagerName	ManagerDept
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

----HARD level problem output----

	ID	YEAR	NPV
1	1	2018	100
2	7	2020	30
3	13	2019	40
4	1	2019	113
5	2	2008	121
6	3	2009	12
7	11	2020	99
8	7	2019	0

	ID	YEAR
1	1	2019
2	2	2008
3	3	2009
4	7	2018
5	7	2019
6	7	2020
7	13	2019

	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

4. Learning Outcomes:

- Mastery of LEFT JOINs:
- Handling Missing Data with COALESCE:
- Data Integration from Multiple Tables:
- Creating and Populating Tables:
- Sorted and Structured Output Generation: