EXPERIMENT- 02

Student Name: Devjot Singh **UID: 23BCS10864**

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:

STEP 1: Create Employee table

```
CREATE TABLE Employee (
```

```
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(100),
    department VARCHAR(50),
    manager_id INT NULL,
    FOREIGN KEY (manager_id) REFERENCES Employee(employee_id)
);
```

STEP 2: Insert sample data with Indian names

```
INSERT INTO Employee (employee_id, employee_name, department, manager_id) VALUES
(1, 'Rajesh Kumar', 'Engineering', NULL),
(2, 'Anita Sharma', 'Engineering', 1),
(3, 'Vikram Patel', 'HR', NULL),
(4, 'Sneha Mehta', 'HR', 3),
(5, 'Aman Verma', 'Engineering', 1),
(6, 'Priya Iyer', 'Engineering', 2);
```

STEP 3: Generate employee-manager relationship report

```
SELECT
    e.employee_name AS Employee_Name,
    e.department AS Employee_Department,
    m.employee_name AS Manager_Name,
    m.department AS Manager_Department
FROM
    Employee e
LEFT JOIN
    Employee m ON e.manager_id = m.employee_id;
```

4. Output:

100 % No issues found

Results Messages

	Employee_Name	Employee_Department	Manager_Name	Manager_Department
1	Rajesh Kumar	Engineering	NULL	NULL
2	Anita Sharma	Engineering	Rajesh Kumar	Engineering
3	Vikram Patel	HR	NULL	NULL
4	Sneha Mehta	HR	Vikram Patel	HR
5	Aman Verma	Engineering	Rajesh Kumar	Engineering
6	Priya Iyer	Engineering	Anita Sharma	Engineering

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 decisionmaking in financial forecasting.

3. DBMS script:

STEP 1: Create Year_tbl (actual NPV records)

```
CREATE TABLE Year_tbl (
   ID INT,
   YEAR INT,
   NPV DECIMAL(18,2)
);
```

STEP 2: Create Queries_tbl (requested lookups)

```
CREATE TABLE Queries_tbl (
   ID INT,
   YEAR INT
);
```

STEP 3: Insert sample data into Year_tbl

```
INSERT INTO Year_tbl (ID, YEAR, NPV) VALUES
(1, 2020, 100000.00),
(1, 2021, 120000.00),
(2, 2020, 95000.00),
(3, 2022, 110000.00);
```

STEP 4: Insert sample data into Queries_tbl

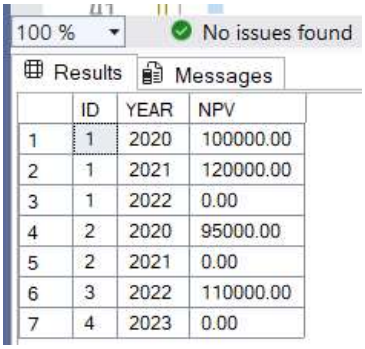
```
INSERT INTO Queries_tbl (ID, YEAR) VALUES
(1, 2020),
(1, 2021),
(1, 2022),
(2, 2020),
(2, 2021),
(3, 2022),
(4, 2023);
```

STEP 5: Query - Find NPV values for each query (default to 0 if missing)

```
SELECT
   q.ID,
   q.YEAR,
   ISNULL(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:



	ID	YEAR	NPV
1	1	2020	100000.00
2	1	2021	120000.00
3	1	2022	0.00
4	2	2020	95000.00
5	2	2021	0.00
6	3	2022	110000.00
7	4	2023	0.00