
# University Institute of Engineering

## Department of Computer Science & Engineering

## EXPERIMENT:2

**NAME : VANSH KUMAR** **UID : 23BCS10117**
**BRANCH : BE-CSE** **SECTION : KRG_1A**
**SEMESTER : 5TH** **SUBJECT : 23CSP-339**
**SUBJECT NAME : ADBMS**

**1. AIM:-**

[ MEDIUM ]

i.You are a Database Engineer at TalentTree Inc., an enterprise HR analytics platform that stores employee data, including their reporting relationships.

ii. The company maintains a centralized Employee relation that holds:

- Each employee's ID
- Name
- Department
- Manager ID (who is also an employee in the same table)

iii. 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)

iv. This will help the HR department visualize the internal reporting hierarchy.

[HARD]

You are given two tables:

1. **YEAR_TABLE** — contains the Net Present Value (NPV) of certain IDs for specific years.
    - Columns: ID, YEAR, NPV
2. **QUERIES_TABLE** — contains a list of (ID, YEAR) pairs for which we need to look up NPV values.
    - Columns: ID, YEAR

Write an SQL query to return, for each (ID, YEAR) in QUERIES_TABLE:

- The ID
- The YEAR
- The corresponding NPV from YEAR_TABLE if it exists
- If there is no matching record in YEAR_TABLE, return 0 as the NPV

The output should include all rows from QUERIES_TABLE regardless of whether a match exists in YEAR_TABLE.

**2.TOOLS USED :-**

SQL server management studio.

**3.CODE:-**

```
--MEDIUM--

CREATE TABLE Employee (
   EmpID INT PRIMARY KEY,
   EmpName VARCHAR(50) NOT NULL,
   Department VARCHAR(50) NOT NULL,
   ManagerID INT NULL  -- Self-reference to EmpID
);


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


-- Insert data into Employee table
INSERT INTO Employee (EmpID, EmpName, Department, ManagerID)
VALUES
(1, 'Alice', 'HR', NULL),       -- Top-level manager
(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],E1.Department AS[EMployeeDept],E2.EmpName
AS [ManagerName],E2.Department AS[MangaerDept]
FROM Employee AS E1
LEFT JOIN
Employee AS E2
ON
E1.ManagerID = E2.EmpID
/* We can use any join at place of self join because no keyword for self join*/

--Hard--
CREATE TABLE YEAR_TABLE(
ID INT,
YEAR INT,
NPV INT
);
INSERT INTO YEAR_TABLE(ID,YEAR,NPV)
```

```
VALUES
(1,2018,100),
(7,2020,30),
(13,2019,40),
(1,2019,13),
(2,2008,121),
(3,2009,12),
(11,2020,99),
(7,2019,0);

CREATE TABLE QUERIES_TABLE(
ID INT,
YEAR INT
);
INSERT INTO QUERIES_TABLE( ID,YEAR)
VALUES
(1,2019),
(2,2008),
(3,2009),
(7,2018),
(7,2019),
(7,2020),
(13,2019);

SELECT Q.ID,Q.YEAR,ISNULL(Y.NPV,0) AS[NPV]
FROM QUERIES_TABLE AS Q
LEFT OUTER JOIN
YEAR_TABLE AS Y
ON
Q.ID = Y.ID
AND
Y.YEAR = Q.YEAR
```

**4.OUTPUT:-**
**[MEDIUM]**

Results | Messages

| | Employee Name | EMployeeDept | ManagerName | MangaerDept |
|---|---|---|---|---|
| 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]**

Results Messages

| | ID | YEAR | NPV |
|---|---|---|---|
| 1 | 1 | 2019 | 13 |
| 2 | 2 | 2008 | 121 |
| 3 | 3 | 2009 | 12 |
| 4 | 7 | 2018 | 0 |
| 5 | 7 | 2019 | 0 |
| 6 | 7 | 2020 | 30 |
| 7 | 13 | 2019 | 40 |

**5.LEARNING OUTCOMES:-**

1. Understand and implement self-joins to model hierarchical relationships within a single table (e.g., employees reporting to other employees).
2. Construct relational queries to fetch meaningful information such as employee manager relationships, including handling NULL values using LEFT JOIN.
3.Design and populate tables using the CREATE TABLE and INSERT INTO statements for real-world hierarchical and time-series data scenarios.
4. Perform multi-table joins to retrieve and match data across different datasets, such as actual vs. requested values (e.g., NPV values for specific years).
5.Handle missing data using functions like ISNULL() to substitute default values during join operations.
6. Apply conditional joins involving multiple keys (e.g., joining on both ID and YEAR) to ensure accurate data mapping.
7. Develop problem-solving approaches using SQL to derive insights from HR records and financial datasets in enterprise applications.