

## **EXPERIMENT -2**

**STUDENT NAME:** ASHLIN JAMES

**UID:** 23BAI70722

**BRANCH:** BE-AIT-CSE

**SECTION/GROUP:** 23AIT\_KRG-G1\_A

**SEMESTER:** 5

**DATE OF PERFORMANCE:** 30 JULY, 2025

**SUBJECT NAME:** ADBMS

**SUBJECT CODE:** 23CSP-333

### **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. Tbl\_Year: 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: 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.

## **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.

## **THEORY:**

### **KEYS (PRIMARY & FOREIGN):**

- The PRIMARY KEY uniquely identifies each row in a table, such as using EmpID for employees.
- The FOREIGN KEY maintains data consistency by ensuring that related data (like ManagerID) always refers to a valid existing record in another (or the same) table.

### **JOINS (LEFT JOIN & SELF JOIN):**

- LEFT JOIN returns all records from the main (left) table (e.g., Employees or Queries), including those that do not have corresponding matches in the related (right) table. This guarantees that you see all items from your primary list, even if related details are missing.
- SELF JOIN allows a table to be joined to itself, which is helpful for representing hierarchical relationships—such as linking employees to their managers within the same Employee table.

### **HANDLING MISSING DATA (NULL & ISNULL):**

- NULL means missing or undefined information. For example, using a LEFT JOIN, fields that lack corresponding data from the related table will appear as NULL.
- The ISNULL() function replaces NULL values with a specific substitute, such as zero, applying your business rule for handling missing or incomplete data.

## **PROCEDURE:**

### 1. Define the Employee Hierarchy Schema:

- The script starts by running a CREATE TABLE Employee statement to define how employee data will be stored.
- Immediately after, an ALTER TABLE command is executed to add a self-referencing FOREIGN KEY constraint.
- This constraint links the ManagerID column to the EmpID column in the same table, establishing a relationship between employees and their managers.

### 2. Populate the Employee Table:

- The INSERT statements are used to add six records to the Employee table.
- Each statement inserts details for one employee, such as EmpID, Name, and ManagerID.
- These records populate the table with initial employee data, including hierarchical relationships through ManagerID where applicable.

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

- The query performs a SELF JOIN on the Employee table, using the aliases E1 for employees and E2 for their managers. This join links each employee to their respective manager within the same table.
- A LEFT JOIN is applied to ensure that all employees are included in the results, even those like Alice who do not have a manager.
- The query selects the name and department columns from both E1 and E2. The final output presents an organizational chart showing employees alongside their managers.

### 4. Define the Financial Data Schema: (Medium Level Query)

- The script continues by executing a CREATE TABLE Year\_tbl statement to establish a table for storing historical NPV (Net Present Value) data.
- It then runs a CREATE TABLE Queries statement to create a table intended to hold a list of stakeholder requests related to the financial task.

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

- INSERT statements are executed to add known financial records into the Year\_tbl table.
- Similarly, INSERT statements populate the Queries table with specific ID-Year pairs that need to be referenced or looked up.
- These inserts set up the data required for subsequent financial analysis and stakeholder queries.

### 6. Generate the NPV Calculation Report: (Hard Level Query)

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

- The query begins with the Queries table to ensure every stakeholder request is included in the results.
- It uses a LEFT JOIN to find matching NPV values from the Year\_tbl table by comparing both ID and YEAR columns.
- If there is no matching NPV data for a request, the ISNULL() function replaces the resulting NULL with zero.

## **CODE:**

--Using AIT\_1A Database

USE AIT\_1A

--Medium Level Problem

--Create table Tbl\_EmpR to store Employee Relations

Create Table Tbl\_EmpR

(

EmpID INT PRIMARY KEY,

Ename VARCHAR(50) NOT NULL,

Department VARCHAR(50) NOT NULL,

ManagerID INT NULL

FOREIGN KEY(ManagerID) REFERENCES Tbl\_EmpR(EmpID)

)

--Constraint added

ALTER TABLE Tbl\_EmpR

ADD CONSTRAINT FK\_EMPLOYEE FOREIGN KEY(ManagerID) REFERENCES  
Tbl\_EmpR(EmpID)

--Insert Values to Tbl\_EmpR

INSERT INTO Tbl\_EmpR(EmpID, EName, 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)

--Perform SELF JOIN(LEFT/RIGHT)

SELECT E1.Ename AS [EMPLOYEE\_NAME], E2.Ename AS

[MANAGER\_NAME], E1.Department AS [DEPARTMENT], E1.ManagerID AS  
[MANAGER\_ID]

FROM Tbl\_EmpR AS E1

LEFT OUTER JOIN

Tbl\_EmpR AS E2

ON E1.ManagerID=E2.EmpID

--Hard Level Problem

--Create Tbl\_Year (holds actual NPV values)

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

--Create Queries table (requested values)

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

--Insert data into Tbl\_Year

```
INSERT INTO Tbl_Year (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

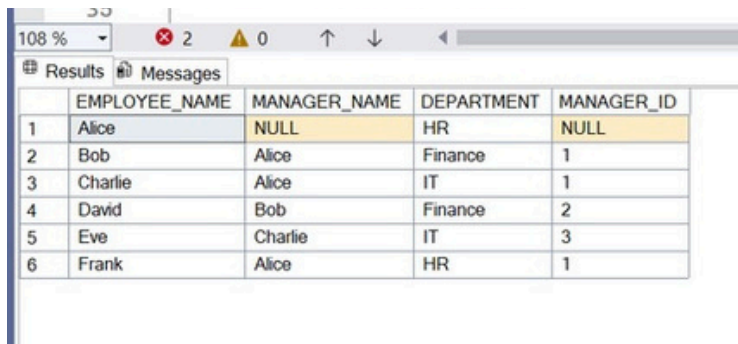
```
INSERT INTO Queries (ID, YEAR)  
VALUES  
  (1, 2019),  
  (2, 2008),  
  (3, 2009),  
  (7, 2018),  
  (7, 2019),  
  (7, 2020),  
  (13, 2019);
```

--Perform LEFT OUTER JOIN to get NPV from Tbl\_Year based on ID and YEAR

```
SELECT  
q.ID,  
q.YEAR,  
ISNULL (y.NPV,0) AS NPV  
FROM  
  Queries q  
LEFT OUTER JOIN  
  Tbl_Year AS y  
ON  
  q.ID = y.ID AND q.YEAR = y.YEAR;
```

## OUTPUT:

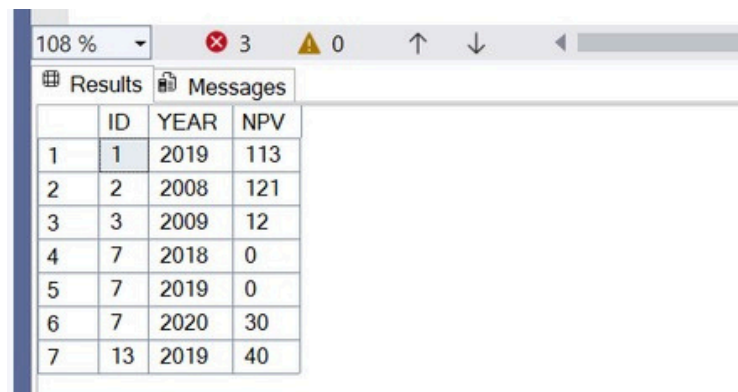
- MEDIUM LEVEL PROBLEM



A screenshot of a database query result window. The window has a toolbar at the top with a zoom level of 108%, 2 errors, 0 warnings, and navigation arrows. Below the toolbar are tabs for 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with 5 columns: EMPLOYEE\_NAME, MANAGER\_NAME, DEPARTMENT, and MANAGER\_ID. The table contains 6 rows of data. The first row shows Alice as an employee with a NULL manager, in the HR department. The subsequent rows show a hierarchy where Bob and Charlie are managed by Alice, David by Bob, Eve by Charlie, and Frank by Alice.

	EMPLOYEE_NAME	MANAGER_NAME	DEPARTMENT	MANAGER_ID
1	Alice	NULL	HR	NULL
2	Bob	Alice	Finance	1
3	Charlie	Alice	IT	1
4	David	Bob	Finance	2
5	Eve	Charlie	IT	3
6	Frank	Alice	HR	1

- HARD LEVEL PROBLEM



A screenshot of a database query result window. The window has a toolbar at the top with a zoom level of 108%, 3 errors, 0 warnings, and navigation arrows. Below the toolbar are tabs for 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with 4 columns: ID, YEAR, and NPV. The table contains 7 rows of data. The first row shows ID 1 for the year 2019 with an NPV of 113. The subsequent rows show IDs 2, 3, 7, 7, 7, and 13 for the years 2008, 2009, 2018, 2019, 2020, and 2019 respectively, with NPV values of 121, 12, 0, 0, 30, and 40.

	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

## LEARNING OUTCOMES:

1. Understand Key Concepts in Database Design
  - Explain the purpose of PRIMARY KEY and FOREIGN KEY constraints and their role in enforcing data integrity.
  - Recognize how self-referencing FOREIGN KEYS can model hierarchical relationships within a single table.
2. Write and Interpret SQL Joins for Complex Data Retrieval
  - Use SELF JOIN to query hierarchical data, such as employees and their managers, from the same table.
  - Apply LEFT JOIN to ensure complete datasets are retrieved, including records without matching entries in related tables.
3. Handle Missing or Incomplete Data Effectively
  - Identify when NULL values arise in query results, especially from outer joins.
  - Use functions like ISNULL() to replace NULLs with default values for clearer, more reliable reporting.
4. Design and Implement SQL Table Structures for Real-World Scenarios
  - Create tables with appropriate constraints to represent organizational hierarchies and stakeholder queries.
  - Develop Queries that Combine Data Modeling and Data Cleaning