University Institute of Engineering

Department of Computer Science & Engineering

Experiment:2

Date of Experiment: 28-07-2025

MEDIUM

1. Aim of the practical:

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:

1. The employee's name and department
2. Their manager's name and department (if applicable)

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

2. Theory:

In SQL, a self-join is used when a table needs to be joined with itself. This is useful when data in a single table is hierarchically related — such as employees reporting to managers, where both roles are stored in the same table.
By assigning aliases to the same table, we can compare records within the same table. In this experiment, we use a self-join to match each employee with their corresponding manager using the Manager_ID field.
The LEFT JOIN ensures all employees are included in the result, even those who do not report to any manager (e.g., top-level executives or CEO).

3. Queries:

```
/* 1. Create the database */

CREATE DATABASE EmployeeDB;
USE EmployeeDB;

/*2. Create the Authors table */
 CREATE TABLE TBL_Employee (
        EmpID INT PRIMARY KEY,
        EmpName VARCHAR(50) NOT NULL,
        Department VARCHAR(50) NOT NULL,
        ManagerID INT NULL
 );

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

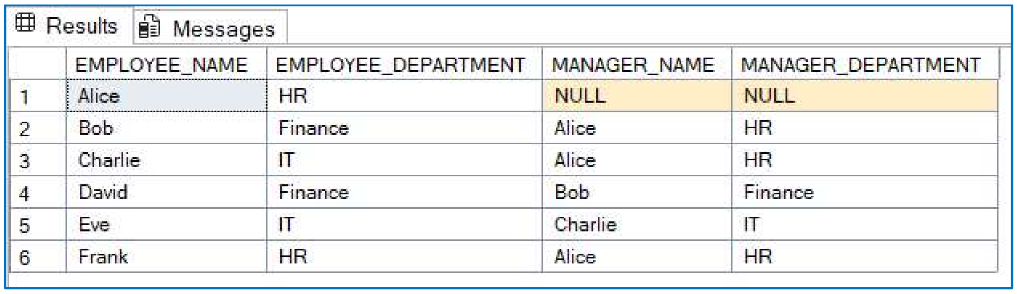
```
/* 3. Insert authors (from 5 different countries) */

 INSERT INTO TBL_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);

/* 4. Self Join */

                SELECT E1.EmpName AS [EMPLOYEE_NAME],
                E1.Department AS [EMPLOYEE_DEPARTMENT],
                E2.EmpName AS [MANAGER_NAME],
                E2.Department AS [MANAGER_DEPARTMENT]
                FROM TBL_Employee AS E1
                Left outer JOIN
                TBL_Employee AS E2
                ON E1.ManagerID = E2.EmpID;
```

5. Output:



	EMPLOYEE_NAME	EMPLOYEE_DEPARTMENT	MANAGER_NAME	MANAGER_DEPARTMENT
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

Result:

The self-join correctly produced a mapping between employees and their managers, revealing internal reporting lines. This hierarchical view supports HR in managing departments and understanding reporting structures.

Learning outcomes (What I have learnt):

1. I have learnt how to perform a self-join to relate data within a single table.
2. I have learnt to model hierarchical relationships using foreign keys.
3. I have learnt the difference between INNER JOIN and LEFT JOIN in preserving non-matching records.
4. I have learnt to visualize reporting structures using SQL.
5. I have learnt how to handle and interpret NULL values in joins when managers are not assigned.

University Institute of Engineering

Department of Computer Science & Engineering

Experiment:2

Date of Experiment: 28-07-2025
Hard

1. Aim of the practical:

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 ordered 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. Theory:

This experiment applies key SQL concepts such as LEFT JOIN, ISNULL() (or COALESCE()), and the ORDER BY clause. The LEFT JOIN is used to combine records from two tables while preserving all entries from the left table—even when matching data is not available in the right table. This is especially useful when dealing with incomplete or inconsistent data. The ISNULL() or COALESCE() function is used to handle missing values by substituting them with default values, which is a common data imputation technique in data engineering. In this case, missing Net Present Values (NPVs) are replaced with zero to maintain a consistent financial report. Additionally, the ORDER BY clause ensures the final output is sorted by ID and YEAR, making the result easier to interpret and analyze. These concepts are widely used in financial data systems and are essential for building robust, real-world SQL queries that handle incomplete datasets gracefully.

3. Queries:

```
/* 1. Create the database */
        CREATE DATABASE NPVDb;
        USE NPVDb;

/* 2. Create the TBL_YEAR table */
        CREATE TABLE TBL_YEAR(
                ID INT,
                YEAR INT,
                NPV INT
         );
```

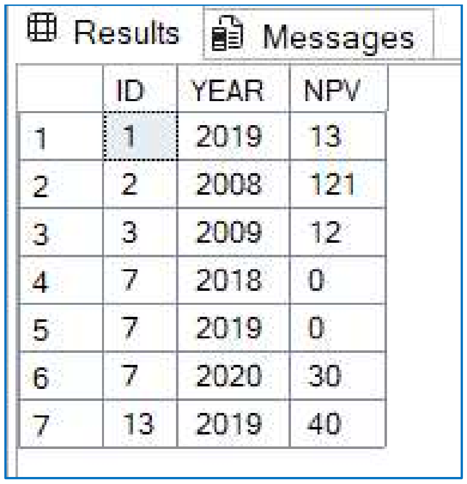
```
/* 3. Insert TBL_YEAR Data */
      INSERT INTO TBL_YEAR(ID,YEAR,NPV) VALUES
      (1,2018,100), (7,2020,30), (13,2019,40), (1,2019,13),
      (2,2008,121), (3,2009,12), (11,2020,99), (7,2019,0);

/* 4. Create the TBL_QUERIES table */
     CREATE TABLE TBL_QUERIES(
           ID INT,
           YEAR INT
     );

/* 5. Insert TBL_QUERIES Data */
       INSERT INTO TBL_QUERIES( ID,YEAR) VALUES
       (1,2019), (2,2008), (3,2009), (7,2018), (7,2019), (7,2020), (13,2019);

/* 6. Retrieve NPV values for given (ID, YEAR) pairs from TBL_QUERIES */
        SELECT
        Q.ID,
        Q.YEAR,
        ISNULL(Y.NPV,0) AS[NPV]
        FROM TBL_QUERIES AS Q
        LEFT OUTER JOIN TBL_YEAR AS Y
        ON Q.ID = Y.ID AND Y.YEAR = Q.YEAR
```

4. Output:



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

Result:

The query successfully matched all requested ID-YEAR pairs with available NPV data. For missing combinations, it returned 0 as the fallback, ensuring a complete and clean financial report sorted by ID and YEAR.

Learning outcomes (What I have learnt):

1. I have learnt to perform LEFT JOIN operations for partial data matching.
2. I have learnt how to handle null results using ISNULL() in SQL.
3. 3. I have learnt to structure queries to meet real-world financial reporting needs.
4. 4. I have learnt the importance of default values in maintaining consistent datasets.
5. 5. I have learnt how to order results for readability and reporting.