Experiment 2

Student Name: Rachit Kumar
UID: 23BCS11597
Branch: BE CSE
Section: 23BCS_KRG-2/A
Subject Name: Advanced Database. and Management System
Subject Code: 23CSP-333
Date :24/07/2025
Semester: 5th

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 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
• The manager's name and department (if applicable)
• This will help the HR department visualize the internal reporting hierarchy.

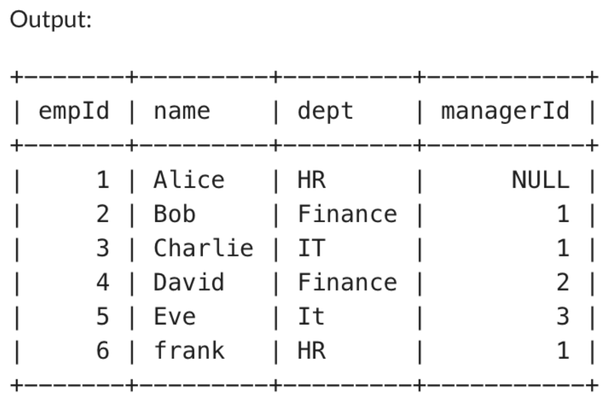
2. Tools Used: SQL Server (One compiler)

3. Code:

```
CREATE TABLE EMPLOYEE (
  empId int primary KEY,
  name varchar(15),
  dept varchar(10),
  managerId int
);

INSERT INTO EMPLOYEE(empId,name,dept,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 * FROM EMPLOYEE;
```

```
ALTER TABLE EMPLOYEE
ADD constraint FK_EMPLoYEE FOREIGN KEY(managerId)
references EMPLOYEE(empId);

select E1.name as [EMPLOYEE_Name], E1.dept as [EMPLOYEE_DEPARTMENT],
E2.name as [Manager_Name], E2.dept as [MANAGER_DEPARTMENT]
from EMPLOYEE as E1
Left Outer join
EMPLOYEE as E2
on E1.managerId = E2.empId;
```

Output:



Output:

```
+--------+---------+---------+-----------+
| empId | name    | dept    | managerId |
+--------+---------+---------+-----------+
|      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. Learning Outcomes

- **Design a table with self-referencing foreign keys** to represent hierarchical data (e.g., employee–manager relationships).

- **Enforce referential integrity** within a single table using foreign key constraints.

- **Insert and manage hierarchical records** where some rows act as parents (managers) and others as children (employees).

- **Apply self-joins** to query relationships between rows in the same table.

- **Differentiate between INNER JOIN and LEFT OUTER JOIN** when handling missing relationships (e.g., top-level managers with no managers).

- **Present meaningful hierarchical reports** showing both employee and manager details.

- Gain practical experience in modeling **organizational structures** using SQL.