DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



**Assignment -2.1**

**Name:** Shubh Rai **UID:** 23BCS12916

**Subject:** DBMS **Semester**: 5th

**Section:** KRG - 3B **Date of performance:**27th July 2025 **Branch: BE -** CSE

**1.1 Medium problem:**

**1: AIM:**

-- Write an SQL query using a self join to:

-- Select the employee's name, aliased as employee\_name

-- Select the manager's name, aliased as manager\_name

-- Select the employee's department, aliased as employee\_department

-- Select the manager's department, aliased as manager\_department

-- Include all employees, even if they do not have a manager

-- Use a LEFT OUTER JOIN between the employee table and itself, with the correct join condition.

**2:CODE**

create table employee( e\_id int primary key, name varchar(10), department varchar(20), manager\_id int, foreign key (manager\_id) references employee (e\_id),

);

insert into employee(e\_id, name, department, manager\_id) 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 e1.name as [employee\_ Name] ,e2.name as[manager\_name]

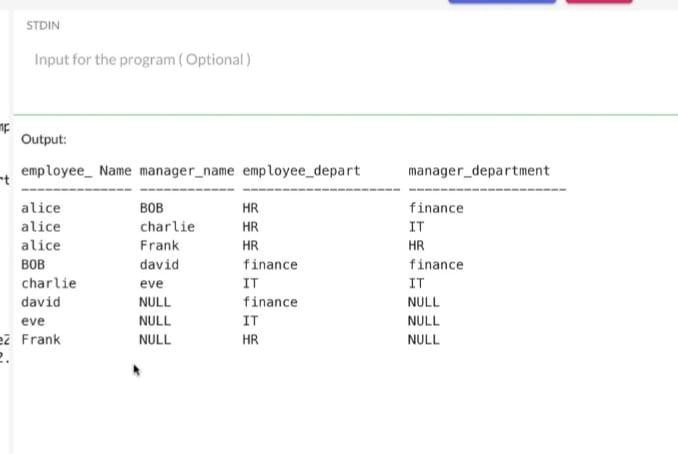
,

e1.department as[employee\_depart] , e2.department as[manager\_department] from employee as e1

left outer join employee e2 on

e1.e\_id= e2.manager\_id;

**3: OUTPUT:**



**4: LEARNING OUTCOME:**

* Understand how to create **self-referencing foreign keys** to represent hierarchical relationships (e.g., employee–manager).

* Learn to use **table aliases** when joining a table with itself.

* Apply **LEFT OUTER JOIN** to retrieve all records from one table even if there's no matching record in the joined table.

* Retrieve **employee-manager relationships** along with their respective departments.

* Understand how to model and query **organizational structures** using SQL.