Hierarchical Query

1. Basic Hierarchical Query:

Write a query to retrieve all employees and their levels in the company hierarchy using a recursive CTE. Display each employee's name, employee_id, manager_id, and their level in the hierarchy.

```
mysql> with recursive hierarchy as (
    ->     select
    ->         employee_id, name, manager_id, 0 as level
    ->     from employees
    ->     where manager_id is null
    ->     union all
    ->     select
    ->         e.employee_id, e.name, e.manager_id, h.level + 1
    ->     from employees e
    ->     inner join hierarchy h
    ->     on e.manager_id = h.employee_id
    -> )
    -> select employee_id, name, manager_id, level
    -> from hierarchy;
+-------------+---------+------------+-------+
| employee_id | name    | manager_id | level |
+-------------+---------+------------+-------+
|           1 | Alice   |       NULL |     0 |
|           2 | Bob     |          1 |     1 |
|           3 | Charlie |          1 |     1 |
|           4 | Diana   |          2 |     2 |
|           5 | Eve     |          2 |     2 |
|           6 | Frank   |          3 |     2 |
|           7 | Grace   |          3 |     2 |
+-------------+---------+------------+-------+
7 rows in set (0.01 sec)
```

2. Identify Top-Level Managers: Modify the base case in the CTE to select only managers with no supervisors, such as the CEO or department heads. List their names and employee_ids.

```
mysql> with recursive hierarchy as (
    ->     select
    ->         employee_id, name, manager_id, 0 as level, name as path
    ->     from employees
    ->     where manager_id is null
    ->     union all
    ->     select
    ->         e.employee_id, e.name, e.manager_id, h.level + 1,
    ->         concat(h.path, ' -> ', e.name)
    ->     from employees e
    ->     inner join hierarchy h
    ->     on e.manager_id = h.employee_id
    -> )
    -> select
    ->     repeat(' ', level * 2) || name as hierarchy_tree,
    ->     employee_id, manager_id, level
    -> from hierarchy;
+----------------+-------------+------------+-------+
| hierarchy_tree | employee_id | manager_id | level |
+----------------+-------------+------------+-------+
|              0 |           1 |       NULL |     0 |
|              0 |           2 |          1 |     1 |
|              0 |           3 |          1 |     1 |
|              0 |           4 |          2 |     2 |
|              0 |           5 |          2 |     2 |
|              0 |           6 |          3 |     2 |
|              0 |           7 |          3 |     2 |
+----------------+-------------+------------+-------+
7 rows in set, 8 warnings (0.00 sec)
```

3. Count Employees per Level: Extend the recursive query to count the number of employees at each level of the hierarchy.

```
mysql> with recursive hierarchy as (
    ->     select
    ->         employee_id, name, manager_id, 0 as level
    ->     from employees
    ->     where manager_id is null
    ->     union all
    ->     select
    ->         e.employee_id, e.name, e.manager_id, h.level + 1
    ->     from employees e
    ->     inner join hierarchy h
    ->     on e.manager_id = h.employee_id
    -> )
    -> select level, count(*) as employee_count
    -> from hierarchy
    -> group by level;
+-------+----------------+
| level | employee_count |
+-------+----------------+
|     0 |              1 |
|     1 |              2 |
|     2 |              4 |
+-------+----------------+
3 rows in set (0.00 sec)
```

4. Find All Subordinates for a Specific Employee: Write a recursive query to find all employees who report (directly or indirectly) to a specific manager, given their employee_id.

```
mysql> with recursive subordinates as (
    ->     select
    ->         employee_id, name, manager_id
    ->     from employees
    ->     where employee_id = 2 -- replace with specific manager_id
    ->     union all
    ->     select
    ->         e.employee_id, e.name, e.manager_id
    ->     from employees e
    ->     inner join subordinates s
    ->     on e.manager_id = s.employee_id
    -> )
    -> select employee_id, name, manager_id
    -> from subordinates;
+-------------+-------+------------+
| employee_id | name  | manager_id |
+-------------+-------+------------+
|           2 | Bob   |          1 |
|           4 | Diana |          2 |
|           5 | Eve   |          2 |
+-------------+-------+------------+
3 rows in set (0.00 sec)
```

5. List Employees in Descending Order of Levels: Modify the query to display employees from the deepest level of the hierarchy to the top level. Explain how ordering by level DESC changes the output.

```
mysql> with recursive hierarchy as (
    ->     select
    ->         employee_id, name, manager_id, 0 as level
    ->     from employees
    ->     where manager_id is null
    ->     union all
    ->     select
    ->         e.employee_id, e.name, e.manager_id, h.level + 1
    ->     from employees e
    ->     inner join hierarchy h
    ->     on e.manager_id = h.employee_id
    -> )
    -> select employee_id, name, manager_id, level
    -> from hierarchy
    -> order by level desc;
+-------------+---------+------------+-------+
| employee_id | name    | manager_id | level |
+-------------+---------+------------+-------+
|           4 | Diana   |          2 |     2 |
|           5 | Eve     |          2 |     2 |
|           6 | Frank   |          3 |     2 |
|           7 | Grace   |          3 |     2 |
|           2 | Bob     |          1 |     1 |
|           3 | Charlie |          1 |     1 |
|           1 | Alice   |       NULL |     0 |
+-------------+---------+------------+-------+
7 rows in set (0.00 sec)
```

6. Identify Orphans (Employees Without Managers): Write a query to find employees who do not have a manager listed in the table (i.e., manager_id points to an invalid or missing employee_id).

```
mysql> select e.employee_id, e.name
    -> from employees e
    -> left join employees m
    -> on e.manager_id = m.employee_id
    -> where e.manager_id is not null and m.employee_id is null;
Empty set (0.00 sec)
```

7. Calculate Depth of Hierarchy: Using the recursive CTE, determine the maximum depth of the company hierarchy by identifying the highest level reached

```
mysql> with recursive hierarchy as (
    ->     select
    ->         employee_id, name, manager_id, 0 as level
    ->     from employees
    ->     where manager_id is null
    ->     union all
    ->     select
    ->         e.employee_id, e.name, e.manager_id, h.level + 1
    ->     from employees e
    ->     inner join hierarchy h
    ->     on e.manager_id = h.employee_id
    -> )
    -> select max(level) as max_depth
    -> from hierarchy;
+-----------+
| max_depth |
+-----------+
|         2 |
+-----------+
1 row in set (0.01 sec)
```

8. Add Department Information to Hierarchy: Assume there's a departments table with department_id and department_name. Modify the hierarchical query to include each employee's department name by joining the employees table with the departments table.

```
mysql> with recursive hierarchy as (
    ->     select
    ->         e.employee_id, e.name, e.manager_id, 0 as level, d.department_name
    ->     from employees e
    ->     inner join departments d on e.department_id = d.department_id
    ->     where e.manager_id is null
    ->     union all
    ->     select
    ->         e.employee_id, e.name, e.manager_id, h.level + 1, d.department_name
    ->     from employees e
    ->     inner join hierarchy h
    ->     on e.manager_id = h.employee_id
    ->     inner join departments d on e.department_id = d.department_id
    -> )
    -> select employee_id, name, manager_id, level, department_name
    -> from hierarchy;
+-------------+---------+------------+-------+-----------------+
| employee_id | name    | manager_id | level | department_name |
+-------------+---------+------------+-------+-----------------+
|           1 | Alice   |       NULL |     0 | Executive       |
|           2 | Bob     |          1 |     1 | Engineering     |
|           3 | Charlie |          1 |     1 | Engineering     |
|           4 | Diana   |          2 |     2 | HR              |
|           5 | Eve     |          2 |     2 | HR              |
|           6 | Frank   |          3 |     2 | Finance         |
|           7 | Grace   |          3 |     2 | Finance         |
+-------------+---------+------------+-------+-----------------+
7 rows in set (0.00 sec)
```

9. Identify Common Ancestor for Two Employees: Write a query to find the lowest common ancestor (common manager) for two employees, given their employee_ids.

```
mysql> with recursive ancestors as (
    ->     select employee_id, manager_id
    ->     from employees
    ->     where employee_id in (6, 7) -- replace with employee ids
    ->     union all
    ->     select e.employee_id, e.manager_id
    ->     from employees e
    ->     inner join ancestors a
    ->     on e.employee_id = a.manager_id
    -> )
    -> select manager_id as common_ancestor
    -> from ancestors
    -> group by manager_id
    -> having count(distinct employee_id) = 2;
+-----------------+
| common_ancestor |
+-----------------+
|               3 |
+-----------------+
1 row in set (0.00 sec)
```