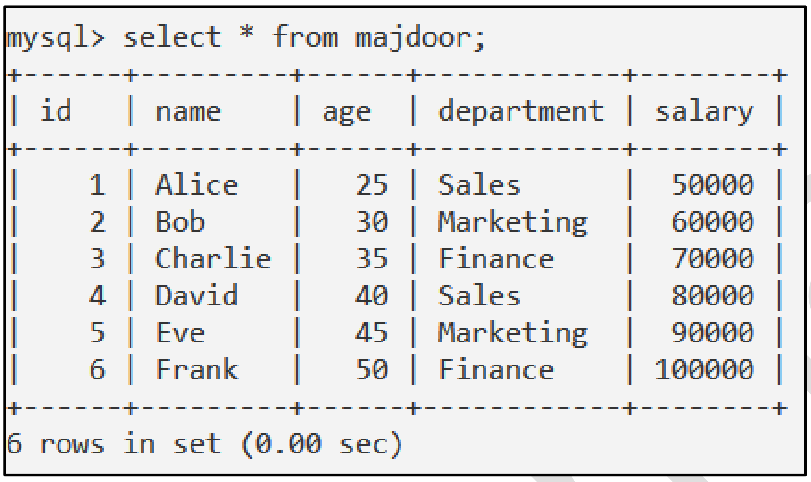
# “Exp 8: Query based on operators and joins • Simple and nested query.”

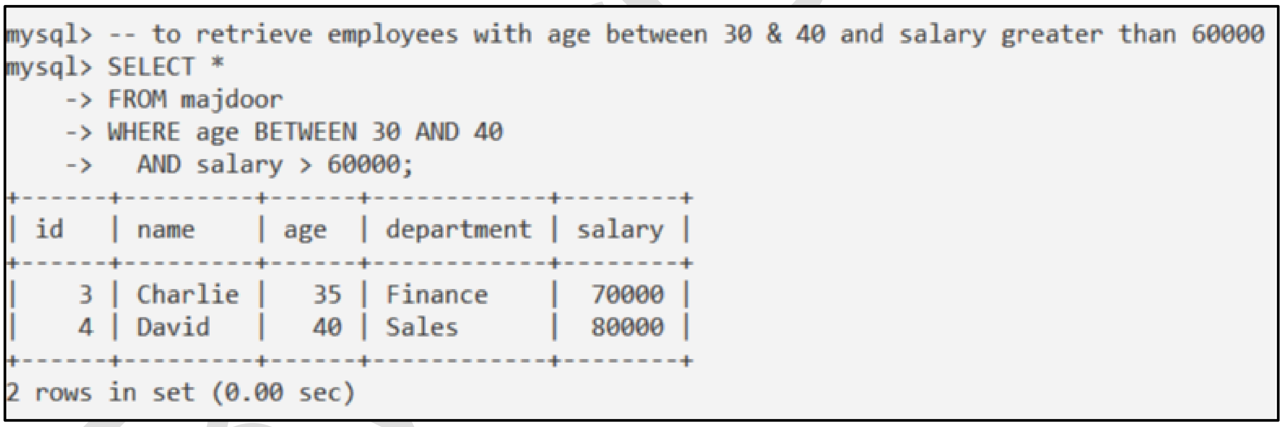
***] Reference Table (for operators) : -**



```
mysql> select * from majdoor;
+------+---------+------+------------+--------+
| id   | name    | age  | department | salary |
+------+---------+------+------------+--------+
|    1 | Alice   |   25 | Sales      |  50000 |
|    2 | Bob     |   30 | Marketing  |  60000 |
|    3 | Charlie |   35 | Finance    |  70000 |
|    4 | David   |   40 | Sales      |  80000 |
|    5 | Eve     |   45 | Marketing  |  90000 |
|    6 | Frank   |   50 | Finance    | 100000 |
+------+---------+------+------------+--------+
6 rows in set (0.00 sec)
```

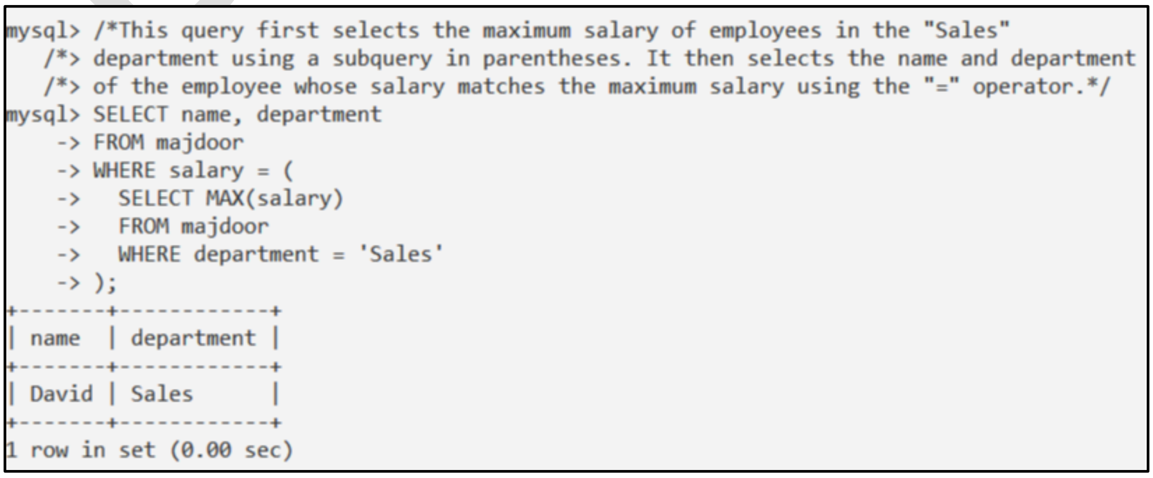
**1] Operator based queries : -**

**a] Simple query : -**



```
mysql> -- to retrieve employees with age between 30 & 40 and salary greater than 60000
mysql> SELECT *
    -> FROM majdoor
    -> WHERE age BETWEEN 30 AND 40
    ->   AND salary > 60000;
+------+---------+------+------------+--------+
| id   | name    | age  | department | salary |
+------+---------+------+------------+--------+
|    3 | Charlie |   35 | Finance    |  70000 |
|    4 | David   |   40 | Sales      |  80000 |
+------+---------+------+------------+--------+
2 rows in set (0.00 sec)
```

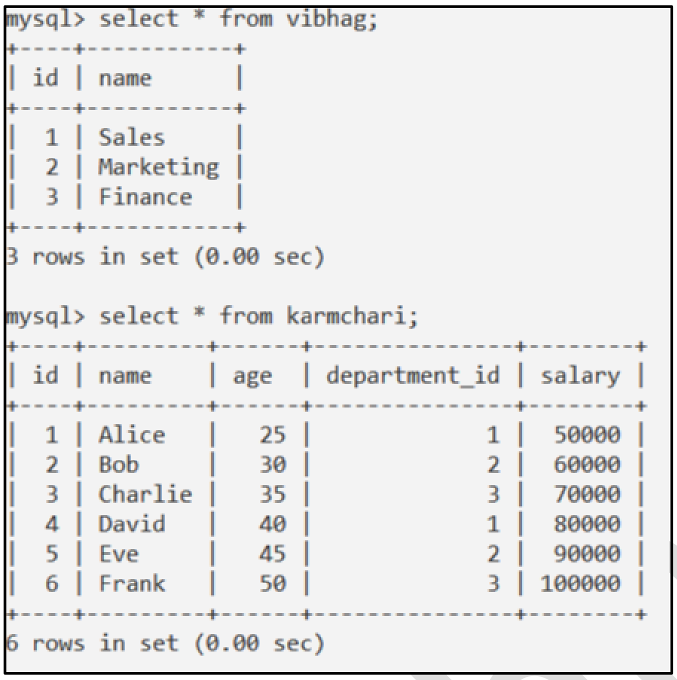
**b] Nested query : -**



```
mysql> /*This query first selects the maximum salary of employees in the "Sales"
   /*> department using a subquery in parentheses. It then selects the name and department
   /*> of the employee whose salary matches the maximum salary using the "=" operator.*/
mysql> SELECT name, department
    -> FROM majdoor
    -> WHERE salary = (
    ->   SELECT MAX(salary)
    ->   FROM majdoor
    ->   WHERE department = 'Sales'
    -> );
+-------+------------+
| name  | department |
+-------+------------+
| David | Sales      |
+-------+------------+
1 row in set (0.00 sec)
```

P.T.O →

***] Reference table (for joins) : -**



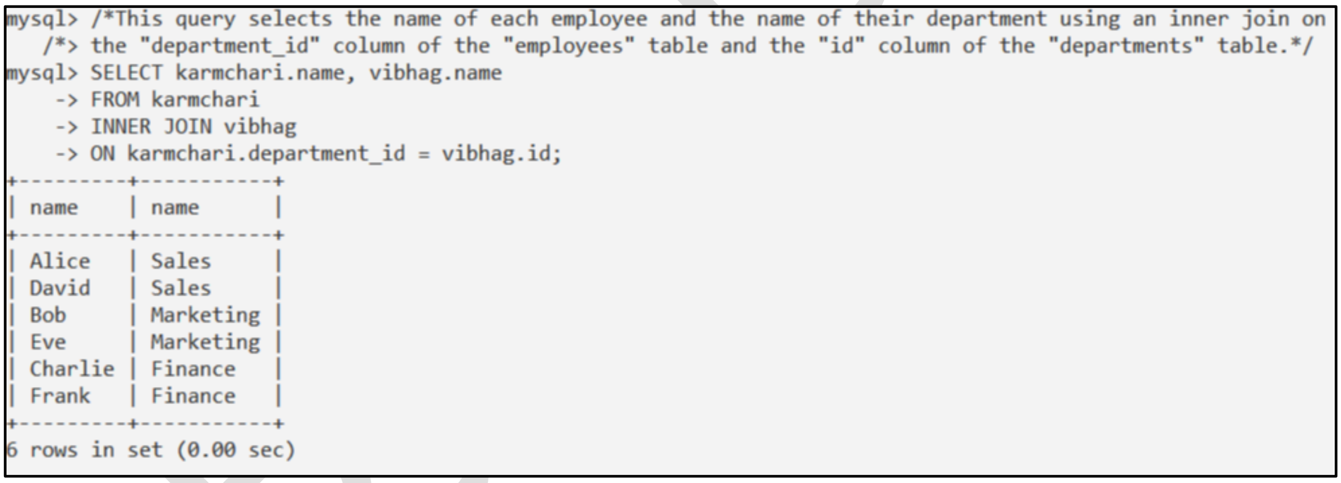
```
mysql> select * from vibhag;
+----+-----------+
| id | name      |
+----+-----------+
|  1 | Sales     |
|  2 | Marketing |
|  3 | Finance   |
+----+-----------+
3 rows in set (0.00 sec)

mysql> select * from karmchari;
+----+---------+------+---------------+--------+
| id | name    | age  | department_id | salary |
+----+---------+------+---------------+--------+
|  1 | Alice   |   25 |             1 |  50000 |
|  2 | Bob     |   30 |             2 |  60000 |
|  3 | Charlie |   35 |             3 |  70000 |
|  4 | David   |   40 |             1 |  80000 |
|  5 | Eve     |   45 |             2 |  90000 |
|  6 | Frank   |   50 |             3 | 100000 |
+----+---------+------+---------------+--------+
6 rows in set (0.00 sec)
```

**2] Joins based queries : -**

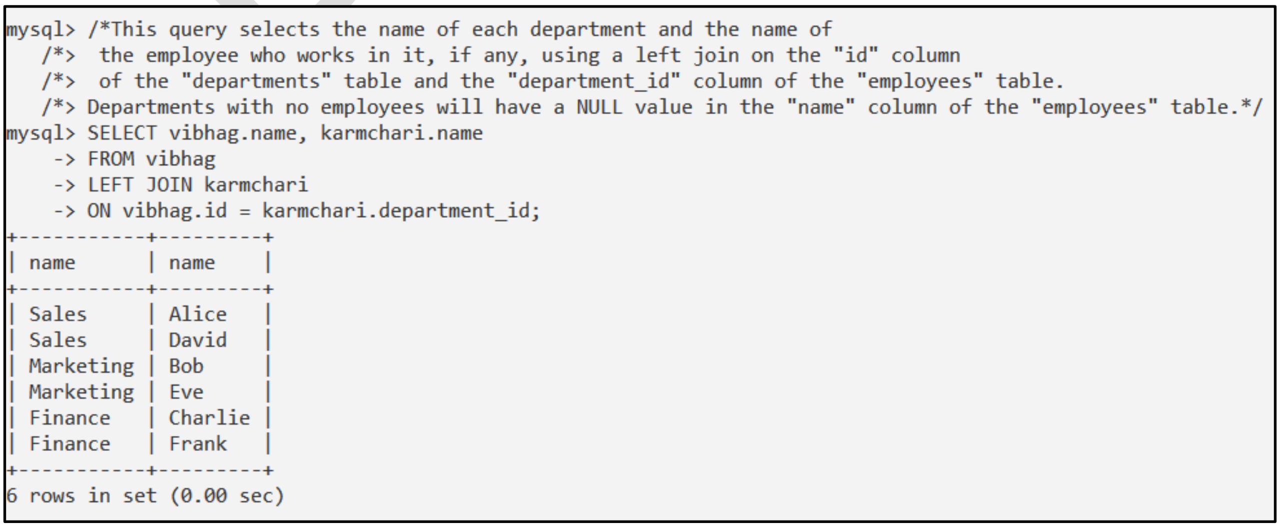
**a] Simple query : -**

**I] Inner Join : -**



```
mysql> /*This query selects the name of each employee and the name of their department using an inner join on
   /*> the "department_id" column of the "employees" table and the "id" column of the "departments" table.*/
mysql> SELECT karmchari.name, vibhag.name
    -> FROM karmchari
    -> INNER JOIN vibhag
    -> ON karmchari.department_id = vibhag.id;
+---------+-----------+
| name    | name      |
+---------+-----------+
| Alice   | Sales     |
| David   | Sales     |
| Bob     | Marketing |
| Eve     | Marketing |
| Charlie | Finance   |
| Frank   | Finance   |
+---------+-----------+
6 rows in set (0.00 sec)
```

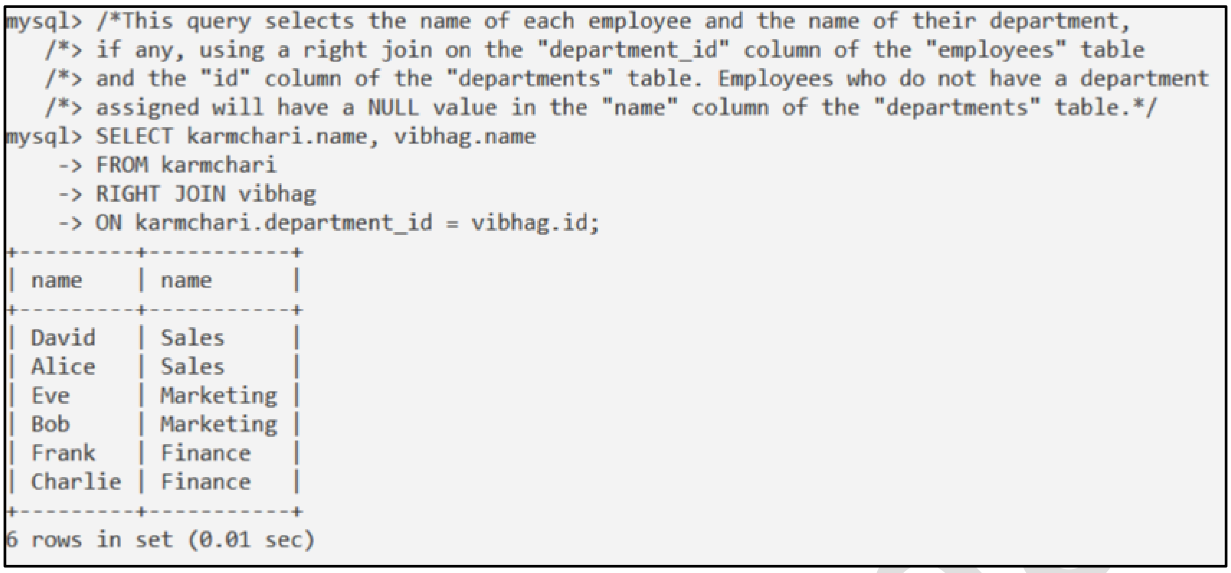
**II] Left Join : -**



```
mysql> /*This query selects the name of each department and the name of
   /*>  the employee who works in it, if any, using a left join on the "id" column
   /*>  of the "departments" table and the "department_id" column of the "employees" table.
   /*> Departments with no employees will have a NULL value in the "name" column of the "employees" table.*/
mysql> SELECT vibhag.name, karmchari.name
    -> FROM vibhag
    -> LEFT JOIN karmchari
    -> ON vibhag.id = karmchari.department_id;
+-----------+---------+
| name      | name    |
+-----------+---------+
| Sales     | Alice   |
| Sales     | David   |
| Marketing | Bob     |
| Marketing | Eve     |
| Finance   | Charlie |
| Finance   | Frank   |
+-----------+---------+
6 rows in set (0.00 sec)
```

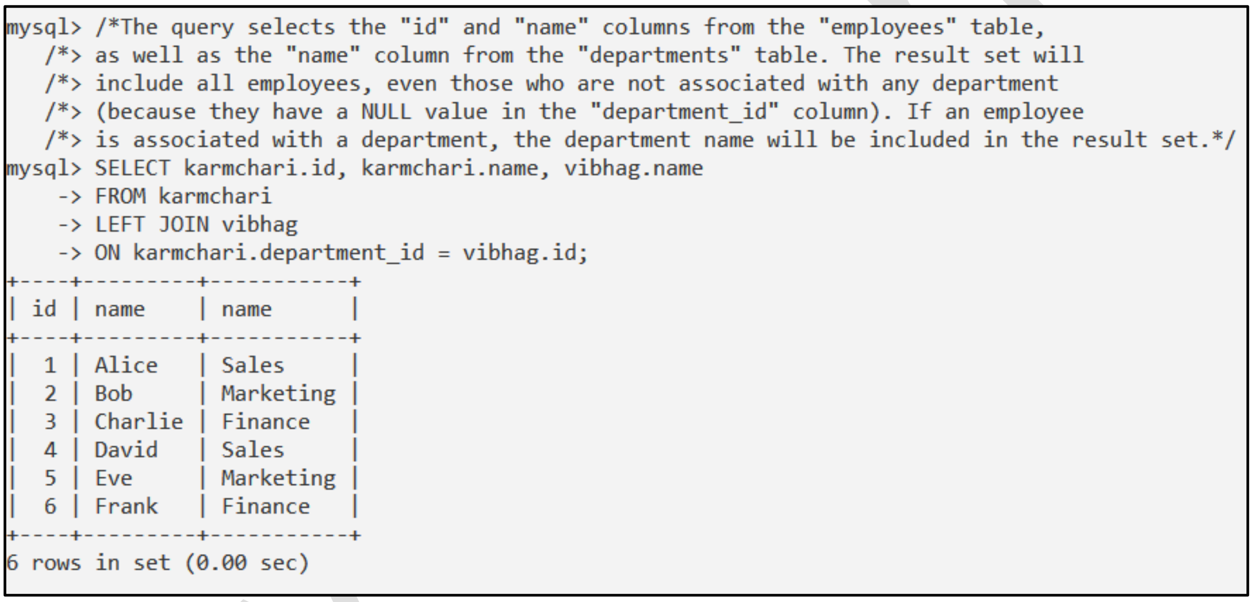
**P.T.O →**

## III] Right Join : -



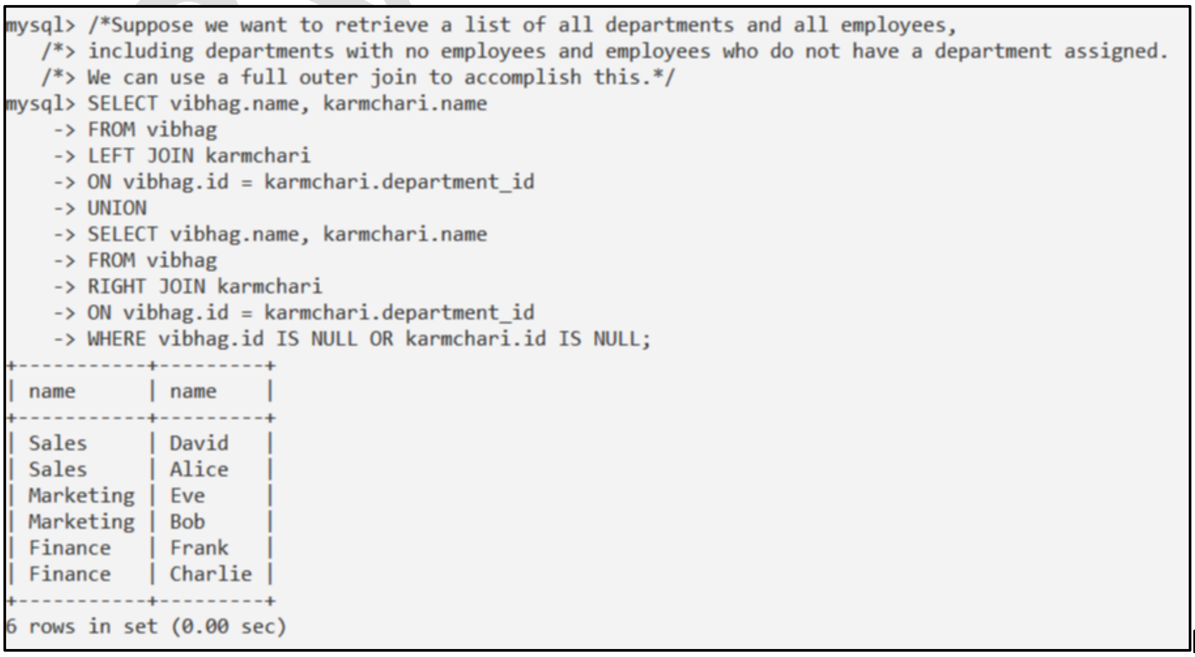
```
mysql> /*This query selects the name of each employee and the name of their department,
   /*> if any, using a right join on the "department_id" column of the "employees" table
   /*> and the "id" column of the "departments" table. Employees who do not have a department
   /*> assigned will have a NULL value in the "name" column of the "departments" table.*/
mysql> SELECT karmchari.name, vibhag.name
    -> FROM karmchari
    -> RIGHT JOIN vibhag
    -> ON karmchari.department_id = vibhag.id;
+---------+-----------+
| name    | name      |
+---------+-----------+
| David   | Sales     |
| Alice   | Sales     |
| Eve     | Marketing |
| Bob     | Marketing |
| Frank   | Finance   |
| Charlie | Finance   |
+---------+-----------+
6 rows in set (0.01 sec)
```

## IV] Left-outer join : -



```
mysql> /*The query selects the "id" and "name" columns from the "employees" table,
   /*> as well as the "name" column from the "departments" table. The result set will
   /*> include all employees, even those who are not associated with any department
   /*> (because they have a NULL value in the "department_id" column). If an employee
   /*> is associated with a department, the department name will be included in the result set.*/
mysql> SELECT karmchari.id, karmchari.name, vibhag.name
    -> FROM karmchari
    -> LEFT JOIN vibhag
    -> ON karmchari.department_id = vibhag.id;
+----+---------+-----------+
| id | name    | name      |
+----+---------+-----------+
|  1 | Alice   | Sales     |
|  2 | Bob     | Marketing |
|  3 | Charlie | Finance   |
|  4 | David   | Sales     |
|  5 | Eve     | Marketing |
|  6 | Frank   | Finance   |
+----+---------+-----------+
6 rows in set (0.00 sec)
```

## V] Right-outer join/(Full outer join) (there is no full outer join in MySQL) : -

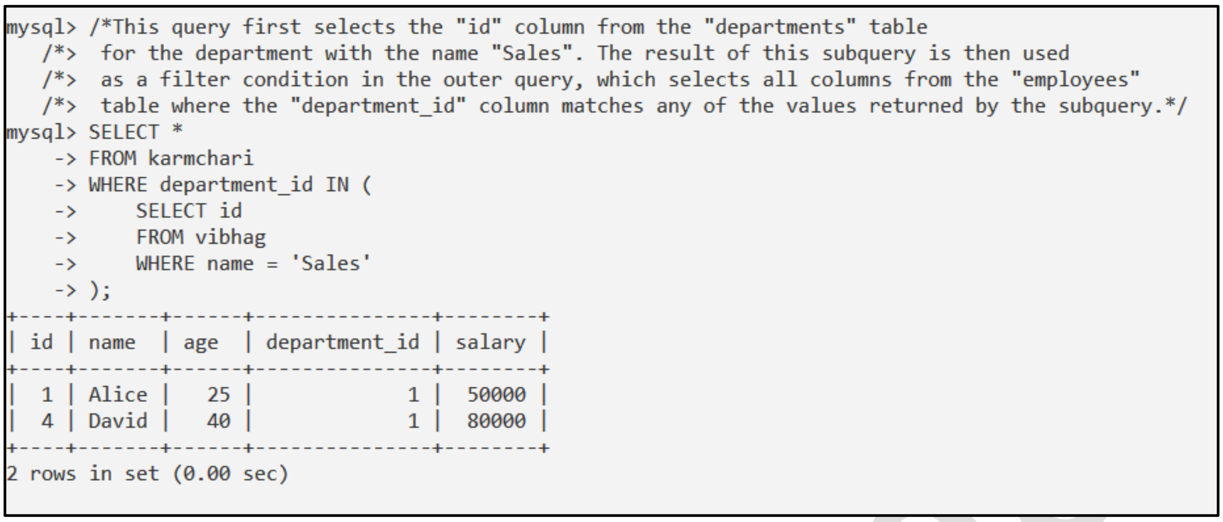


```
mysql> /*Suppose we want to retrieve a list of all departments and all employees,
   /*> including departments with no employees and employees who do not have a department assigned.
   /*> We can use a full outer join to accomplish this.*/
mysql> SELECT vibhag.name, karmchari.name
    -> FROM vibhag
    -> LEFT JOIN karmchari
    -> ON vibhag.id = karmchari.department_id
    -> UNION
    -> SELECT vibhag.name, karmchari.name
    -> FROM vibhag
    -> RIGHT JOIN karmchari
    -> ON vibhag.id = karmchari.department_id
    -> WHERE vibhag.id IS NULL OR karmchari.id IS NULL;
+-----------+---------+
| name      | name    |
+-----------+---------+
| Sales     | David   |
| Sales     | Alice   |
| Marketing | Eve     |
| Marketing | Bob     |
| Finance   | Frank   |
| Finance   | Charlie |
+-----------+---------+
6 rows in set (0.00 sec)
```

P.T.O →

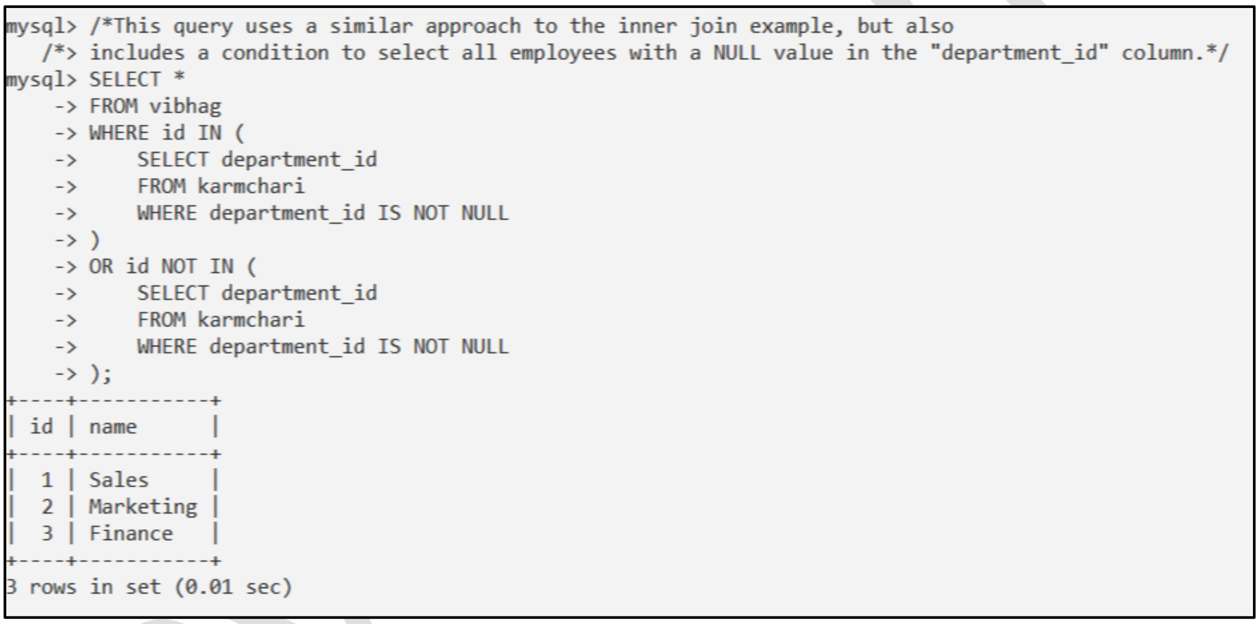
## b] Nested query : -

### a] INNER JOIN : -



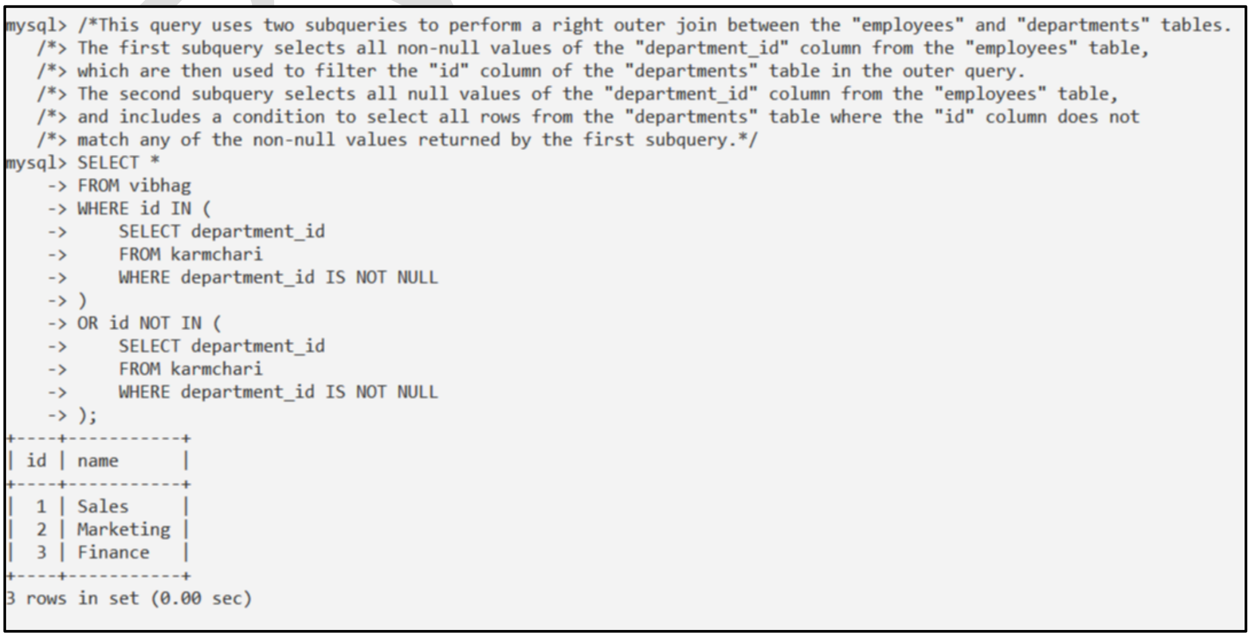
```
mysql> /*This query first selects the "id" column from the "departments" table
   /*>  for the department with the name "Sales". The result of this subquery is then used
   /*>  as a filter condition in the outer query, which selects all columns from the "employees"
   /*>  table where the "department_id" column matches any of the values returned by the subquery.*/
mysql> SELECT *
    -> FROM karmchari
    -> WHERE department_id IN (
    ->     SELECT id
    ->     FROM vibhag
    ->     WHERE name = 'Sales'
    -> );
+----+-------+------+---------------+--------+
| id | name  | age  | department_id | salary |
+----+-------+------+---------------+--------+
|  1 | Alice |   25 |             1 |  50000 |
|  4 | David |   40 |             1 |  80000 |
+----+-------+------+---------------+--------+
2 rows in set (0.00 sec)
```

### b] NESTED LEFT OUTER JOIN : -



```
mysql> /*This query uses a similar approach to the inner join example, but also
   /*> includes a condition to select all employees with a NULL value in the "department_id" column.*/
mysql> SELECT *
    -> FROM vibhag
    -> WHERE id IN (
    ->     SELECT department_id
    ->     FROM karmchari
    ->     WHERE department_id IS NOT NULL
    -> )
    -> OR id NOT IN (
    ->     SELECT department_id
    ->     FROM karmchari
    ->     WHERE department_id IS NOT NULL
    -> );
+----+-----------+
| id | name      |
+----+-----------+
|  1 | Sales     |
|  2 | Marketing |
|  3 | Finance   |
+----+-----------+
3 rows in set (0.01 sec)
```

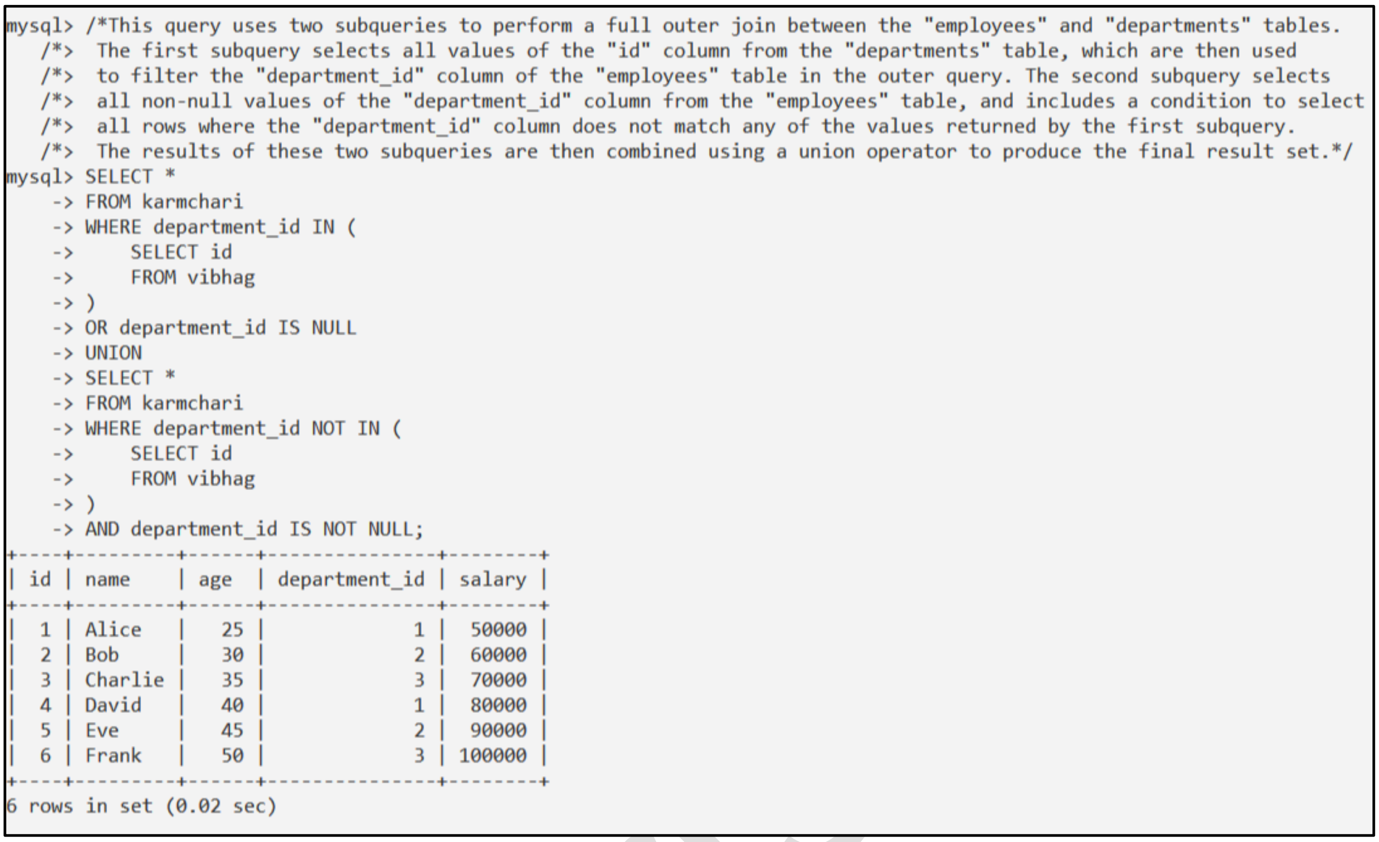
### c] NESTED RIGHT OUTER JOIN : -



```
mysql> /*This query uses two subqueries to perform a right outer join between the "employees" and "departments" tables.
   /*> The first subquery selects all non-null values of the "department_id" column from the "employees" table,
   /*> which are then used to filter the "id" column of the "departments" table in the outer query.
   /*> The second subquery selects all null values of the "department_id" column from the "employees" table,
   /*> and includes a condition to select all rows from the "departments" table where the "id" column does not
   /*> match any of the non-null values returned by the first subquery.*/
mysql> SELECT *
    -> FROM vibhag
    -> WHERE id IN (
    ->     SELECT department_id
    ->     FROM karmchari
    ->     WHERE department_id IS NOT NULL
    -> )
    -> OR id NOT IN (
    ->     SELECT department_id
    ->     FROM karmchari
    ->     WHERE department_id IS NOT NULL
    -> );
+----+-----------+
| id | name      |
+----+-----------+
|  1 | Sales     |
|  2 | Marketing |
|  3 | Finance   |
+----+-----------+
3 rows in set (0.00 sec)
```

P.T.O →

## d] NESTED FULL OUTER JOIN : -



```
mysql> /*This query uses two subqueries to perform a full outer join between the "employees" and "departments" tables.
   /*>  The first subquery selects all values of the "id" column from the "departments" table, which are then used
   /*>  to filter the "department_id" column of the "employees" table in the outer query. The second subquery selects
   /*>  all non-null values of the "department_id" column from the "employees" table, and includes a condition to select
   /*>  all rows where the "department_id" column does not match any of the values returned by the first subquery.
   /*>  The results of these two subqueries are then combined using a union operator to produce the final result set.*/
mysql> SELECT *
    -> FROM karmchari
    -> WHERE department_id IN (
    ->     SELECT id
    ->     FROM vibhag
    -> )
    -> OR department_id IS NULL
    -> UNION
    -> SELECT *
    -> FROM karmchari
    -> WHERE department_id NOT IN (
    ->     SELECT id
    ->     FROM vibhag
    -> )
    -> AND department_id IS NOT NULL;
+----+---------+------+---------------+--------+
| id | name    | age  | department_id | salary |
+----+---------+------+---------------+--------+
|  1 | Alice   |   25 |             1 |  50000 |
|  2 | Bob     |   30 |             2 |  60000 |
|  3 | Charlie |   35 |             3 |  70000 |
|  4 | David   |   40 |             1 |  80000 |
|  5 | Eve     |   45 |             2 |  90000 |
|  6 | Frank   |   50 |             3 | 100000 |
+----+---------+------+---------------+--------+
6 rows in set (0.02 sec)
```