```
-- Time Sheet db
USE timesheetdb;
-- Q1. Which PROJECT has maximum number of EMPLOYEES?
SELECT PROJECT_ID, PROJECT_NAME, EMPLOYEE_COUNT
FROM (
   SELECT P.PROJECT_ID, P.PROJECT_NAME, COUNT(A.EMP_ID) AS EMPLOYEE_COUNT,
        RANK() OVER (ORDER BY COUNT(A.EMP_ID) DESC) AS rnk
   FROM T_PROJECT P
   JOIN ALLOCATION A ON P.PROJECT_ID = A.PROJECT_ID
   GROUP BY P.PROJECT_ID, P.PROJECT_NAME
) rnked
WHERE rnk = 1;
-- Q2. Which EMPLOYEE has not yet been allocated to any PROJECT?
SELECT EMP_ID, EMP_NAME
FROM EMPLOYEE
WHERE EMP_ID NOT IN (
   SELECT EMP_ID
   FROM ALLOCATION
);

-- Q2. Which EMPLOYEE has not yet been allocated to any PROJECT?
SELECT E.EMP_ID, E.EMP_NAME
FROM EMPLOYEE E
LEFT JOIN ALLOCATION A ON E.EMP_ID = A.EMP_ID
WHERE A.PROJECT_ID IS NULL;
-- Q3. Which role played by the employee 'E03' frequently?
SELECT ROLE_TITLE , ROLE_ID
FROM (
   SELECT R.ROLE_TITLE, A.ROLE_ID ,  COUNT(A.ROLE_ID) AS RoleFrequency,
        RANK() OVER (ORDER BY COUNT(A.ROLE_ID) DESC) AS rnk
   FROM ALLOCATION A
```

```
    JOIN ROLE R ON A.ROLE_ID = R.ROLE_ID
    WHERE A.EMP_ID = 'E03'
    GROUP BY R.ROLE_TITLE , A.ROLE_ID
) ranked
WHERE rnk = 1;
-- Q4. Which is the costliest Project?
SELECT PROJECT_NAME, total_cost
FROM (
    SELECT p.PROJECT_NAME, SUM(a.AMOUNT_PER_DAY * DATEDIFF(a.TO_DATE, a.FROM_DATE)) AS
total_cost,
        RANK() OVER (ORDER BY SUM(a.AMOUNT_PER_DAY * DATEDIFF(a.TO_DATE, a.FROM_DATE))
DESC) AS rnk
    FROM ALLOCATION a
    JOIN T_PROJECT p ON a.PROJECT_ID = p.PROJECT_ID
    GROUP BY p.PROJECT_NAME
) ranked
WHERE rnk = 1;
/*5. How many employees were there in the costliest Project?*/
SELECT PROJECT_ID, PROJECT_NAME, EMPLOYEE_COUNT
FROM (
    SELECT P.PROJECT_ID, P.PROJECT_NAME, COUNT(DISTINCT A.EMP_ID) AS EMPLOYEE_COUNT,
        RANK() OVER (ORDER BY SUM(A.AMOUNT_PER_DAY * DATEDIFF(A.TO_DATE, A.FROM_DATE))
DESC) AS rnk
    FROM T_PROJECT P
    JOIN ALLOCATION A ON P.PROJECT_ID = A.PROJECT_ID
    GROUP BY P.PROJECT_ID, P.PROJECT_NAME
) rnked
WHERE rnk = 1;
-- Q6. Which is the cheapest Project in the year 2012?
SELECT PROJECT_ID, PROJECT_NAME, total_cost
FROM (
    SELECT p.PROJECT_ID, p.PROJECT_NAME,
```

```
        SUM(a.AMOUNT_PER_DAY * DATEDIFF(a.TO_DATE, a.FROM_DATE)) AS total_cost,
        RANK() OVER (ORDER BY SUM(a.AMOUNT_PER_DAY * DATEDIFF(a.TO_DATE, a.FROM_DATE))
ASC) AS rnk
    FROM T_PROJECT p
    JOIN ALLOCATION a ON p.PROJECT_ID = a.PROJECT_ID
    WHERE YEAR(a.TO_DATE) = 2012
    GROUP BY p.PROJECT_ID, p.PROJECT_NAME
) ranked
WHERE rnk = 1;
-- Q7. What is the salary of the employee who played maximum roles in Project 'P07'?
SELECT EMP_NAME, SALARY
FROM (
    SELECT e.EMP_ID, e.EMP_NAME, e.SALARY, COUNT(a.ROLE_ID) AS role_count,
        RANK() OVER (ORDER BY COUNT(a.ROLE_ID) DESC) AS rnk
    FROM EMPLOYEE e
    JOIN ALLOCATION a ON e.EMP_ID = a.EMP_ID
    WHERE a.PROJECT_ID = 'P07'
    GROUP BY e.EMP_ID, e.EMP_NAME, e.SALARY
) ranked
WHERE rnk = 1;
-- Q8. How many projects are handled by senior most employee?
SELECT EMP_NAME, PROJECT_COUNT
FROM (
    SELECT e.EMP_ID, e.EMP_NAME, COUNT(a.PROJECT_ID) AS PROJECT_COUNT,
        RANK() OVER (ORDER BY e.HIRE_DATE) AS rnk
    FROM EMPLOYEE e
    JOIN ALLOCATION a ON e.EMP_ID = a.EMP_ID
    GROUP BY e.EMP_ID, e.EMP_NAME
) ranked
WHERE rnk = 1;
/* 9. What is the total amount spent for unassigned employees? */
```

```
SELECT SUM(e.SALARY) AS total_spent
FROM EMPLOYEE e
LEFT JOIN ALLOCATION a ON e.EMP_ID = a.EMP_ID
WHERE a.PROJECT_ID IS NULL;
/* 10. How many projects are completed till date (Assume to_date is completion date in Allocation table)? */
SELECT COUNT(DISTINCT a.PROJECT_ID) AS completed_projects
FROM ALLOCATION a
WHERE a.TO_DATE <= CURRENT_DATE;

/* 11. How many employees have worked for less than 10 Projects? */
SELECT e.EMP_NAME ,COUNT(DISTINCT e.EMP_ID) AS employees_less_than_10_projects
FROM EMPLOYEE e
LEFT JOIN ALLOCATION a ON e.EMP_ID = a.EMP_ID
GROUP BY e.EMP_ID
HAVING COUNT(DISTINCT a.PROJECT_ID) < 10;

/* 12. How many employees are working with role 'R02' in project 'P04'? */
SELECT COUNT(DISTINCT a.EMP_ID) AS employees_in_role_R02_in_P04
FROM ALLOCATION a
WHERE a.PROJECT_ID = 'P04' AND a.ROLE_ID = 'R02';
/* 13. Which client has given maximum number of Projects? */
SELECT CLIENT_NAME, PROJECT_COUNT
FROM (
   SELECT P.CLIENT_NAME,
        COUNT(P.PROJECT_ID) AS PROJECT_COUNT,
        RANK() OVER (ORDER BY COUNT(P.PROJECT_ID) DESC) AS rnk
   FROM T_PROJECT P
   GROUP BY P.CLIENT_NAME
) ranked
WHERE rnk = 1;
```

```
/* 14. Which employee has not been allocated to any project in the year 2010? */
SELECT EMP_ID, EMP_NAME
FROM (
   SELECT E.EMP_ID,
        E.EMP_NAME,
        COUNT(A.PROJECT_ID) AS PROJECT_COUNT
   FROM EMPLOYEE E
   LEFT JOIN ALLOCATION A
      ON E.EMP_ID = A.EMP_ID
      AND YEAR(A.FROM_DATE) = 2010
   GROUP BY E.EMP_ID, E.EMP_NAME
) sub
WHERE PROJECT_COUNT = 0;
/*15.Find the total number of days worked by the employee 'E04' in project 'P02'?*/
  SELECT DATEDIFF(a.TO_DATE, a.FROM_DATE) AS total_days_worked
    FROM ALLOCATION a
    WHERE a.EMP_ID = 'E04' AND a.PROJECT_ID = 'P02';
/* 15. Find the total number of days worked by the employee 'E04' in project 'P02' */
SELECT EMP_ID, PROJECT_ID, TOTAL_DAYS
FROM (
   SELECT A.EMP_ID,
        A.PROJECT_ID,
        SUM(DATEDIFF(A.TO_DATE, A.FROM_DATE)) AS TOTAL_DAYS
   FROM ALLOCATION A
   WHERE A.EMP_ID = 'E04'
    AND A.PROJECT_ID = 'P02'
   GROUP BY A.EMP_ID, A.PROJECT_ID
) sub;
/*16.Which Project has been completed exactly on deadline date?*/
SELECT p.PROJECT_NAME
FROM T_PROJECT p
```

```
JOIN ALLOCATION a ON p.PROJECT_ID = a.PROJECT_ID
WHERE a.TO_DATE = p.DEADLINE;
/*16.Which Project has been completed exactly on deadline date?*/
SELECT PROJECT_ID, PROJECT_NAME
FROM (
   SELECT p.PROJECT_ID,
        p.PROJECT_NAME,
        MAX(a.TO_DATE) AS COMPLETION_DATE,
        p.DEADLINE_DATE,
        RANK() OVER (ORDER BY p.PROJECT_ID) AS rnk
   FROM T_PROJECT p
   JOIN ALLOCATION a ON p.PROJECT_ID = a.PROJECT_ID
   GROUP BY p.PROJECT_ID, p.PROJECT_NAME, p.DEADLINE_DATE
) ranked
WHERE COMPLETION_DATE = DEADLINE_DATE;
/*17.How many employees were working for the Project, which has crossed the deadline?*/
  SELECT COUNT(DISTINCT a.EMP_ID) AS employees_working_on_overdue_project
    FROM ALLOCATION a
    JOIN T_PROJECT p ON a.PROJECT_ID = p.PROJECT_ID
    WHERE a.TO_DATE > p.DEADLINE;
/*18.Which Project has been completed so earlier?*/
SELECT PROJECT_ID, PROJECT_NAME, earliest_completion_date
FROM (
   SELECT p.PROJECT_ID,
        p.PROJECT_NAME,
        MIN(a.TO_DATE) AS earliest_completion_date,
        RANK() OVER (ORDER BY MIN(a.TO_DATE) ASC) AS rnk
   FROM T_PROJECT p
   JOIN ALLOCATION a ON p.PROJECT_ID = a.PROJECT_ID
   GROUP BY p.PROJECT_ID, p.PROJECT_NAME
) ranked
```

```
WHERE rnk = 1;
/*19.Which Project has taken maximum duration?*/
SELECT PROJECT_ID, PROJECT_NAME, total_duration
FROM (
   SELECT p.PROJECT_ID,
        p.PROJECT_NAME,
        SUM(DATEDIFF(a.TO_DATE, a.FROM_DATE)) AS total_duration,
        RANK() OVER (ORDER BY SUM(DATEDIFF(a.TO_DATE, a.FROM_DATE)) DESC) AS rnk
   FROM T_PROJECT p
   JOIN ALLOCATION a ON p.PROJECT_ID = a.PROJECT_ID
   GROUP BY p.PROJECT_ID, p.PROJECT_NAME
) ranked
WHERE rnk = 1;
/*20.Prepare a report in following format
Emp Id     Total Number of Days in Bench */

SELECT e.EMP_ID,
          DATEDIFF(CURDATE(), e.HIRE_DATE) AS Total_Days_in_Bench
     FROM EMPLOYEE e
     LEFT JOIN ALLOCATION a ON e.EMP_ID = a.EMP_ID
     WHERE a.PROJECT_ID IS NULL;
/*21.Prepare a report in following format
  /*> Project Name Number of Employees*/
  SELECT p.PROJECT_NAME,
          COUNT(DISTINCT a.EMP_ID) AS Number_of_Employees
     FROM ALLOCATION a
     JOIN T_PROJECT p ON a.PROJECT_ID = p.PROJECT_ID
     GROUP BY p.PROJECT_NAME;
/*22.Prepare a report in following format
  /*> Role Name Number of Employees*/
```

```
  SELECT r.ROLE_TITLE,
        COUNT(DISTINCT a.EMP_ID) AS Number_of_Employees
    FROM ALLOCATION a
    JOIN ROLE r ON a.ROLE_ID = r.ROLE_ID
    GROUP BY r.ROLE_TITLE;
/*23.Prepare a report in following format
  /*> Emp Name Number of Projects
  /*> */
  SELECT e.EMP_NAME,
        COUNT(DISTINCT a.PROJECT_ID) AS Number_of_Projects
    FROM ALLOCATION a
    JOIN EMPLOYEE e ON a.EMP_ID = e.EMP_ID
    GROUP BY e.EMP_NAME;
/*24.Prepare a report in following format
  /*> Emp Name Number of Roles*/
  SELECT e.EMP_NAME,
        COUNT(DISTINCT a.ROLE_ID) AS Number_of_Roles
    FROM ALLOCATION a
    JOIN EMPLOYEE e ON a.EMP_ID = e.EMP_ID
    GROUP BY e.EMP_NAME;
/*25.Prepare a report in this format
  /*> Role Name Number of Employees*/
  SELECT r.ROLE_TITLE,
        COUNT(DISTINCT a.EMP_ID) AS Number_of_Employees
    FROM ALLOCATION a
    JOIN ROLE r ON a.ROLE_ID = r.ROLE_ID
    GROUP BY r.ROLE_TITLE;
/*26.Prepare a report in this format
  /*> Role Name Number of Projects*/
  SELECT r.ROLE_TITLE,
        COUNT(DISTINCT a.PROJECT_ID) AS Number_of_Projects
```

```
    FROM ALLOCATION a
    JOIN ROLE r ON a.ROLE_ID = r.ROLE_ID
    GROUP BY r.ROLE_TITLE;
/* 27.Prepare a report in this format
  /*> Emp Name Role Name Number of Projects
  /*> */
  SELECT e.EMP_NAME,
         r.ROLE_TITLE,
         COUNT(DISTINCT a.PROJECT_ID) AS Number_of_Projects
    FROM ALLOCATION a
    JOIN EMPLOYEE e ON a.EMP_ID = e.EMP_ID
    JOIN ROLE r ON a.ROLE_ID = r.ROLE_ID
    GROUP BY e.EMP_NAME, r.ROLE_TITLE;
/*28.Prepare a report in this format
  /*> Project Name Role Name Number of Employees*/
  SELECT p.PROJECT_NAME,
         r.ROLE_TITLE,
         COUNT(DISTINCT a.EMP_ID) AS Number_of_Employees
    FROM ALLOCATION a
    JOIN T_PROJECT p ON a.PROJECT_ID = p.PROJECT_ID
    JOIN ROLE r ON a.ROLE_ID = r.ROLE_ID
    GROUP BY p.PROJECT_NAME, r.ROLE_TITLE;
/*29.Prepare a report in this format
  /*> Role Name Emp Name Number of Projects
  /*> */
  SELECT r.ROLE_TITLE,
         e.EMP_NAME,
         COUNT(DISTINCT a.PROJECT_ID) AS Number_of_Projects
    FROM ALLOCATION a
    JOIN EMPLOYEE e ON a.EMP_ID = e.EMP_ID
    JOIN ROLE r ON a.ROLE_ID = r.ROLE_ID
```

```
    GROUP BY r.ROLE_TITLE, e.EMP_NAME;


/*30.Prepare a report in this format
/*> Dept Id Number of Employees*/
  SELECT e.DEPT_ID,
        COUNT(e.EMP_ID) AS Number_of_Employees
    FROM EMPLOYEE e
    GROUP BY e.DEPT_ID;
/*31.Prepare a report in this format
  /*> Mgr_id Number of Employees*/
  SELECT e.MGR_ID,
        COUNT(e.EMP_ID) AS Number_of_Employees
    FROM EMPLOYEE e
    WHERE e.MGR_ID IS NOT NULL
    GROUP BY e.MGR_ID;
  /*32.Prepare a report in this format
  /*> Emp Name Role Name Project Name*/
  SELECT e.EMP_NAME,
        r.ROLE_TITLE,
        p.PROJECT_NAME
    FROM ALLOCATION a
    JOIN EMPLOYEE e ON a.EMP_ID = e.EMP_ID
    JOIN ROLE r ON a.ROLE_ID = r.ROLE_ID
    JOIN T_PROJECT p ON a.PROJECT_ID = p.PROJECT_ID;
/*33. Prepare a report in this format: Project ID, Emp ID, Total Amount collected
 Sort the report with respect to the Total Amount collected in Descending Order*/


SELECT a.PROJECT_ID, a.EMP_ID,
     SUM(a.AMOUNT_PER_DAY * DATEDIFF(a.TO_DATE, a.FROM_DATE)) AS
TOTAL_AMOUNT_COLLECTED
FROM ALLOCATION a
```

```
GROUP BY a.PROJECT_ID, a.EMP_ID
ORDER BY TOTAL_AMOUNT_COLLECTED DESC;
/*33.Prepare a report in this format
  /*> Project id Emp id Total Amount collected*/
  SELECT a.PROJECT_ID,
         a.EMP_ID,
         SUM(m.AMOUNT) AS Total_Amount_Collected
    FROM ALLOCATION a
    JOIN MOBILERECHARGE m ON a.EMP_ID = m.MOBILE_NO
    GROUP BY a.PROJECT_ID, a.EMP_ID;
/* 34. Prepare a report in this format: Emp ID, Role ID, Total Amount Collected */

SELECT
   a.EMP_ID,
   a.ROLE_ID,
   SUM(a.AMOUNT_PER_DAY * DATEDIFF(a.TO_DATE, a.FROM_DATE)) AS Total_Amount_Collected
FROM
   ALLOCATION a
GROUP BY
   a.EMP_ID, a.ROLE_ID;
/*34.Prepare a report in this format
  /*> Emp id Role id Total Amount collected*/
  SELECT a.EMP_ID,
         a.ROLE_ID,
         SUM(m.AMOUNT) AS Total_Amount_Collected
    FROM ALLOCATION a
    JOIN MOBILERECHARGE m ON a.EMP_ID = m.MOBILE_NO
    GROUP BY a.EMP_ID, a.ROLE_ID;
/*35.Prepare a report in this format
  /*> Emp id Role id Project id Total Amount collected
  /*> Sort the report with respect to the Total Amount collected in Descending Order.*/
```

```
  SELECT a.EMP_ID,
         a.ROLE_ID,
         a.PROJECT_ID,
         SUM(m.AMOUNT) AS Total_Amount_Collected
    FROM ALLOCATION a
    JOIN MOBILERECHARGE m ON a.EMP_ID = m.MOBILE_NO
    GROUP BY a.EMP_ID, a.ROLE_ID, a.PROJECT_ID
    ORDER BY Total_Amount_Collected DESC;
/*35. Prepare a report in this format:
 Emp ID | Role ID | Project ID | Total Amount Collected
 Sorted by Total Amount Collected (Descending)*/
SELECT a.EMP_ID,
     a.ROLE_ID,
     a.PROJECT_ID,
     SUM(a.AMOUNT_PER_DAY * DATEDIFF(a.TO_DATE, a.FROM_DATE)) AS
TOTAL_AMOUNT_COLLECTED
FROM ALLOCATION a
GROUP BY a.EMP_ID, a.ROLE_ID, a.PROJECT_ID
ORDER BY TOTAL_AMOUNT_COLLECTED DESC;
/*36. Emp ID | Mgr ID | Comments */
-- Emp ID | Mgr ID | Comments
-- If Manager ID is NULL, comment should be “No Manager” else “Has Manager”
SELECT EMP_ID,
     MGR_ID,
     CASE
        WHEN MGR_ID IS NULL THEN 'No Manager'
        ELSE 'Has Manager'
     END AS Comments
FROM EMPLOYEE;
/*37. Ram works for Ashok */
-- Ram works for Ashok
```

```
-- Where Ram is EMP_NAME and Ashok is his corresponding Manager's EMP_NAME
SELECT e.EMP_NAME || ' works for ' || m.EMP_NAME AS Relation
FROM EMPLOYEE e
JOIN EMPLOYEE m ON e.MGR_ID = m.EMP_ID;

/*38. Employees earning more than their managers */

SELECT e.EMP_ID, e.EMP_NAME, e.SALARY AS EMP_SALARY,
      m.EMP_NAME AS MANAGER_NAME, m.SALARY AS MANAGER_SALARY
FROM EMPLOYEE e
JOIN EMPLOYEE m ON e.MGR_ID = m.EMP_ID
WHERE e.SALARY > m.SALARY;

/*39. Managers who joined after their subordinates */

SELECT e.EMP_ID AS EMPLOYEE_ID, e.EMP_NAME AS EMPLOYEE,
      m.EMP_ID AS MANAGER_ID, m.EMP_NAME AS MANAGER,
      e.HIRE_DATE AS EMP_HIRE, m.HIRE_DATE AS MGR_HIRE
FROM EMPLOYEE e
JOIN EMPLOYEE m ON e.MGR_ID = m.EMP_ID
WHERE e.HIRE_DATE < m.HIRE_DATE;

/*40. Employees earning more than avg salary of their department */

WITH DeptAvgSal AS (
   SELECT DEPT_ID, AVG(SALARY) AS AVG_SAL
   FROM EMPLOYEE
   GROUP BY DEPT_ID
)

SELECT e.EMP_ID, e.EMP_NAME, e.DEPT_ID, e.SALARY
```

```
FROM EMPLOYEE e
WHERE e.SALARY > (
   SELECT AVG_SAL
   FROM DeptAvgSal d
   WHERE d.DEPT_ID = e.DEPT_ID
);

/*41. Employees who have changed their roles at least twice - Using Correlated Subquery */

SELECT e.EMP_ID, e.EMP_NAME
FROM EMPLOYEE e
WHERE (
   SELECT COUNT(DISTINCT a.ROLE_ID)
   FROM ALLOCATION a
   WHERE a.EMP_ID = e.EMP_ID
) >= 3;
/*42. Departments with no employees */
 -- 42.Display the departments that does not have employees(ALL POSSIBILITIES)?
SELECT d.DEPT_ID, d.DEPT_NAME
FROM DEPARTMENT d
LEFT JOIN EMPLOYEE e ON d.DEPT_ID = e.DEPT_ID
WHERE e.EMP_ID IS NULL;

/*43. Departments with at least one employee */

SELECT DISTINCT d.DEPT_ID, d.DEPT_NAME
FROM DEPARTMENT d
JOIN EMPLOYEE e ON d.DEPT_ID = e.DEPT_ID;
/* 44. Using ROLLUP
  > Generate a report:
  > Project ID | Role ID | No of Employees */
```

```
SELECT
    a.PROJECT_ID,
    a.ROLE_ID,
    COUNT(DISTINCT a.EMP_ID) AS No_of_Employees
FROM ALLOCATION a
GROUP BY a.PROJECT_ID, a.ROLE_ID WITH ROLLUP;

/*45. Using ROLL UP and CUBE
  /*> Generate a report:
  /*> Employee ID Project ID Total salary
  /*> */
SELECT a.EMP_ID,
          a.PROJECT_ID,
          SUM(e.SALARY) AS Total_Salary
     FROM ALLOCATION a
     JOIN EMPLOYEE e ON a.EMP_ID = e.EMP_ID
     GROUP BY a.EMP_ID, a.PROJECT_ID WITH ROLLUP;
-- 46. Hierarchical report starting with 'Raja'
WITH RECURSIVE Employee_Hierarchy AS (
    SELECT
        e.EMP_ID,
        e.EMP_NAME,
        e.MGR_ID,
        e.ROLE_ID,
        a.PROJECT_ID,
        1 AS LEVEL
    FROM EMPLOYEE e
    LEFT JOIN ALLOCATION a ON e.EMP_ID = a.EMP_ID
    WHERE e.EMP_NAME = 'Raja'
```

```
    UNION ALL

    SELECT
        e.EMP_ID,
        e.EMP_NAME,
        e.MGR_ID,
        e.ROLE_ID,
        a.PROJECT_ID,
        eh.LEVEL + 1
    FROM EMPLOYEE e
    LEFT JOIN ALLOCATION a ON e.EMP_ID = a.EMP_ID
    JOIN Employee_Hierarchy eh ON e.MGR_ID = eh.EMP_ID
)

SELECT LEVEL, EMP_NAME AS EMPLOYEE, ROLE_ID, PROJECT_ID
FROM Employee_Hierarchy
ORDER BY LEVEL;
-- 47. Classify employees based on number of skills
SELECT
    e.EMP_ID,
    COUNT(s.SKILL_ID) AS Number_of_Skills,
    CASE
        WHEN COUNT(s.SKILL_ID) > 5 THEN 'Major Resource'
        WHEN COUNT(s.SKILL_ID) > 3 THEN 'Useful Resource'
        WHEN COUNT(s.SKILL_ID) > 1 THEN 'Resource'
        ELSE 'Needs Training'
    END AS Description
FROM EMPLOYEE e
LEFT JOIN SKILLS s ON e.EMP_ID = s.EMP_ID
GROUP BY e.EMP_ID;
-- 48. Leave description based on number of leaves
```

```
SELECT
   l.EMP_ID,
   COUNT(l.LEAVE_ID) AS No_of_Leaves,
   CASE
      WHEN COUNT(l.LEAVE_ID) = 0 THEN 'Bonus'
      WHEN COUNT(l.LEAVE_ID) <= 6 THEN 'No loss of pay'
      ELSE 'Loss of Pay'
   END AS Description
FROM LEAVE l
RIGHT JOIN EMPLOYEE e ON e.EMP_ID = l.EMP_ID
GROUP BY e.EMP_ID;
-- 49. Top 5 salaried employees using RANK and subquery
SELECT *
FROM (
   SELECT *,
        RANK() OVER (ORDER BY SALARY DESC) AS rnk
   FROM EMPLOYEE
) ranked_employees
WHERE rnk <= 5;
/*50. List TOP 3 Departments (with respect to maximum number of employees)*/
SELECT *
FROM (
   SELECT d.DEPT_ID,
        COUNT(e.EMP_ID) AS No_of_Employees,
        RANK() OVER (ORDER BY COUNT(e.EMP_ID) DESC) AS rnk
   FROM DEPARTMENT d
   JOIN EMPLOYEE e ON d.DEPT_ID = e.DEPT_ID
   GROUP BY d.DEPT_ID
) ranked_departments
WHERE rnk <= 3;
```

```sql
-- 51. 2nd max salary per department
SELECT *
FROM (
   SELECT
      DEPT_ID,
      EMP_ID,
      SALARY,
      RANK() OVER (PARTITION BY DEPT_ID ORDER BY SALARY DESC) AS rnk
   FROM EMPLOYEE
) ranked
WHERE rnk = 2;

/*52.Generate a report:
  /*> Emp Name Number of skills*/
  SELECT e.EMP_NAME,
         COUNT(es.SKILL_ID) AS Number_of_Skills
    FROM EMPLOYEE e
    LEFT JOIN EMPLOYEESKILL es ON e.EMP_ID = es.EMP_ID
    GROUP BY e.EMP_NAME;


  /*53.Generate a report :
  /*> Emp Name Number of recharges done so far
  /*> */
  SELECT e.EMP_NAME,
         COUNT(m.TRANS_ID) AS Number_of_Recharges
    FROM EMPLOYEE e
    LEFT JOIN MOBILERECHARGE m ON e.MOBILE_NO = m.MOBILE_NO
    GROUP BY e.EMP_NAME;

/*54.Delete duplicate rows from Employee table(Using ROWID)?*/
```

```
DELETE FROM EMPLOYEE e
  WHERE e.ROWID NOT IN (
     SELECT MIN(e1.ROWID)
     FROM EMPLOYEE e1
     GROUP BY e1.EMP_ID, e1.EMP_NAME, e1.SALARY, e1.DEPT_ID, e1.MOBILE_NO
  );
```