-- Time Sheet db

USE timesheetdb;

-- Q1. Which PROJECT has maximum number of EMPLOYEES?

-- Original: JOIN with RANK() window function

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;

-- Alternative 1: JOIN with subquery

SELECT P.PROJECT\_ID, P.PROJECT\_NAME, COUNT(A.EMP\_ID) AS EMPLOYEE\_COUNT

FROM T\_PROJECT P

JOIN ALLOCATION A ON P.PROJECT\_ID = A.PROJECT\_ID

GROUP BY P.PROJECT\_ID, P.PROJECT\_NAME

HAVING COUNT(A.EMP\_ID) = (

SELECT MAX(emp\_count)

FROM (

SELECT COUNT(EMP\_ID) AS emp\_count

FROM ALLOCATION

GROUP BY PROJECT\_ID

) counts

);

-- Alternative 2: Using NOT IN (finding projects that don't have fewer employees than any other)

SELECT P.PROJECT\_ID, P.PROJECT\_NAME, COUNT(A.EMP\_ID) AS EMPLOYEE\_COUNT

FROM T\_PROJECT P

JOIN ALLOCATION A ON P.PROJECT\_ID = A.PROJECT\_ID

GROUP BY P.PROJECT\_ID, P.PROJECT\_NAME

HAVING COUNT(A.EMP\_ID) NOT IN (

SELECT COUNT(A2.EMP\_ID)

FROM ALLOCATION A2

GROUP BY A2.PROJECT\_ID

HAVING COUNT(A2.EMP\_ID) < (

SELECT MAX(emp\_count)

FROM (

SELECT COUNT(EMP\_ID) AS emp\_count

FROM ALLOCATION

GROUP BY PROJECT\_ID

) max\_counts

)

);

-- Q2. Which EMPLOYEE has not yet been allocated to any PROJECT?

-- Original 1: NOT IN

SELECT EMP\_ID, EMP\_NAME

FROM EMPLOYEE

WHERE EMP\_ID NOT IN (

SELECT EMP\_ID

FROM ALLOCATION

);

-- Original 2: LEFT JOIN

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;

-- Alternative 1: NOT EXISTS (correlated subquery)

SELECT EMP\_ID, EMP\_NAME

FROM EMPLOYEE E

WHERE NOT EXISTS (

SELECT 1

FROM ALLOCATION A

WHERE A.EMP\_ID = E.EMP\_ID

);

-- Alternative 2: EXCEPT (set operator)

SELECT EMP\_ID, EMP\_NAME

FROM EMPLOYEE

EXCEPT

SELECT E.EMP\_ID, E.EMP\_NAME

FROM EMPLOYEE E

JOIN ALLOCATION A ON E.EMP\_ID = A.EMP\_ID;

-- Q3. Which role played by the employee 'E03' frequently?

-- Original: JOIN with RANK() window function

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;

-- Alternative 1: JOIN with subquery

SELECT R.ROLE\_TITLE, A.ROLE\_ID

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

HAVING COUNT(\*) = (

SELECT MAX(role\_count)

FROM (

SELECT COUNT(\*) AS role\_count

FROM ALLOCATION

WHERE EMP\_ID = 'E03'

GROUP BY ROLE\_ID

) counts

);

-- Q4. Which is the costliest Project?

-- Original: JOIN with RANK() window function

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;

-- Alternative 1: JOIN with subquery

SELECT p.PROJECT\_NAME,

SUM(a.AMOUNT\_PER\_DAY \* DATEDIFF(a.TO\_DATE, a.FROM\_DATE)) AS total\_cost

FROM ALLOCATION a

JOIN T\_PROJECT p ON a.PROJECT\_ID = p.PROJECT\_ID

GROUP BY p.PROJECT\_NAME

HAVING SUM(a.AMOUNT\_PER\_DAY \* DATEDIFF(a.TO\_DATE, a.FROM\_DATE)) = (

SELECT MAX(total\_cost)

FROM (

SELECT SUM(AMOUNT\_PER\_DAY \* DATEDIFF(TO\_DATE, FROM\_DATE)) AS total\_cost

FROM ALLOCATION

GROUP BY PROJECT\_ID

) costs

);

-- Q5. How many employees were there in the costliest Project?

-- Original: JOIN with RANK() window function

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;

-- Alternative 1: JOIN with subquery

SELECT P.PROJECT\_ID, P.PROJECT\_NAME, COUNT(DISTINCT A.EMP\_ID) AS EMPLOYEE\_COUNT

FROM T\_PROJECT P

JOIN ALLOCATION A ON P.PROJECT\_ID = A.PROJECT\_ID

GROUP BY P.PROJECT\_ID, P.PROJECT\_NAME

HAVING SUM(A.AMOUNT\_PER\_DAY \* DATEDIFF(A.TO\_DATE, A.FROM\_DATE)) = (

SELECT MAX(total\_cost)

FROM (

SELECT SUM(AMOUNT\_PER\_DAY \* DATEDIFF(TO\_DATE, FROM\_DATE)) AS total\_cost

FROM ALLOCATION

GROUP BY PROJECT\_ID

) costs

);

-- Q6. Which is the cheapest Project in the year 2012?

-- Original: JOIN with RANK() window function

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;

-- Alternative 1: JOIN with subquery

SELECT p.PROJECT\_ID, p.PROJECT\_NAME,

SUM(a.AMOUNT\_PER\_DAY \* DATEDIFF(a.TO\_DATE, a.FROM\_DATE)) AS total\_cost

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

HAVING SUM(a.AMOUNT\_PER\_DAY \* DATEDIFF(a.TO\_DATE, a.FROM\_DATE)) = (

SELECT MIN(total\_cost)

FROM (

SELECT SUM(AMOUNT\_PER\_DAY \* DATEDIFF(TO\_DATE, FROM\_DATE)) AS total\_cost

FROM ALLOCATION

WHERE YEAR(TO\_DATE) = 2012

GROUP BY PROJECT\_ID

) costs

);

-- Q7. What is the salary of the employee who played maximum roles in Project 'P07'?

-- Original: JOIN with RANK() window function

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;

-- Alternative 1: JOIN with subquery

SELECT e.EMP\_NAME, e.SALARY

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

HAVING COUNT(a.ROLE\_ID) = (

SELECT MAX(role\_count)

FROM (

SELECT COUNT(ROLE\_ID) AS role\_count

FROM ALLOCATION

WHERE PROJECT\_ID = 'P07'

GROUP BY EMP\_ID

) counts

);

-- Q8. How many projects are handled by senior most employee?

-- Original: JOIN with RANK() window function

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;

-- Alternative 1: JOIN with subquery

SELECT e.EMP\_NAME, COUNT(a.PROJECT\_ID) AS PROJECT\_COUNT

FROM EMPLOYEE e

JOIN ALLOCATION a ON e.EMP\_ID = a.EMP\_ID

WHERE e.HIRE\_DATE = (

SELECT MIN(HIRE\_DATE)

FROM EMPLOYEE

)

GROUP BY e.EMP\_ID, e.EMP\_NAME;

-- Q9. What is the total amount spent for unassigned employees?

-- Original: LEFT JOIN

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;

-- Alternative 1: NOT EXISTS (correlated subquery)

SELECT SUM(SALARY) AS total\_spent

FROM EMPLOYEE e

WHERE NOT EXISTS (

SELECT 1

FROM ALLOCATION A

WHERE A.EMP\_ID = e.EMP\_ID

);

-- Alternative 2: NOT IN

SELECT SUM(SALARY) AS total\_spent

FROM EMPLOYEE

WHERE EMP\_ID NOT IN (

SELECT DISTINCT EMP\_ID

FROM ALLOCATION

);

-- Alternative 3: EXCEPT (set operator)

SELECT SUM(SALARY) AS total\_spent

FROM (

SELECT EMP\_ID, SALARY

FROM EMPLOYEE

EXCEPT

SELECT E.EMP\_ID, E.SALARY

FROM EMPLOYEE E

JOIN ALLOCATION A ON E.EMP\_ID = A.EMP\_ID

) unassigned;

-- Q10. How many projects are completed till date?

-- Original: Simple query

SELECT COUNT(DISTINCT a.PROJECT\_ID) AS completed\_projects

FROM ALLOCATION a

WHERE a.TO\_DATE <= CURRENT\_DATE;

-- Alternative 1: Using JOIN

SELECT COUNT(DISTINCT p.PROJECT\_ID) AS completed\_projects

FROM T\_PROJECT p

JOIN ALLOCATION a ON p.PROJECT\_ID = a.PROJECT\_ID

WHERE a.TO\_DATE <= CURRENT\_DATE;

-- Q11. How many employees have worked for less than 10 Projects?

-- Original: LEFT JOIN with GROUP BY and HAVING

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;

-- Alternative 1: Using subquery

SELECT COUNT(\*) AS employees\_less\_than\_10\_projects

FROM (

SELECT EMP\_ID

FROM ALLOCATION

GROUP BY EMP\_ID

HAVING COUNT(DISTINCT PROJECT\_ID) < 10

) sub;

-- Q12. How many employees are working with role 'R02' in project 'P04'?

-- Original: Simple query

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';

-- Alternative 1: Using JOIN

SELECT COUNT(DISTINCT e.EMP\_ID) AS employees\_in\_role\_R02\_in\_P04

FROM EMPLOYEE e

JOIN ALLOCATION a ON e.EMP\_ID = a.EMP\_ID

WHERE a.PROJECT\_ID = 'P04' AND a.ROLE\_ID = 'R02';

-- Q13. Which client has given maximum number of Projects?

-- Original: RANK() window function

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;

-- Alternative 1: Using JOIN with subquery

SELECT CLIENT\_NAME, COUNT(PROJECT\_ID) AS PROJECT\_COUNT

FROM T\_PROJECT

GROUP BY CLIENT\_NAME

HAVING COUNT(PROJECT\_ID) = (

SELECT MAX(project\_count)

FROM (

SELECT COUNT(PROJECT\_ID) AS project\_count

FROM T\_PROJECT

GROUP BY CLIENT\_NAME

) counts

);

-- Q14. Which employee has not been allocated to any project in the year 2010?

-- Original: LEFT JOIN with filtering

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;

-- Alternative 1: NOT EXISTS (correlated subquery)

SELECT EMP\_ID, EMP\_NAME

FROM EMPLOYEE E

WHERE NOT EXISTS (

SELECT 1

FROM ALLOCATION A

WHERE A.EMP\_ID = E.EMP\_ID

AND YEAR(A.FROM\_DATE) = 2010

);

-- Alternative 2: EXCEPT (set operator)

SELECT EMP\_ID, EMP\_NAME

FROM EMPLOYEE

EXCEPT

SELECT E.EMP\_ID, E.EMP\_NAME

FROM EMPLOYEE E

JOIN ALLOCATION A ON E.EMP\_ID = A.EMP\_ID

WHERE YEAR(A.FROM\_DATE) = 2010;

-- Q15. Find the total number of days worked by the employee 'E04' in project 'P02'

-- Original: Simple query

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;

-- Alternative 1: Using JOIN

SELECT A.EMP\_ID, A.PROJECT\_ID,

SUM(DATEDIFF(A.TO\_DATE, A.FROM\_DATE)) AS TOTAL\_DAYS

FROM ALLOCATION A

JOIN EMPLOYEE E ON A.EMP\_ID = E.EMP\_ID

WHERE A.EMP\_ID = 'E04' AND A.PROJECT\_ID = 'P02'

GROUP BY A.EMP\_ID, A.PROJECT\_ID;

-- Q16. Which Project has been completed exactly on deadline date?

-- Original: JOIN

SELECT p.PROJECT\_NAME

FROM T\_PROJECT p

JOIN ALLOCATION a ON p.PROJECT\_ID = a.PROJECT\_ID

WHERE a.TO\_DATE = p.DEADLINE;

-- Alternative 1: EXISTS (correlated subquery)

SELECT p.PROJECT\_NAME

FROM T\_PROJECT p

WHERE EXISTS (

SELECT 1

FROM ALLOCATION a

WHERE a.PROJECT\_ID = p.PROJECT\_ID

AND a.TO\_DATE = p.DEADLINE

);

-- Q17. How many employees were working for the Project, which has crossed the deadline?

-- Original: JOIN

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;

-- Alternative 1: Using subquery

SELECT COUNT(DISTINCT EMP\_ID) AS employees\_working\_on\_overdue\_project

FROM ALLOCATION

WHERE PROJECT\_ID IN (

SELECT PROJECT\_ID

FROM T\_PROJECT p

JOIN ALLOCATION a ON p.PROJECT\_ID = a.PROJECT\_ID

WHERE a.TO\_DATE > p.DEADLINE

);

-- Q18. Which Project has been completed so earlier?

-- Original: RANK() window function

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;

-- Alternative 1: JOIN with subquery

SELECT p.PROJECT\_ID, p.PROJECT\_NAME, MIN(a.TO\_DATE) AS earliest\_completion\_date

FROM T\_PROJECT p

JOIN ALLOCATION a ON p.PROJECT\_ID = a.PROJECT\_ID

GROUP BY p.PROJECT\_ID, p.PROJECT\_NAME

HAVING MIN(a.TO\_DATE) = (

SELECT MIN(completion\_date)

FROM (

SELECT MIN(TO\_DATE) AS completion\_date

FROM ALLOCATION

GROUP BY PROJECT\_ID

) dates

);

-- Q19. Which Project has taken maximum duration?

-- Original: RANK() window function

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;

-- Alternative 1: JOIN with subquery

SELECT p.PROJECT\_ID, p.PROJECT\_NAME,

SUM(DATEDIFF(a.TO\_DATE, a.FROM\_DATE)) AS total\_duration

FROM T\_PROJECT p

JOIN ALLOCATION a ON p.PROJECT\_ID = a.PROJECT\_ID

GROUP BY p.PROJECT\_ID, p.PROJECT\_NAME

HAVING SUM(DATEDIFF(a.TO\_DATE, a.FROM\_DATE)) = (

SELECT MAX(duration)

FROM (

SELECT SUM(DATEDIFF(TO\_DATE, FROM\_DATE)) AS duration

FROM ALLOCATION

GROUP BY PROJECT\_ID

) durations

);

-- Q20. Prepare a report showing Emp Id and Total Number of Days in Bench

-- Original: LEFT JOIN

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;

-- Alternative 1: NOT EXISTS (correlated subquery)

SELECT EMP\_ID,

DATEDIFF(CURDATE(), HIRE\_DATE) AS Total\_Days\_in\_Bench

FROM EMPLOYEE e

WHERE NOT EXISTS (

SELECT 1

FROM ALLOCATION a

WHERE a.EMP\_ID = e.EMP\_ID

);

-- Alternative 2: EXCEPT (set operator)

SELECT EMP\_ID, DATEDIFF(CURDATE(), HIRE\_DATE) AS Total\_Days\_in\_Bench

FROM (

SELECT EMP\_ID, HIRE\_DATE

FROM EMPLOYEE

EXCEPT

SELECT E.EMP\_ID, E.HIRE\_DATE

FROM EMPLOYEE E

JOIN ALLOCATION A ON E.EMP\_ID = A.EMP\_ID

) benched\_employees;

-- Q21. Project Name and Number of Employees

-- Original: JOIN with GROUP BY

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;

-- Alternative 1: Using subquery

SELECT p.PROJECT\_NAME,

(SELECT COUNT(DISTINCT a2.EMP\_ID)

FROM ALLOCATION a2

WHERE a2.PROJECT\_ID = p.PROJECT\_ID) AS Number\_of\_Employees

FROM T\_PROJECT p

WHERE EXISTS (

SELECT 1 FROM ALLOCATION a WHERE a.PROJECT\_ID = p.PROJECT\_ID

);

-- Alternative 2: Using LEFT JOIN to include projects with no employees

SELECT p.PROJECT\_NAME,

COUNT(DISTINCT a.EMP\_ID) AS Number\_of\_Employees

FROM T\_PROJECT p

LEFT JOIN ALLOCATION a ON p.PROJECT\_ID = a.PROJECT\_ID

GROUP BY p.PROJECT\_NAME;

-- Q22. Role Name and Number of Employees

-- Original: JOIN with GROUP BY

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;

-- Alternative 1: Using subquery

SELECT r.ROLE\_TITLE,

(SELECT COUNT(DISTINCT a2.EMP\_ID)

FROM ALLOCATION a2

WHERE a2.ROLE\_ID = r.ROLE\_ID) AS Number\_of\_Employees

FROM ROLE r

WHERE EXISTS (

SELECT 1 FROM ALLOCATION a WHERE a.ROLE\_ID = r.ROLE\_ID

);

-- Alternative 2: Using LEFT JOIN to include roles with no employees

SELECT r.ROLE\_TITLE,

COUNT(DISTINCT a.EMP\_ID) AS Number\_of\_Employees

FROM ROLE r

LEFT JOIN ALLOCATION a ON r.ROLE\_ID = a.ROLE\_ID

GROUP BY r.ROLE\_TITLE;

-- Q23. Emp Name and Number of Projects

-- Original: JOIN with GROUP BY

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;

-- Alternative 1: Using subquery

SELECT e.EMP\_NAME,

(SELECT COUNT(DISTINCT a2.PROJECT\_ID)

FROM ALLOCATION a2

WHERE a2.EMP\_ID = e.EMP\_ID) AS Number\_of\_Projects

FROM EMPLOYEE e

WHERE EXISTS (

SELECT 1 FROM ALLOCATION a WHERE a.EMP\_ID = e.EMP\_ID

);

-- Alternative 2: Using LEFT JOIN to include employees with no projects

SELECT e.EMP\_NAME,

COUNT(DISTINCT a.PROJECT\_ID) AS Number\_of\_Projects

FROM EMPLOYEE e

LEFT JOIN ALLOCATION a ON e.EMP\_ID = a.EMP\_ID

GROUP BY e.EMP\_NAME;

-- Q24. Emp Name and Number of Roles

-- Original: JOIN with GROUP BY

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;

-- Alternative 1: Using subquery

SELECT e.EMP\_NAME,

(SELECT COUNT(DISTINCT a2.ROLE\_ID)

FROM ALLOCATION a2

WHERE a2.EMP\_ID = e.EMP\_ID) AS Number\_of\_Roles

FROM EMPLOYEE e

WHERE EXISTS (

SELECT 1 FROM ALLOCATION a WHERE a.EMP\_ID = e.EMP\_ID

);

-- Alternative 2: Using LEFT JOIN to include employees with no roles

SELECT e.EMP\_NAME,

COUNT(DISTINCT a.ROLE\_ID) AS Number\_of\_Roles

FROM EMPLOYEE e

LEFT JOIN ALLOCATION a ON e.EMP\_ID = a.EMP\_ID

GROUP BY e.EMP\_NAME;

-- Q25. Role Name and Number of Employees

-- Original: JOIN with GROUP BY

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;

-- Alternative 1: Using subquery (same as Q22 alternative 1)

-- Alternative 2: Using LEFT JOIN (same as Q22 alternative 2)

-- Q26. Role Name and Number of Projects

-- Original: JOIN with GROUP BY

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;

-- Alternative 1: Using subquery

SELECT r.ROLE\_TITLE,

(SELECT COUNT(DISTINCT a2.PROJECT\_ID)

FROM ALLOCATION a2

WHERE a2.ROLE\_ID = r.ROLE\_ID) AS Number\_of\_Projects

FROM ROLE r

WHERE EXISTS (

SELECT 1 FROM ALLOCATION a WHERE a.ROLE\_ID = r.ROLE\_ID

);

-- Alternative 2: Using LEFT JOIN to include roles with no projects

SELECT r.ROLE\_TITLE,

COUNT(DISTINCT a.PROJECT\_ID) AS Number\_of\_Projects

FROM ROLE r

LEFT JOIN ALLOCATION a ON r.ROLE\_ID = a.ROLE\_ID

GROUP BY r.ROLE\_TITLE;

-- Q27. Emp Name, Role Name and Number of Projects

-- Original: JOIN with GROUP BY

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;

-- Alternative 1: Using subquery

SELECT e.EMP\_NAME, r.ROLE\_TITLE,

(SELECT COUNT(DISTINCT a2.PROJECT\_ID)

FROM ALLOCATION a2

WHERE a2.EMP\_ID = e.EMP\_ID

AND a2.ROLE\_ID = r.ROLE\_ID) AS Number\_of\_Projects

FROM EMPLOYEE e

CROSS JOIN ROLE r

WHERE EXISTS (

SELECT 1 FROM ALLOCATION a

WHERE a.EMP\_ID = e.EMP\_ID AND a.ROLE\_ID = r.ROLE\_ID

);

-- Alternative 2: Using LEFT JOIN to include all combinations

SELECT e.EMP\_NAME, r.ROLE\_TITLE,

COUNT(DISTINCT a.PROJECT\_ID) AS Number\_of\_Projects

FROM EMPLOYEE e

CROSS JOIN ROLE r

LEFT JOIN ALLOCATION a ON e.EMP\_ID = a.EMP\_ID AND r.ROLE\_ID = a.ROLE\_ID

GROUP BY e.EMP\_NAME, r.ROLE\_TITLE;

-- Q28. Project Name, Role Name, Number of Employees

-- Original: JOIN with GROUP BY

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;

-- Alternative 1: Using subquery

SELECT p.PROJECT\_NAME, r.ROLE\_TITLE,

(SELECT COUNT(DISTINCT a2.EMP\_ID)

FROM ALLOCATION a2

WHERE a2.PROJECT\_ID = p.PROJECT\_ID

AND a2.ROLE\_ID = r.ROLE\_ID) AS Number\_of\_Employees

FROM T\_PROJECT p

CROSS JOIN ROLE r

WHERE EXISTS (

SELECT 1 FROM ALLOCATION a

WHERE a.PROJECT\_ID = p.PROJECT\_ID AND a.ROLE\_ID = r.ROLE\_ID

);

-- Alternative 2: Using NOT IN (showing projects/roles with no employees)

SELECT p.PROJECT\_NAME, r.ROLE\_TITLE,

COUNT(DISTINCT a.EMP\_ID) AS Number\_of\_Employees

FROM T\_PROJECT p

CROSS JOIN ROLE r

LEFT JOIN ALLOCATION a ON p.PROJECT\_ID = a.PROJECT\_ID AND r.ROLE\_ID = a.ROLE\_ID

GROUP BY p.PROJECT\_NAME, r.ROLE\_TITLE;

-- Q29. Role Name, Emp Name, Number of Projects

-- Original: JOIN with GROUP BY

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;

-- Alternative 1: Using correlated subquery

SELECT r.ROLE\_TITLE, e.EMP\_NAME,

(SELECT COUNT(DISTINCT a2.PROJECT\_ID)

FROM ALLOCATION a2

WHERE a2.EMP\_ID = e.EMP\_ID

AND a2.ROLE\_ID = r.ROLE\_ID) AS Number\_of\_Projects

FROM EMPLOYEE e

CROSS JOIN ROLE r

WHERE EXISTS (

SELECT 1 FROM ALLOCATION a

WHERE a.EMP\_ID = e.EMP\_ID AND a.ROLE\_ID = r.ROLE\_ID

);

-- Q30. Dept Id, Number of Employees

-- Original: GROUP BY

SELECT e.DEPT\_ID,

COUNT(e.EMP\_ID) AS Number\_of\_Employees

FROM EMPLOYEE e

GROUP BY e.DEPT\_ID;

-- Alternative 1: Using subquery

SELECT d.DEPT\_ID,

(SELECT COUNT(\*) FROM EMPLOYEE e WHERE e.DEPT\_ID = d.DEPT\_ID) AS Number\_of\_Employees

FROM DEPARTMENT d

WHERE EXISTS (SELECT 1 FROM EMPLOYEE e WHERE e.DEPT\_ID = d.DEPT\_ID);

-- Alternative 2: Using LEFT JOIN to include empty departments

SELECT d.DEPT\_ID,

COUNT(e.EMP\_ID) AS Number\_of\_Employees

FROM DEPARTMENT d

LEFT JOIN EMPLOYEE e ON d.DEPT\_ID = e.DEPT\_ID

GROUP BY d.DEPT\_ID;

-- Q31. Mgr\_id, Number of Employees

-- Original: GROUP BY with filter

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;

-- Alternative 1: Using correlated subquery

SELECT m.EMP\_ID AS MGR\_ID,

(SELECT COUNT(\*) FROM EMPLOYEE e WHERE e.MGR\_ID = m.EMP\_ID) AS Number\_of\_Employees

FROM EMPLOYEE m

WHERE m.EMP\_ID IN (SELECT DISTINCT MGR\_ID FROM EMPLOYEE WHERE MGR\_ID IS NOT NULL);

-- Q32. Emp Name, Role Name, Project Name

-- Original: Simple JOIN

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;

-- Alternative 1: Using subqueries (less efficient, just for demonstration)

SELECT

(SELECT EMP\_NAME FROM EMPLOYEE WHERE EMP\_ID = a.EMP\_ID) AS EMP\_NAME,

(SELECT ROLE\_TITLE FROM ROLE WHERE ROLE\_ID = a.ROLE\_ID) AS ROLE\_TITLE,

(SELECT PROJECT\_NAME FROM T\_PROJECT WHERE PROJECT\_ID = a.PROJECT\_ID) AS PROJECT\_NAME

FROM ALLOCATION a;

-- Q33. Project ID, Emp ID, Total Amount collected (sorted descending)

-- Original: GROUP BY with calculation

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;

-- Alternative 1: Using window function

SELECT DISTINCT a.PROJECT\_ID, a.EMP\_ID,

SUM(a.AMOUNT\_PER\_DAY \* DATEDIFF(a.TO\_DATE, a.FROM\_DATE))

OVER (PARTITION BY a.PROJECT\_ID, a.EMP\_ID) AS TOTAL\_AMOUNT\_COLLECTED

FROM ALLOCATION a

ORDER BY TOTAL\_AMOUNT\_COLLECTED DESC;

-- Q34. Emp ID, Role ID, Total Amount Collected

-- Original: GROUP BY with calculation

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;

-- Alternative 1: Using correlated subquery

SELECT DISTINCT a.EMP\_ID, a.ROLE\_ID,

(SELECT SUM(a2.AMOUNT\_PER\_DAY \* DATEDIFF(a2.TO\_DATE, a2.FROM\_DATE))

FROM ALLOCATION a2

WHERE a2.EMP\_ID = a.EMP\_ID AND a2.ROLE\_ID = a.ROLE\_ID) AS Total\_Amount\_Collected

FROM ALLOCATION a;

-- Q35. Emp ID, Role ID, Project ID, Total Amount collected (sorted descending)

-- Original: GROUP BY with calculation

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;

-- Alternative 1: Using window function

SELECT DISTINCT a.EMP\_ID, a.ROLE\_ID, a.PROJECT\_ID,

SUM(a.AMOUNT\_PER\_DAY \* DATEDIFF(a.TO\_DATE, a.FROM\_DATE))

OVER (PARTITION BY a.EMP\_ID, a.ROLE\_ID, a.PROJECT\_ID) AS TOTAL\_AMOUNT\_COLLECTED

FROM ALLOCATION a

ORDER BY TOTAL\_AMOUNT\_COLLECTED DESC;

-- Q36. Emp ID, Mgr ID, Comments

-- Original: CASE expression

SELECT EMP\_ID, MGR\_ID,

CASE

WHEN MGR\_ID IS NULL THEN 'No Manager'

ELSE 'Has Manager'

END AS Comments

FROM EMPLOYEE;

-- Alternative 1: Using COALESCE

SELECT EMP\_ID, MGR\_ID,

COALESCE('Has Manager', 'No Manager') AS Comments

FROM EMPLOYEE;

-- Q37. "Ram works for Ashok" relationship

-- Original: Self JOIN

SELECT e.EMP\_NAME || ' works for ' || m.EMP\_NAME AS Relation

FROM EMPLOYEE e

JOIN EMPLOYEE m ON e.MGR\_ID = m.EMP\_ID;

-- Alternative 1: Using subquery

SELECT e.EMP\_NAME || ' works for ' ||

(SELECT m.EMP\_NAME FROM EMPLOYEE m WHERE m.EMP\_ID = e.MGR\_ID) AS Relation

FROM EMPLOYEE e

WHERE e.MGR\_ID IS NOT NULL;

-- Q38. Employees earning more than their managers

-- Original: Self JOIN

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;

-- Alternative 1: Using EXISTS

SELECT e.EMP\_ID, e.EMP\_NAME, e.SALARY AS EMP\_SALARY

FROM EMPLOYEE e

WHERE EXISTS (

SELECT 1 FROM EMPLOYEE m

WHERE m.EMP\_ID = e.MGR\_ID AND e.SALARY > m.SALARY

);

-- Q39. Managers who joined after their subordinates

-- Original: Self JOIN

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;

-- Alternative 1: Using IN

SELECT e.EMP\_ID, e.EMP\_NAME, e.HIRE\_DATE

FROM EMPLOYEE e

WHERE e.MGR\_ID IN (

SELECT m.EMP\_ID FROM EMPLOYEE m

WHERE m.EMP\_ID = e.MGR\_ID AND e.HIRE\_DATE < m.HIRE\_DATE

);

-- Q40. Employees earning more than avg salary of their department

-- Original: WITH clause (CTE)

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);

-- Alternative 1: Using JOIN

SELECT e.EMP\_ID, e.EMP\_NAME, e.DEPT\_ID, e.SALARY

FROM EMPLOYEE e

JOIN (

SELECT DEPT\_ID, AVG(SALARY) AS AVG\_SAL

FROM EMPLOYEE

GROUP BY DEPT\_ID

) d ON e.DEPT\_ID = d.DEPT\_ID

WHERE e.SALARY > d.AVG\_SAL;

-- Q41. Employees who have changed their roles at least twice

-- Original: 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;

-- Alternative 1: Using GROUP BY and HAVING

SELECT e.EMP\_ID, e.EMP\_NAME

FROM EMPLOYEE e

JOIN ALLOCATION a ON e.EMP\_ID = a.EMP\_ID

GROUP BY e.EMP\_ID, e.EMP\_NAME

HAVING COUNT(DISTINCT a.ROLE\_ID) >= 3;

-- Q42. Departments with no employees

-- Original: LEFT JOIN

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;

-- Alternative 1: Using NOT EXISTS

SELECT d.DEPT\_ID, d.DEPT\_NAME

FROM DEPARTMENT d

WHERE NOT EXISTS (

SELECT 1 FROM EMPLOYEE e WHERE e.DEPT\_ID = d.DEPT\_ID

);

-- Alternative 2: Using NOT IN

SELECT d.DEPT\_ID, d.DEPT\_NAME

FROM DEPARTMENT d

WHERE d.DEPT\_ID NOT IN (SELECT DISTINCT DEPT\_ID FROM EMPLOYEE);

-- Q43. Departments with at least one employee

-- Original: JOIN

SELECT DISTINCT d.DEPT\_ID, d.DEPT\_NAME

FROM DEPARTMENT d

JOIN EMPLOYEE e ON d.DEPT\_ID = e.DEPT\_ID;

-- Alternative 1: Using EXISTS

SELECT d.DEPT\_ID, d.DEPT\_NAME

FROM DEPARTMENT d

WHERE EXISTS (SELECT 1 FROM EMPLOYEE e WHERE e.DEPT\_ID = d.DEPT\_ID);

-- Q44. Project ID, Role ID, No of Employees (with ROLLUP)

-- Original: WITH ROLLUP

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;

-- Alternative 1: Using UNION ALL to simulate ROLLUP

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

UNION ALL

SELECT a.PROJECT\_ID, NULL, COUNT(DISTINCT a.EMP\_ID)

FROM ALLOCATION a

GROUP BY a.PROJECT\_ID

UNION ALL

SELECT NULL, NULL, COUNT(DISTINCT EMP\_ID)

FROM ALLOCATION;

-- Q45. Employee ID, Project ID, Total salary (with ROLLUP)

-- Original: WITH ROLLUP

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;

-- Q46. Hierarchical report starting with 'Raja'

-- Original: Recursive CTE

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;

-- Q47. Classify employees based on number of skills

-- Original: CASE expression

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;

-- Alternative 1: Using derived table

SELECT emp.EMP\_ID, emp.Number\_of\_Skills,

CASE

WHEN emp.Number\_of\_Skills > 5 THEN 'Major Resource'

WHEN emp.Number\_of\_Skills > 3 THEN 'Useful Resource'

WHEN emp.Number\_of\_Skills > 1 THEN 'Resource'

ELSE 'Needs Training'

END AS Description

FROM (

SELECT e.EMP\_ID, COUNT(s.SKILL\_ID) AS Number\_of\_Skills

FROM EMPLOYEE e

LEFT JOIN SKILLS s ON e.EMP\_ID = s.EMP\_ID

GROUP BY e.EMP\_ID

) emp;

-- Q48. Leave description based on number of leaves

-- Original: CASE expression

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;

-- Q49. Top 5 salaried employees using RANK

-- Original: RANK window function

SELECT \*

FROM (

SELECT \*,

RANK() OVER (ORDER BY SALARY DESC) AS rnk

FROM EMPLOYEE

) ranked\_employees

WHERE rnk <= 5;

-- Alternative 1: Using DENSE\_RANK

SELECT \*

FROM (

SELECT \*,

DENSE\_RANK() OVER (ORDER BY SALARY DESC) AS rnk

FROM EMPLOYEE

) ranked\_employees

WHERE rnk <= 5;

-- Q50. Top 3 Departments by employee count

-- Original: RANK window function

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;

-- Alternative 1: Using DENSE\_RANK

SELECT \*

FROM (

SELECT d.DEPT\_ID,

COUNT(e.EMP\_ID) AS No\_of\_Employees,

DENSE\_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;

-- Q51. 2nd max salary per department

-- Original: RANK window function

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;

-- Alternative 1: Using DENSE\_RANK

SELECT \*

FROM (

SELECT DEPT\_ID, EMP\_ID, SALARY,

DENSE\_RANK() OVER (PARTITION BY DEPT\_ID ORDER BY SALARY DESC) AS rnk

FROM EMPLOYEE

) ranked

WHERE rnk = 2;

-- Q52. Emp Name, Number of skills

-- Original: LEFT JOIN with GROUP BY

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;

-- Alternative 1: Using correlated subquery

SELECT e.EMP\_NAME,

(SELECT COUNT(\*) FROM EMPLOYEESKILL es WHERE es.EMP\_ID = e.EMP\_ID) AS Number\_of\_Skills

FROM EMPLOYEE e;

-- Q53. Emp Name, Number of recharges

-- Original: LEFT JOIN with GROUP BY

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;

-- Alternative 1: Using correlated subquery

SELECT e.EMP\_NAME,

(SELECT COUNT(\*) FROM MOBILERECHARGE m WHERE m.MOBILE\_NO = e.MOBILE\_NO) AS Number\_of\_Recharges

FROM EMPLOYEE e;

-- Q54. Delete duplicate rows from Employee table

-- Original: 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

);

-- Alternative 1: Using EXISTS

DELETE FROM EMPLOYEE e1

WHERE EXISTS (

SELECT 1 FROM EMPLOYEE e2

WHERE e2.EMP\_ID = e1.EMP\_ID

AND e2.ROWID < e1.ROWID

);