**SQL QUERIES:**

You're working with a company that manages projects across different departments. They want to analyze project performance, employee contributions, and department efficiency. Your task is to write a SQL query that:

* Ranks employees within each department based on the number of projects they've completed.
* Calculates the average project duration for each department.
* Identifies projects that took longer than the department's average.
* Finds the top 3 most efficient employees in each department (based on average project duration).
* Compares each project's duration to the previous project in the same department.

***Answer:***

WITH ProjectStats AS (

-- Calculate project duration and count

SELECT

p.ProjectID,

p.ProjectName,

p.DepartmentID,

p.EmployeeID,

e.EmployeeName,

d.DepartmentName,

DATEDIFF(p.EndDate, p.StartDate) AS ProjectDuration, -- Corrected the order of StartDate and EndDate

COUNT(\*) OVER (PARTITION BY p.EmployeeID) AS ProjectCount,

AVG(DATEDIFF(p.EndDate, p.StartDate)) OVER (PARTITION BY p.DepartmentID) AS AvgDeptDuration

FROM

Projects p

JOIN Employees e ON p.EmployeeID = e.EmployeeID

JOIN Departments d ON p.DepartmentID = d.DepartmentID

),

RankedEmployees AS (

-- Rank employees within departments based on project count

SELECT

\*,

RANK() OVER (PARTITION BY DepartmentID ORDER BY ProjectCount DESC) AS EmployeeRank,

DENSE\_RANK() OVER (PARTITION BY DepartmentID ORDER BY ProjectDuration) AS EfficiencyRank

FROM

ProjectStats

),

ProjectComparison AS (

-- Compare project duration to previous project in the same department

SELECT

p.ProjectID,

p.ProjectName,

p.DepartmentID,

p.EmployeeID,

DATEDIFF(p.EndDate, p.StartDate) AS ProjectDuration, -- Recalculating ProjectDuration here for the comparison

LAG(DATEDIFF(p.EndDate, p.StartDate)) OVER (PARTITION BY p.DepartmentID ORDER BY p.StartDate) AS PrevProjectDuration,

DATEDIFF(p.EndDate, p.StartDate) - LAG(DATEDIFF(p.EndDate, p.StartDate)) OVER (PARTITION BY p.DepartmentID ORDER BY p.StartDate) AS DurationDifference

FROM

Projects p

)

SELECT

re.DepartmentName,

re.EmployeeName,

re.ProjectCount,

re.EmployeeRank,

re.EfficiencyRank,

p.ProjectName,

p.ProjectDuration,

p.AvgDeptDuration,

CASE

WHEN p.ProjectDuration > p.AvgDeptDuration THEN 'Above Average'

WHEN p.ProjectDuration < p.AvgDeptDuration THEN 'Below Average'

ELSE 'Average'

END AS DurationComparison,

pc.PrevProjectDuration,

pc.DurationDifference

FROM

RankedEmployees re

JOIN ProjectStats p ON re.ProjectID = p.ProjectID

JOIN ProjectComparison pc ON p.ProjectID = pc.ProjectID

WHERE

-- Top 3 most efficient employees

re.EfficiencyRank <= 3

ORDER BY

re.DepartmentName,

re.EfficiencyRank,

p.ProjectDuration;

You are working with a retail company that wants to analyze its sales data across different departments. They have provided you with a table containing employee sales information. Your task is to write a SQL query that accomplishes the following:

* Calculate the total sales, average sales, and number of employees for each department.
* Rank the departments based on their total sales.
* Display this information in a single result set, ordered by total sales descending.

***Answer1:***

– – For the days only on which sales have happened

select

department,

sum(sales\_amount) as total\_sales,

sum(sales\_amount)/ count(distinct sales\_date) as avg\_daily\_sales,

count(employee\_id) as employee\_count,

rank() over(order by sum(sales\_amount) desc) as ranking

from employee\_sales

group by department;

***Answer2:***

– – For all the days from start date to end date

select

department,

sum(sales\_amount) as total\_sales,

sum(sales\_amount)/ datediff(max(sales\_date),min(sales\_date)) as avg\_daily\_sales,

count(employee\_id) as employee\_count,

rank() over(order by sum(sales\_amount) desc) as ranking

from employee\_sales

group by department;