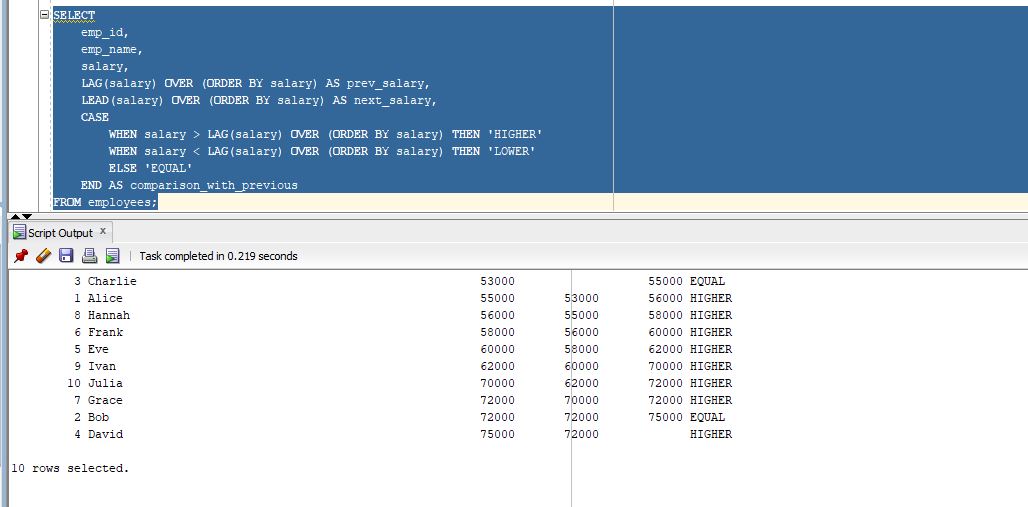
26331 Habanzintwali Ihirwe Kelia.

26017 Tesi Divine.

**PL/SQL ASSIGNMENT.**

1. Comparing values with Previous and Next



This SQL query compares each employee's current salary to the **previous** and **next** salaries in the sorted list (based on salary), and then tells you if the current salary is higher, lower or equal to the previous salary.

It uses LAG(salary) to get the previous salary and LEAD(salary)to get the next salary.

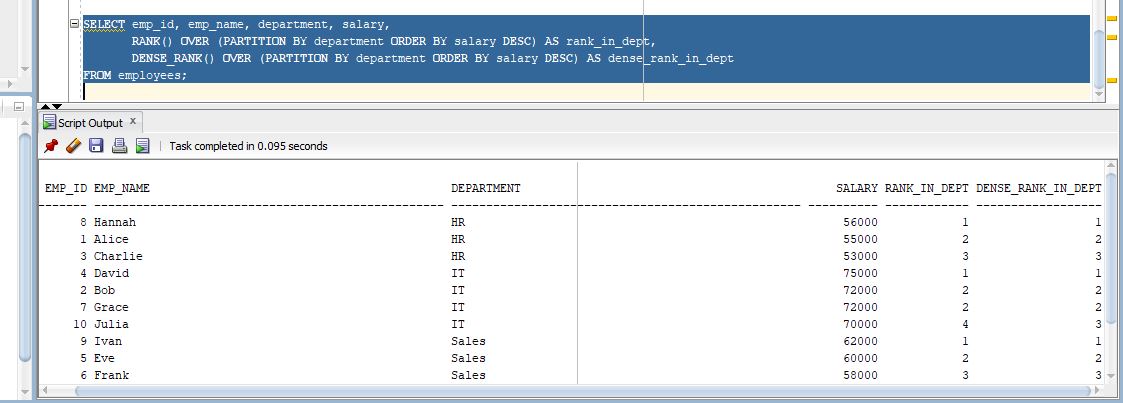
Helps HR analyze salary growth trends.

Real-Life Use:

Compare monthly sales, student scores, stock prices, etc.

this is for Compare Values with Previous or Next Records (LAG/LEAD)

1. Ranking Data within a Category



This query ranks employees within each department based on their salary (highest to lowest) using two different functions:

**RANK()** – gives the same rank to ties but skips the next rank(s).

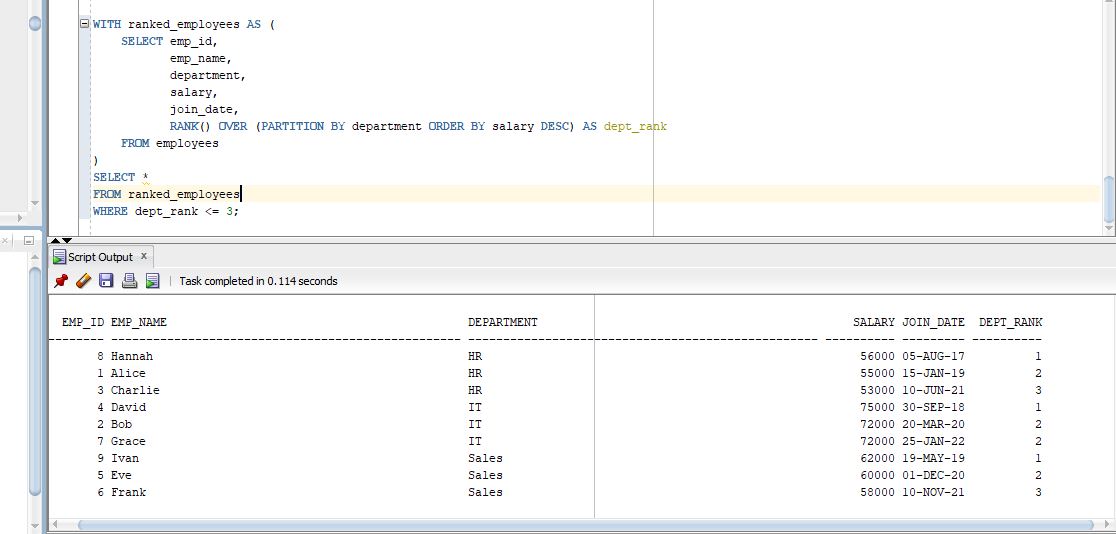
**DENSE\_RANK()** – gives the same rank to ties but doesn’t skip the next number.

Useful for fair evaluations, bonuses, etc.

Real-Life Use:

Rank employees in departments, products in categories, students in classes. this is for Ranking Data within a Category (RANK vs. DENSE\_RANK)

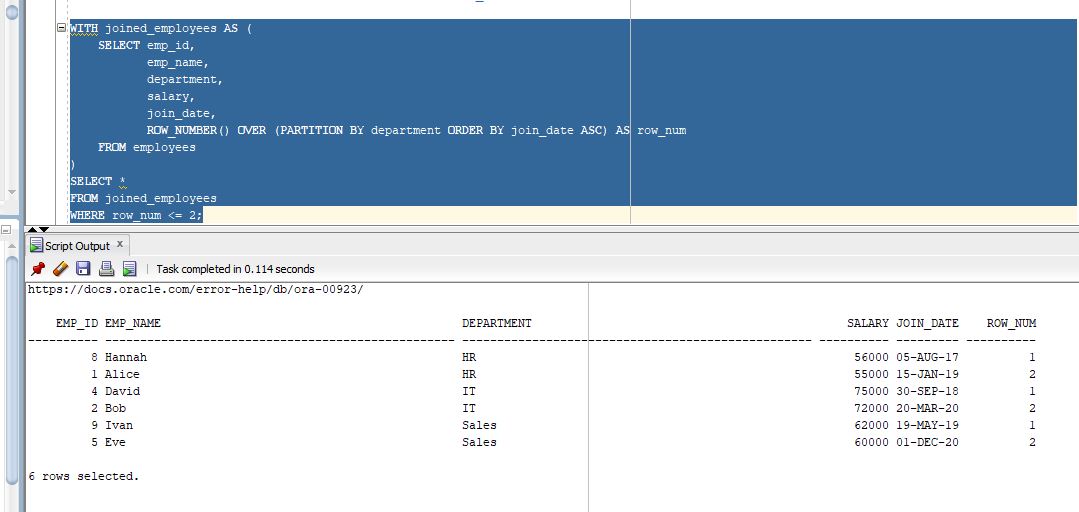
1. Identifying Top Records



This SQL query selects the top 3 highest-paid employees within each department using the RANK() function.

* RANK() OVER (PARTITION BY department ORDER BY salary DESC) assigns a rank per department, based on salary in descending order.
* Only rows where dept\_rank <= 3 are selected (i.e., top 3 earners per department).

1. Finding the Earliest Records

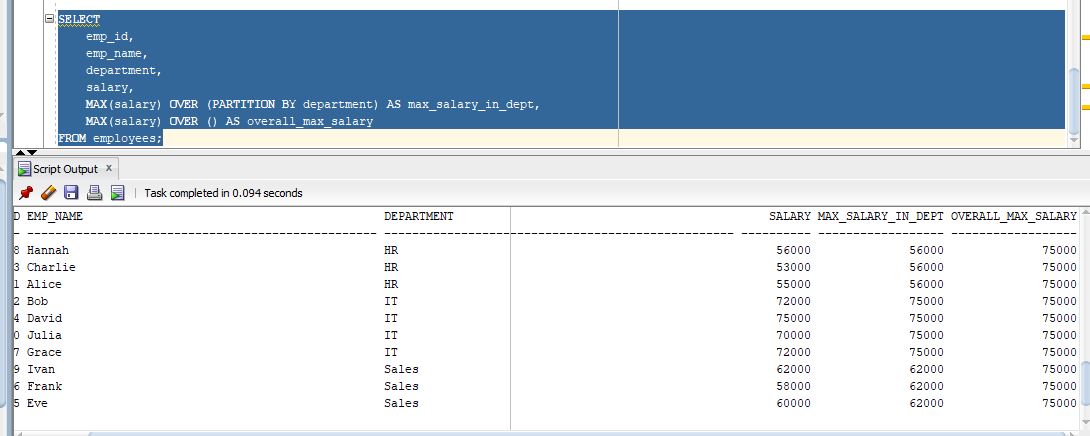


This query assigns a unique row number to each employee within their department based on their joining date (earliest first). It then selects only the employees with row numbers 1 or 2 — meaning the two employees who joined first in each department.

Real-Life example:

Imagine you’re an HR manager and you want to reward the first two employees who joined each department for their loyalty. This query helps you quickly find them.

1. Aggregation with Window Functions.



This query finds the maximum salary within each department (MAX() OVER (PARTITION BY department)) and the overall maximum salary across the entire company (MAX() OVER ()) for each employee.

Real-Life Example:

Suppose you're preparing a salary benchmarking report. You want to know how each employee’s salary compares to the highest salary in their department and the highest salary overall in the company.