✅ Oracle SQL Solutions (Q16 – Q39)

Q16. Employees working in more than one project?

SQL Sol:

```
SELECT emp_id
FROM employee_projects
GROUP BY emp_id
HAVING COUNT(DISTINCT project_id) > 1;
```

Alternative (show project list):

```
SELECT emp_id, LISTAGG(project_id, ', ') WITHIN GROUP (ORDER BY project_id) AS projects
FROM employee_projects
GROUP BY emp_id
HAVING COUNT(DISTINCT project_id) > 1
```

Q17. Difference between INNER JOIN and LEFT JOIN

SQL Solu:

INNER JOIN → only matches:

```
SELECT e.emp_name, d.dept_name
FROM employees e
INNER JOIN departments d ON e.dept_id = d.dept_id;
```

LEFT JOIN → all left rows + matching right:

```
SELECT e.emp_name, d.dept_name
FROM employees e
LEFT JOIN departments d ON e.dept_id = d.dept_id;
```

Q18. Products ordered by more than 5 unique customers

SQL Solu:

```
SELECT product_id
FROM order_tbl
GROUP BY product_id
HAVING COUNT(DISTINCT cust_id) > 5;
```

Alternative (≥5 instead of >5):

```
HAVING COUNT(DISTINCT cust_id) >= 5;
```

Q19. Retrieve Employees who never made a sale (anti join)?

SQL Solu:

```
SELECT e.emp_id, e.emp_name
FROM employees e
LEFT JOIN sales_tbl s ON e.emp_id = s.emp_id
WHERE s.emp_id IS NULL;
```

Alternative (NOT IN):

```
SELECT emp_id, emp_name
FROM employees
WHERE emp_id NOT IN (SELECT emp_id FROM sales_tbl);
```

Q20. Write a query using CROSS JOIN?

SQL Solu:

```
SELECT e.emp_name, d.dept_name
FROM employees e
CROSS JOIN departments d;
```

Q21. Find the Second highest salary?

SQL Solu:

```
SELECT MAX(salary) AS second_highest
FROM employees
WHERE salary < (SELECT MAX(salary) FROM employees);
```

Using DENSE_RANK():

```
SELECT salary
FROM (
   SELECT salary,
      DENSE_RANK() OVER (ORDER BY salary DESC) AS rnk
   FROM employees
) t
WHERE rnk = 2;
```

Q22.Get Nth highest salary (N=3)?

SQL Solu:

```
SELECT salary
FROM (
  SELECT salary, DENSE_RANK() OVER (ORDER BY salary DESC) rnk
  FROM employees
)
WHERE rnk = :N;
```

Q23. List Departments where avg salary > 60,000?

SQL Solu:

```
SELECT dept_id, AVG(salary) AS avg_sal
FROM employees
GROUP BY dept_id
HAVING AVG(salary) > 60000;
```

Q24. Find Employees earning more than department average?

SQL Sol:

```
SELECT e.emp_id, e.emp_name, e.salary
FROM employees e
JOIN (
   SELECT dept_id, AVG(salary) AS avg_sal
   FROM employees
   GROUP BY dept_id
) d ON e.dept_id = d.dept_id
WHERE e.salary > d.avg_sal;
```

Alternative (Analytic):

```
SELECT emp_id, emp_name, salary
FROM (
   SELECT e.*, AVG(salary) OVER (PARTITION BY dept_id) dept_avg
   FROM employees e
)
WHERE salary > dept_avg
order by emp_id, emp_name, salary desc;
```

Q25. Write a Query to Calculate Running total of sales?

SQL Solu:

```
SELECT sale_id, emp_id, sale_amt,
     SUM(sale_amt) OVER (ORDER BY sale_date) AS running_total
FROM sales_tbl;
```

Q26. Find Employees whose salary is above company Average?

SQL Solu:

```
SELECT emp_id, emp_name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

Alternative (Analytic):

```
SELECT emp_id, emp_name, salary
FROM (
   SELECT e.*, AVG(salary) OVER() comp_avg
   FROM employees e
)
WHERE salary > comp_avg;
```

Q27. Get Employees with the same salary?

SQL Solu:

```
SELECT e1.emp_id, e1.emp_name, e1.salary
FROM employees e1
JOIN employees e2
ON e1.salary = e2.salary AND e1.emp_id <> e2.emp_id;
```

Alternative (Grouped):
If you want to see them grouped together by salary
(to avoid duplicates)
If you only want unique groups of employees with same salary, use GROUP BY:
Use this query instead:

```
SELECT salary,
      LISTAGG(emp_name, ', ') WITHIN GROUP (ORDER BY emp_name) AS
employees
FROM employees
GROUP BY salary
HAVING COUNT(*) > 1;
```

Q28. Find Employees who earn more than their manager?

SQL Solu:

```
SELECT e.emp_id, e.emp_name, e.salary, m.emp_name AS manager, m.salary AS
mgr_salary
FROM employees e
JOIN employees m ON e.manager_id = m.emp_id
WHERE e.salary > m.salary;
```

Q29. Get the Difference between max & min salary in each department?

SQL Solu:

```
SELECT dept_id,
      MAX(salary) - MIN(salary) AS diff
FROM employees
GROUP BY dept_id;
```

Q30. Find Employees whose salary is in top 10% overall?

SQL Solu:

Using NTILE (simple):

```
SELECT emp_id, emp_name, salary
FROM (
  SELECT emp_id, emp_name, salary,
        NTILE(10) OVER (ORDER BY salary DESC) decile
  FROM employees
)
WHERE decile = 1;
```

Alternative 1 (PERCENT_RANK):

```
SELECT emp_id, emp_name, salary
FROM (
  SELECT emp_id, emp_name, salary,
        PERCENT_RANK() OVER (ORDER BY salary DESC) perc
  FROM employees
)
WHERE perc <= 0.10;
```

Alternative 2 (PERCENTILE_CONT):

```
SELECT emp_id, emp_name, salary
FROM employees
WHERE salary >= (
   SELECT PERCENTILE_CONT(0.90) WITHIN GROUP (ORDER BY salary)
   FROM employees
);
```

Alternative 3: Using ROWNUM + Ordered Subquery:

```
SELECT emp_id, emp_name, salary
FROM (
   SELECT e.*, ROWNUM AS rn
   FROM (
      SELECT emp_id, emp_name, salary
      FROM employees
      ORDER BY salary DESC
   ) e
)
WHERE rn <= (SELECT CEIL(COUNT(*) * 0.10) FROM employees);
```

Alternative 4: Using ROW_NUMBER() Window Function:

```
SELECT emp_id, emp_name, salary
FROM (
   SELECT emp_id, emp_name, salary,
        ROW_NUMBER() OVER (ORDER BY salary DESC) AS rn,
        COUNT(*) OVER () AS total_count
   FROM employees
)
WHERE rn <= CEIL(total_count * 0.10);
```

Q31. Write a query to Rank employees by salary within each department?

SQL Solu:

```
SELECT emp_id, emp_name, dept_id, salary,
     RANK() OVER (PARTITION BY dept_id ORDER BY salary DESC) AS rnk
FROM employees;
```

Q32.Find the First & last order date of each customer?

SQL Solu:

```
SELECT cust_id,
      MIN(order_date) AS first_order,
      MAX(order_date) AS last_order
FROM order_tbl
GROUP BY cust_id;
```

Q33.Calculate 7-day moving average of sales?

SQL Solu:

```
SELECT sale_date,
      ROUND(AVG(sale_amt) OVER (
         ORDER BY sale_date
         ROWS BETWEEN 6 PRECEDING AND CURRENT ROW
      )) AS mov_avg
FROM sales_tbl;
```

Option 2: Moving average for last 7 calendar days

```
SELECT s1.sale_date,
      (SELECT ROUND(AVG(s2.sale_amt))
       FROM sales_tbl s2
       WHERE s2.sale_date BETWEEN s1.sale_date - 6 AND s1.sale_date
      ) AS mov_avg
FROM sales_tbl s1
ORDER BY s1.sale_date;
```

Q34. Find Employees whose salary decreased compared to last year?

SQL Solu:

Option 1: Use Only salary_history (without emp_name)

```
SELECT emp_id, salary AS current_salary, prev_salary, effective_date
FROM (
   SELECT emp_id, salary, effective_date,
         LAG(salary) OVER (PARTITION BY emp_id ORDER BY effective_date) AS
prev_salary
   FROM salary_history
)
WHERE prev_salary IS NOT NULL
  AND salary < prev_salary;
```

Option 2: Join With employees Table (to fetch emp_name):

```
SELECT sh.emp_id, e.emp_name, sh.salary AS current_salary, sh.prev_salary,
sh.effective_date
FROM (
   SELECT emp_id, salary, effective_date,
         LAG(salary) OVER (PARTITION BY emp_id ORDER BY effective_date) AS
prev_salary
```

```
   FROM salary_history
) sh
JOIN employees e ON sh.emp_id = e.emp_id
WHERE sh.prev_salary IS NOT NULL
  AND sh.salary < sh.prev_salary;
```

Q35. Identify Customers with 3 consecutive failed transactions?

SQL Sol:

```
SELECT DISTINCT cust_id
FROM (
  SELECT cust_id, order_id, status,
        LAG(status,1) OVER (PARTITION BY cust_id ORDER BY order_date) AS
prev1,
        LAG(status,2) OVER (PARTITION BY cust_id ORDER BY order_date) AS prev2
  FROM order_tbl
)
WHERE status='FAILED' AND prev1='FAILED' AND prev2='FAILED';
```

Q36.Get the Previous & next salary for each employee using LAG and LEAD?

SQL Solu:

```
SELECT emp_id, emp_name, salary,
      LAG(salary)  OVER (ORDER BY salary) AS prev_salary,
      LEAD(salary) OVER (ORDER BY salary) AS next_salary
FROM employees;
```

Q37. Calculate Retention : users who logged in on D1, D7, D30?

SQL Sol:

```
SELECT cust_id
FROM order_tbl
WHERE TRUNC(order_date) IN (TRUNC(SYSDATE), TRUNC(SYSDATE-7),
TRUNC(SYSDATE-30))
GROUP BY cust_id
HAVING COUNT(DISTINCT TRUNC(order_date)) = 3;
```

Alternative (presence checks):

```
SELECT cust_id
FROM order_tbl
WHERE order_date >= TRUNC(SYSDATE-30)  -- last 30 days
GROUP BY cust_id
HAVING SUM(CASE WHEN TRUNC(order_date) = TRUNC(SYSDATE) THEN 1
END) > 0
  AND SUM(CASE WHEN TRUNC(order_date) = TRUNC(SYSDATE-7) THEN 1
END) > 0
```

```
  AND SUM(CASE WHEN TRUNC(order_date) = TRUNC(SYSDATE-30) THEN 1
END) > 0;
```

Q38. Find Running average of employee salaries?

SQL Solu:

Main (rounded):

```
SELECT emp_id, emp_name, salary,
      ROUND(
       AVG(salary) OVER (ORDER BY salary ROWS BETWEEN UNBOUNDED
PRECEDING AND CURRENT ROW),2
      ) AS running_avg
FROM employees;
```

Alternative (manual SUM/COUNT):

```
SELECT emp_id, emp_name, salary,
      ROUND(
       SUM(salary) OVER (ORDER BY salary ROWS BETWEEN UNBOUNDED
PRECEDING AND CURRENT ROW) /
       COUNT(*)  OVER (ORDER BY salary ROWS BETWEEN UNBOUNDED
PRECEDING AND CURRENT ROW),2
      ) AS running_avg
FROM employees;
```

Q39. Find Employees with highest salary in each department using window function not using subqueries?

SQL Solu:

Using Rank():

```
SELECT emp_id, emp_name, dept_id, salary
FROM (
  SELECT emp_id, emp_name, dept_id, salary,
       RANK() OVER (PARTITION BY dept_id ORDER BY salary DESC) AS rnk
  FROM employees
)
WHERE rnk = 1;
```

Alternative (GROUP BY + HAVING):

```
SELECT e.emp_id, e.emp_name, e.dept_id, e.salary
FROM employees e
JOIN (
  SELECT dept_id, MAX(salary) AS max_sal
  FROM employees
  GROUP BY dept_id
) d ON e.dept_id = d.dept_id AND e.salary = d.max_sal;
```

IF any one say by using subquery then use this below:

📌 **Alternative (Correlated Subquery):**

```
SELECT emp_id, emp_name, dept_id, salary
FROM employees e
WHERE salary = (
   SELECT MAX(salary)
   FROM employees
   WHERE dept_id = e.dept_id
);
```

Q40: show rank vs dense_rank difference with example?

SQL Solu:

Both are analytic functions used for ranking rows based on ORDER BY.
The key difference is how they handle ties (duplicate values).

📌 **Example Data (employees table subset):**

```
EMP_ID | EMP_NAME | SALARY
-------+----------+--------
1      | Alice    | 60000
2      | Bob      | 55000
3      | Charlie  | 70000
4      | David    | 55000
5      | Eve      | 90000
6      | Frank    | 40000
```

Using RANK():

```
SELECT emp_id, emp_name, salary,
     RANK() OVER (ORDER BY salary DESC) AS rnk
FROM employees;
```

Output:

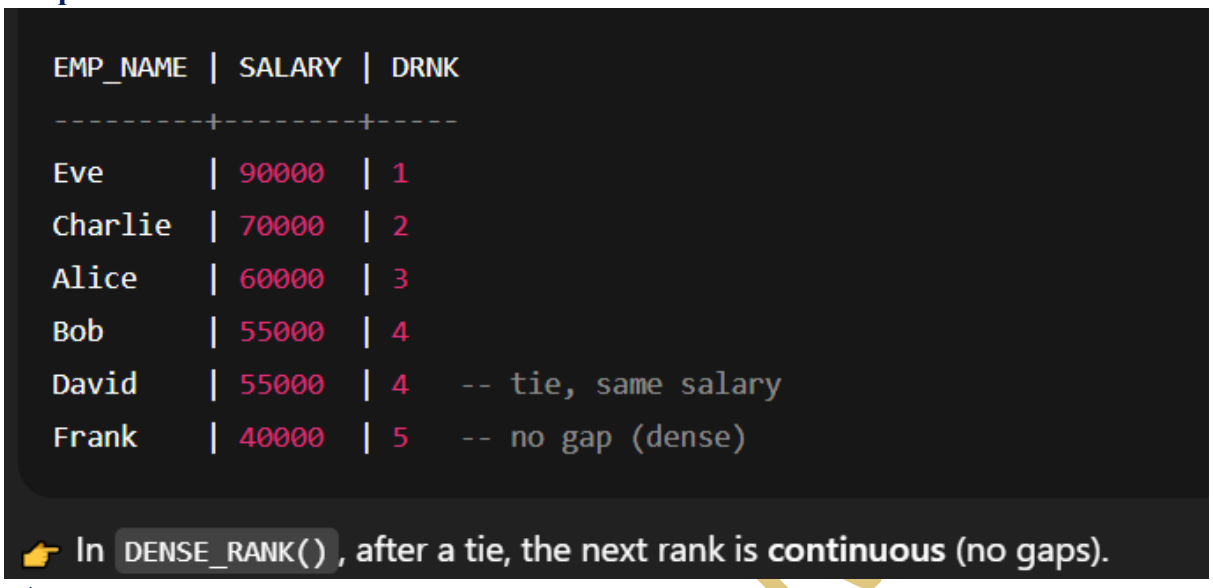
```
EMP_NAME | SALARY | RNK
---------+--------+----
Eve      | 90000  | 1
Charlie  | 70000  | 2
Alice    | 60000  | 3
Bob      | 55000  | 4
David    | 55000  | 4   -- tie, same salary
Frank    | 40000  | 6   -- rank jumps (5 skipped)
```

👉 In `RANK()`, after a tie, the next rank **jumps**. (Here: 4 → 6)

🔷 **Using DENSE_RANK():**

SELECT emp_id, emp_name, salary,
DENSE_RANK() OVER (ORDER BY salary DESC) AS drnk
FROM employees;

Output:



📌 **When to Use**

- **RANK() → when you want to keep original rank positions (good for competition scores, where ties still consume rank numbers).**
- **DENSE_RANK() → when you want continuous numbering (good for top-N salary queries, reporting).**