Session Examples : Oracle Advannced SQL

JOINs

1. INNER JOIN

```
SELECT e.employee_id, e.first_name, d.department_name
FROM employees e
INNER JOIN departments d
ON e.department_id = d.department_id;
```

Returns only employees who belong to a department.

---

2. LEFT OUTER JOIN

```
SELECT e.employee_id, e.first_name, d.department_name
FROM employees e
LEFT OUTER JOIN departments d
ON e.department_id = d.department_id;
```

Shows all employees, including those **without** a department.

---

3. SELF JOIN

```
SELECT e.employee_id, e.first_name AS Employee, m.first_name AS Manager
FROM employees e
JOIN employees m
ON e.manager_id = m.employee_id;
```

Retrieves each employee along with their manager’s name.

Aggregations & GROUP BY

1. Total Salary by Department

```
SELECT department_id, SUM(salary) AS total_salary
FROM employees
GROUP BY department_id;
```

---

2. Average Salary by Job Title

```sql
SELECT job_id, ROUND(AVG(salary), 2) AS avg_salary
FROM employees
GROUP BY job_id;
```

---

3. Count of Employees per Department (with HAVING)

```sql
SELECT department_id, COUNT(*) AS emp_count
FROM employees
GROUP BY department_id
HAVING COUNT(*) > 5;
```

Filters only departments with more than 5 employees.

Filtering and Sorting

1. Filtering with WHERE

```sql
SELECT first_name, salary
FROM employees
WHERE salary > 5000;
```

---

2. Multiple Conditions

```sql
SELECT first_name, department_id, hire_date
FROM employees
WHERE department_id = 90
AND hire_date > TO_DATE('01-JAN-2020', 'DD-MON-YYYY');
```

---

3. Sorting Results

```sql
SELECT first_name, salary
FROM employees
ORDER BY salary DESC;
```

Subqueries

1. Simple Subquery in WHERE

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

Finds employees earning above the average salary.

---

2. Subquery in FROM Clause

```
SELECT department_id, avg_salary
FROM (
    SELECT department_id, AVG(salary) AS avg_salary
    FROM employees
    GROUP BY department_id
)
WHERE avg_salary > 8000;
```

Filters departments whose average salary exceeds 8000.

---

3. Correlated Subquery

```
SELECT e.first_name, e.salary
FROM employees e
WHERE e.salary > (
    SELECT AVG(salary)
    FROM employees
    WHERE department_id = e.department_id
);
```

Finds employees earning more than the **average of their department**.

Set Operators

1. UNION

```
SELECT employee_id, first_name FROM employees
WHERE department_id = 10
UNION
SELECT employee_id, first_name FROM employees
WHERE department_id = 20;
```

Combines results from both queries, **removing duplicates**.

---

2. UNION ALL

```sql
SELECT department_id FROM departments
UNION ALL
SELECT department_id FROM employees;
```

Combines results **including duplicates**.

---

3. INTERSECT

```sql
SELECT employee_id FROM employees WHERE department_id = 10
INTERSECT
SELECT employee_id FROM job_history;
```

Returns employees who are **currently** in department 10 **and** have job history records.
