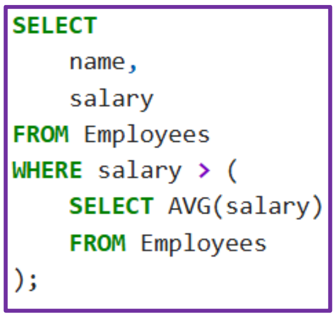
SQL – Scenario based questions

Scenario 1: Employee Salary Analysis

Question:
Find the names and salaries of employees who earn more than the average salary in the company.

Answer:

- First, we need the average salary of all employees.
- So, we need to use **AVG(salary)** to calculate it.
- Then, we must compare each employee's salary with this average salary.
- This requires a subquery because the average salary must be calculated separately.
- Finally, retrieve the name and salary of those employees whose salary is greater than the company's average.



```sql
SELECT
    name,
    salary
FROM Employees
WHERE salary > (
    SELECT AVG(salary)
    FROM Employees
);
```

Scenario 2: Customer Orders without Matching Records

Question:
Retrieve a list of customer names who have not placed any orders.

Answer:

- We have two tables:
  - **Customers** → contains customer details
  - **Orders** → contains order records
- We need to Perform a **LEFT JOIN** between Customers and Orders

- Select only rows where the order record is **NULL** (meaning no order found)

```
SELECT
    c.customer_name
FROM Customers c
LEFT JOIN Orders o
    ON c.customer_id = o.customer_id
WHERE o.customer_id IS NULL;
```

Scenario 3: Product Sales Summary

Question:
Display the total sales amount for each product.

Answer:

- We have a Sales table that contains multiple sales records.
- Each record includes:
    - **product_id (or product_name)**
    - **sales_amount**
- We need to calculate the total sales for each product.
- To do this, SQL must:
    - Group all rows by product
    - Sum the sales amount of each group
- Use **GROUP BY + SUM().**

```
SELECT
    product_id,
    SUM(sales_amount) AS total_sales
FROM Sales
GROUP BY product_id;
```

Scenario 4: Department-Wise Employee Count

Question:
List each department name with the number of employees working in it.

Answer:

- You have an **Employees** table.

- Each employee belongs to a **department** (via department or department_id).
- You need to count **how many employees** are in each department.
- Use:
    - **GROUP BY** department to group employees by their department.
    - **COUNT(*)** to count employees in each group.

```
SELECT
    department,
    COUNT(*) AS employee_count
FROM Employees
GROUP BY department;
```

Scenario 5: Top 3 Highest Sales

Question:
Find the top 3 highest sales transactions.

Answer:

- We have a **Sales** table with multiple transactions.
- Each transaction has a **sales_amount**.
- We need to find the **highest 3 sales amounts**.
- To do that:
    - Sort the records in **descending order** of sales_amount.
    - Pick the **top 3 rows** using LIMIT 3.

```
SELECT *
FROM Sales
ORDER BY sales_amount DESC
LIMIT 3;
```

Scenario 6: Calculate Employee Salary Ranks by Department

Question:
Write a query to display each employee's name, department name, salary, and their salary rank within their respective department.

Answer:

- We need to display:
  - employee name
  - department name
  - salary
  - salary rank *inside that department*
- Employees must be grouped **department-wise** for ranking.
- Within each department, employees should be ranked based on salary.
  - Higher salary = better rank (rank 1 is highest salary).
- To assign ranks inside groups, we use a **window function**:
  - **RANK() or DENSE_RANK() or ROW_NUMBER()**
  - Here, **RANK()** is preferred.
- Use partitioning:
  - **PARTITION BY** department → creates separate ranking groups for each department.
  - **ORDER BY** salary **DESC** → higher salaries get rank 1.

```sql
SELECT
    name,
    department,
    salary,
    RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS salary_rank
FROM Employees;
```