**SQL Project: Employee Management System Analysis**

This project will simulate a **real-life employee management system analysis** for a fictional company using the **employee database** you have. The goal is to design queries that could help **HR managers** and **senior management** make decisions about employee salaries, tenure, department structure, and performance insights.

**Project Title**

**"HR Insights: Employee Performance and Department Analysis"**

**Project Objectives**

By the end of this project, you will:

1. Retrieve and analyze employee data to gain actionable insights.
2. Use various SQL concepts: **SELECT**, **JOINs**, **WHERE** **GROUP BY**, **HAVING**, **subqueries**, and **aggregations**.
3. Provide reports and analysis similar to what HR departments use in real life.

**Project Tasks**

Below is a list of tasks that simulate real-world HR reporting:

**1. Employee and Salary Insights**

1. **Task 1.1**: Find all employees along with their current job titles and salaries.
   * **Hint**: Use employees, titles, and salaries tables.
2. **Task 1.2**: Identify the top 5 highest-paid employees in the company.
   * **Hint**: Use ORDER BY salary DESC and LIMIT.
3. **Task 1.3**: Find employees whose salaries are above the company-wide average salary.
   * **Hint**: Use a subquery to calculate the average salary.
4. **Task 1.4**: Retrieve the salary history for a specific employee (e.g., employee with emp\_no = 10001).
   * **Hint**: Use the salaries table and filter with WHERE.

**2. Department Analysis**

1. **Task 2.1**: List all departments and the number of employees in each department.
   * **Hint**: Use GROUP BY dept\_no with a COUNT () function.
2. **Task 2.2**: Retrieve the total salary paid in each department.
   * **Hint**: Use a JOIN between dept\_emp, departments, and salaries tables.
3. **Task 2.3**: Find the longest-serving employee in each department.
   * **Hint**: Use DATEDIFF() to calculate service time and GROUP BY for departments.
4. **Task 2.4**: Identify departments where the average salary is greater than 60,000.
   * **Hint**: Use GROUP BY with HAVING AVG(salary) > 60000.

**3. Management and Promotions**

1. **Task 3.1**: List all managers and the departments they manage, along with their hire dates.
   * **Hint**: Use the dept\_manager, employees, and departments tables.
2. **Task 3.2**: Identify employees who have held more than one title during their tenure.
   * **Hint**: Use COUNT(DISTINCT title) with a GROUP BY emp\_no and HAVING.
3. **Task 3.3**: Find employees who work in the same department as a specific employee (e.g., employee with emp\_no = 10001).
   * **Hint**: Use a subquery to find the dept\_no of the specific employee.

**4. Advanced Insights**

1. **Task 4.1**: Generate a report showing each employee's tenure (in years) in the company.
   * **Hint**: Use DATEDIFF() and divide by 365 to convert days into years.
2. **Task 4.2**: Find the percentage of employees in each department compared to the total number of employees.
   * **Hint**: Use COUNT() and calculate percentages.
3. **Task 4.3**: Retrieve employees who have salaries higher than the highest-paid manager.
   * **Hint**: Use a subquery to find the highest salary among managers.
4. **Task 4.4**: Identify the employee(s) with the longest overall tenure in the company.
   * **Hint**: Use DATEDIFF() between hire\_date and the current date.

**Deliverables**

For each task:

1. Write the **SQL query**.
2. Provide a short explanation of what the query does.
3. Optionally, describe what insight this query provides (e.g., how HR could use the result).

**Real-Life Use Case Example**

* **Top Earners Report**: Helps HR identify high-performing employees for bonuses or promotions.
* **Longest-Serving Employees**: Useful for recognizing loyal employees or identifying retirement candidates.
* **Department Insights**: Helps management decide on staffing and budgeting for each department.
* **Average Salary Analysis**: Ensures fair compensation and identifies departments where salaries might be too low or too high.

**Next Steps**

1. Start with **basic queries** (Tasks 1.1 to 1.4) to build confidence.
2. Move on to **grouping, aggregations, and joins** (Tasks 2.1 to 2.4).
3. Tackle **subqueries and advanced tasks** (Tasks 3.1 to 4.4).

If you need help solving or testing any query, just ask, and I’ll guide you step by step! 😊