# MySQL Employee Schema queries solution

28 May 2025 19:15

1)Employees Earning More Than Their Department's Average:

```
with CTE as(
                select
                        e.first_name as first_name,
                        e.salary as salary,
                        d.department_name as department_name,
                        avg(e.salary)over(partition by
                        d.department_name) avg_dept
                from employees e join departments d
                on e.department_id=d.department_id
      )
select
        first_name,
   salary,
   department_name,
   avg_dept
from CTE where salary>avg_dept;
```

2)Display each employee's first_name, last_name, and their manager's job_title. Include employees who do not have a manager.

```
select
        concat(e.first_name,' ', e.last_name),
   case
                when concat(m.first_name,' ', m.last_name) is null then 'No manager'
      else concat(m.first_name,' ', m.last_name) end ManagerFullName,
        case
                when m.job_title is null then 'No Manager'
      else m.job_title end as JobTitleOfManager
from employees e left join employees m
on e.manager_id=m.employee_id
```

3)Find the first_name, last_name, and job_title for employees whose job_title is unique within their specific department (i.e., no other employee in that department has the same job title).

```
WITH JobRank AS (
   SELECT
      first_name,
      last_name,
      job_title,
      department_id,
      ROW_NUMBER() OVER (PARTITION BY department_id, job_title ORDER BY employee_id) AS jo
b_rank,
      COUNT(*) OVER (PARTITION BY department_id, job_title) AS job_count
   FROM
      employees
)
SELECT
   first_name,
   last_name,
   job_title
FROM
```

```
    JobRank
WHERE
    job_count = 1;
```

4)For each employee, calculate their salary as a percentage of the cumulative salary within their department, ordered by hire_date.

- *Desired columns:* department_id, first_name, salary, cumulative_dept_salary, percent_of_cumulative_dept_salary

```
WITH DepartmentSalaries AS (
    SELECT
        department_id,
        first_name,
        salary,
        SUM(salary) OVER (PARTITION BY department_id) AS cumulative_dept_salary,
        hire_date
    FROM
        employees
)
SELECT
    department_id,
    first_name,
    salary,
    cumulative_dept_salary,
    (salary / cumulative_dept_salary) * 100 AS percent_of_cumulative_dept_salary
FROM
    DepartmentSalaries
ORDER BY
    hire_date;
```

5)**Rank of Manager by Number of Direct Reports:**

- Rank managers (employees who have direct reports) based on the number of employees they manage in descending order.
- *Desired columns:* manager_id, manager_first_name, manager_last_name, num_direct_reports, manager_rank

```
    select
            m.employee_id as manager_id,
            m.first_name as manager_first_name,
            count(e.employee_id) as num_direct_reports,
    dense_rank()over(order by count(e.employee_id) desc) manager_rank
      from
      employees e join employees m
      on e.manager_id=m.employee_id
      group by
            m.employee_id,
            m.first_name
```

6)**Difference in Salary from Previous Employee (Overall):**

- For each employee, calculate the difference between their current salary and the salary of the employee immediately preceding them (ordered by employee_id).

- *Desired columns:* employee_id, first_name, salary, previous_salary, salary_difference

```
select
        employee_id,
    first_name,
    salary,
    lag(salary)over() as previous_salary,
    ABS(salary-lag(salary)over()) as salary_difference
from
employees
```

7)**Employees Who Are the 2nd Highest Paid in Their Department:**

- Find the first_name, last_name, salary, and department_name of employees who are the second highest paid in their respective departments.

```
with secondHighSal as(select
    e.first_name,
    e.salary,
    d.department_name,
    dense_rank()over(partition by d.department_name order by e.salary desc) salRnk
from
employees e inner join departments d
on e.department_id=d.department_id)

select * from secondHighSal where salRnk = 2;
```

8)**Average Salary of Employees Hired Within a 90-Day Rolling Window (Ordered by Hire Date):**

- Calculate a 90-day rolling average of salaries for all employees, ordered by hire_date.
- *Desired columns:* employee_id, first_name, hire_date, salary, rolling_90_day_avg_salary

```
select
        employee_id,
    first_name,
    salary,
    hire_date,
    avg(salary)over(order by hire_date range between interval 89 day preceding and current
row) as rolling_90_day_avg_salary
from
        employees
order by hire_date;
```

9)**Count of Employees Hired in the Same Month and Year (for each employee):**

- For each employee, display their first_name, hire_date, and a count of how many employees were hired in the *same month and year* as them.

```
select
        first_name,
    hire_date,
    count(employee_id)over(partition by year(hire_date)) as hiredInYear,
    count(employee_id)over(partition by month(hire_date)) as hiredInMonth
from
```

```
        employees
order by hire_date;
```

10)**Salary Quartiles for All Employees:**

- Assign a quartile (1-4) to each employee based on their salary across all employees.
- *Desired columns:* employee_id, first_name, salary, salary_quartile

```
select
        employee_id,
        first_name,
   salary,
   ntile(4)over(order by salary) salary_quartile
from
        employees
order by salary;
```

11)**Difference in Hire Date from Department's First Hire:**

- For each employee, calculate the number of days between their hire_date and the hire_date of the first employee hired in their department.
- *Desired columns:* department_id, first_name, hire_date, days_since_dept_first_hire

```
select
        department_id,
        first_name,
   hire_date,
   datediff(hire_date,min(hire_date)over(partition by department_id order by hire_date))
days_since_dept_first_hire
from
        employees
```

12)**Average Salary of Manager's Direct Reports:**

- For each employee who is a manager, calculate the average salary of their *direct reports*.
- *Desired columns:* manager_id, manager_first_name, manager_last_name, avg_direct_report_salary

```
select
        m.manager_id,
        m.first_name,
   avg(e.salary)over(partition by m.manager_id) avg_direct_report_salary
from
        employees e join employees m
on e.manager_id=m.employee_id
```

13)**Percentage of Department's Total Salary for Each Employee:**

- Calculate each employee's salary as a percentage of the total salary of their department.
- *Desired columns:* department_id, first_name, salary, percent_of_dept_salary

```
select
        department_id,
   first_name,
   salary,
   cume_dist()over(partition by department_id order by salary)*100
```

```
cum_percent_of_dept_salary,
   percent_rank()over(partition by department_id order by salary)*100
percent_of_dept_salary
from
        employees
```

14)**Employees with the Same Salary as the Department's Lowest Earner:**

- Identify employees whose salary is equal to the minimum salary in their respective department.
- *Desired columns:* department_id, first_name, last_name, salary

```
with CTE as(select
        employee_id,
        department_id,
   first_name,
   salary,
   min(salary)over(partition by department_id) as minSal
from employees)
select
        *
from CTE where salary=minSal;
```

15)**Running Count of Employees Hired in Each Department (Ordered by Hire Date):**

- For each department, calculate a running count of employees hired, ordered by hire_date.
- *Desired columns:* department_id, first_name, hire_date, running_employee_count_in_dept

```
select
        employee_id,
        department_id,
   first_name,
   hire_date,
   count(hire_date)over(order by hire_date)as running_employee_count_in_dept
from employees
```

16)**Employees Who Are the Oldest Hired in Their Job Title (Across All Departments):**

- For each unique job_title, find the first_name, last_name, and hire_date of the employee who was hired earliest for that job_title.

```
with CTE as(select
   first_name,
   hire_date,
   job_title,
   row_number()over(partition by job_title order by hire_date)as old_emp_in_dept
from employees)
select
        first_name,
   hire_date,
   job_title
from CTE where old_emp_in_dept = 1;
```

17)**Average Salary of the Current and Next Two Employees (Ordered by Hire Date):**

- For each employee, calculate the average salary of themselves and the next two employees hired (overall, not per department).
- *Desired columns:* employee_id, first_name, hire_date, salary, avg_current_and_next_two

```
select
      employee_id,
  first_name,
  hire_date,
  salary,
  avg(salary)over(order by hire_date rows between current row and 2 following)
avg_current_and_next_two
from employees
```

18)**Employees Whose Salary is within 10% of the Department Average:**

- Find employees whose salary is within a 10% range (above or below) of the average salary for their department.
- *Desired columns:* department_id, first_name, last_name, salary, department_average_salary

```
WITH EmployeeWithAvgSalary AS (
  SELECT
    department_id,
    first_name,
    last_name,
    salary,
    AVG(salary) OVER (PARTITION BY department_id) AS department_average_salary
  FROM employees
)
SELECT
  department_id,
  first_name,
  last_name,
  salary,
  department_average_salary
FROM EmployeeWithAvgSalary
WHERE salary BETWEEN department_average_salary * 0.9 AND department_average_salary
 * 1.1;
```

19)**Count of Employees Whose Salary is Greater Than the Previous Employee's Salary (Ordered by Hire Date):**

- Count, for each employee, how many times their salary is strictly greater than the salary of the employee hired immediately before them (overall).
- *Desired columns:* employee_id, first_name, salary, is_salary_increasing_flag (1 if increasing, 0 otherwise), running_count_increasing_salary

```
WITH SalaryComparison AS (
  SELECT
    employee_id,
    first_name,
    hire_date,
    salary,
    LAG(salary) OVER (ORDER BY hire_date) AS previous_salary
  FROM employees
),
SalaryFlag AS (
  SELECT
    employee_id,
    first_name,
    salary,
    hire_date,
    CASE
      WHEN salary > previous_salary THEN 1
      ELSE 0
    END AS is_salary_increasing_flag
  FROM SalaryComparison
)
SELECT
  employee_id,
  first_name,
  salary,
  is_salary_increasing_flag,
  SUM(is_salary_increasing_flag) OVER (ORDER BY hire_date) AS running_count_increasing
_salary
FROM SalaryFlag;
```

20)**Highest Salary and Lowest Salary in a Rolling 3-Employee Window (Ordered by Salary):**

- For each employee, determine the maximum and minimum salary within a window of themselves and the two employees with the next highest salaries.
- *Desired columns:* employee_id, first_name, salary, rolling_max_salary, rolling_min_salary

```
    SELECT
            employee_id,
            first_name,
            salary,
            max(salary) OVER (ORDER BY salary rows between current row and 2 following) AS
            rolling_max_salary,
      min(salary) OVER (ORDER BY salary rows between current row and 2 following) AS
    rolling_max_salary
    FROM employees
```

21)**Gap in Hire Dates from Previous Employee (Overall):**

- Calculate the number of days between the current employee's hire_date and the hire_date of the previous employee (ordered by hire_date).
- *Desired columns:* employee_id, first_name, hire_date, days_since_previous_hire

```
    with CTE as(SELECT
            employee_id,
            first_name,
            hire_date,
            lag(hire_date) OVER (ORDER BY hire_date) AS PrevHire
    FROM employees)
    select
            employee_id,
            first_name,
            hire_date,
      hire_date-PrevHire as days_since_previous_hire
    from CTE;
```

22)**Department's Top Earner (Name and Salary) alongside Every Employee:**

- For every employee, display their first_name, last_name, salary, and also the first_name, last_name, and salary of the highest-paid employee in their department.

```
SELECT
      first_name,
  last_name,
      salary,
  department_id,
      first_value(salary)over(partition by department_id order by salary desc)
      dept_top_earner_salary,
  first_value(first_name)over(partition by department_id order by salary desc)
dept_top_earner_first_name,
      first_value(last_name)over(partition by department_id order by salary desc)
      dept_top_earner_last_name
FROM employees
#order by department_id, salary desc;
```