1. Employee Compensation Analysis:

- Write a SQL query to find the average salary of employees in each department. Include the department name and order the results by the average salary in descending order.

```
SELECT department.department_name,
    AVG(employee.salary) AS average_salary
FROM employee
JOIN department ON employee.department_id = department.department_id
GROUP BY department.department_name
ORDER BY average_salary DESC;
```

2. Employee Tenure and Compensation:

- Create a query to calculate the average salary of employees who have been with the company for more than 5 years. Group the results by gender and marital status.

```
SELECT gender, marital_status,
    AVG(salary) AS average_salary
FROM employee
WHERE hire_date <= CURDATE() - INTERVAL 5 YEAR
GROUP BY gender, marital_status;
```

3. Employee Shift and Pay Frequency:

- Write a SQL query to analyze the correlation between the shift employees work (morning, afternoon, or night) and their pay frequency (weekly, bi-weekly, or monthly). Show the count of employees for each shift and pay frequency combination.

```
SELECT shift, pay_frequency,
    COUNT(employee_id) AS employee_count
FROM employee
GROUP BY shift, pay_frequency;
```

4. Employee Performance Metrics:

- Using the employee data, write a SQL query to find employees who have not taken any sick leave for the past 12 months. Include the employee ID, name, and hire date.

```
SELECT employee_id, name, hire_date
FROM employee
WHERE employee_id NOT IN (
   SELECT employee_id
   FROM leave_records
   WHERE leave_type = 'Sick'
   AND leave_date >= CURDATE() - INTERVAL 12 MONTH
);
```

5. Department Performance Comparison:

- Write a query to compare the performance of different departments based on the total salary expense for the last quarter. Use the `employeepayhist` and `department` tables to extract the necessary information.

```
SELECT department.department_name,
     SUM(employeepayhist.salary) AS total_salary_expense
FROM employeepayhist
JOIN department ON employeepayhist.department_id = department.department_id
WHERE employeepayhist.payment_date BETWEEN CURDATE() - INTERVAL 3 MONTH AND
CURDATE()
GROUP BY department.department_name;
```

6. Employee Job Candidates and Performance:

- Write a SQL query to identify which job candidates (from the `jobcandidate` table) had the highest compensation offers based on their job interviews, and include the corresponding department name.

```
SELECT jobcandidate.candidate_id,
     jobcandidate.name,
     jobcandidate.salary_offer,
     department.department_name
FROM jobcandidate
JOIN department ON jobcandidate.department_id = department.department_id
ORDER BY jobcandidate.salary_offer DESC
LIMIT 1;
```

7. Employees With Bonus Eligibility:

- Create a query to list employees who are eligible for a bonus. Consider an employee eligible for a bonus if their sales quota (from the `salespersonquot` table) has been met or exceeded. Show employee ID, name, and bonus eligibility status.

```
SELECT employee.employee_id,
    employee.name,
    CASE
       WHEN salespersonquot.sales_achieved >= salespersonquot.sales_quota THEN 'Eligible'
       ELSE 'Not Eligible'
    END AS bonus_eligibility
FROM employee
JOIN salespersonquot ON employee.employee_id = salespersonquot.employee_id;
```

8. Performance Review Comparison:

- Write a SQL query to compare employee performance based on their performance review rating. Retrieve the employee ID, name, department, and performance rating from the `productreview` table and join it with the `employee` table.

```
SELECT employee.employee_id,
    employee.name,
    department.department_name,
    productreview.performance_rating
FROM employee
JOIN productreview ON employee.employee_id = productreview.employee_id
JOIN department ON employee.department_id = department.department_id;
```

9. Employee Compensation Based on Education:

- Write a SQL query to find the average compensation (salary) of employees based on their education level. Include the education level and the corresponding average salary in the results.

```
SELECT education_level,
    AVG(salary) AS average_salary
FROM employee
GROUP BY education_level;
```

10. Employee Leave and Compensation:

- Write a query to identify the total compensation for employees who have used more than 10 days of sick leave in the past year. The output should include employee ID, name, total compensation, and sick leave used.

```
SELECT employee.employee_id,
    employee.name,
    SUM(employeepayhist.salary) AS total_compensation,
    COUNT(leave_records.leave_id) AS sick_leave_used
FROM employee
JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id
JOIN leave_records ON employee.employee_id = leave_records.employee_id
WHERE leave_records.leave_type = 'Sick'
AND leave_records.leave_date >= CURDATE() - INTERVAL 1 YEAR
GROUP BY employee.employee_id
HAVING sick_leave_used > 10;
```

11. Total Compensation (Salary + Bonus) for Each Employee:

- How would you calculate the total compensation (Salary + Bonus) for each employee in the employee table, and display the results along with the employee's ID and name?
  Required Tables: employee, salesperson

```
SELECT employee.employee_id,

    employee.name,

    (employee.salary + IFNULL(salesperson.bonus, 0)) AS total_compensation

FROM employee

LEFT JOIN salesperson ON employee.employee_id = salesperson.employee_id;
```

12. Total Hours Worked by Each Employee in a Given Month:

- How would you write a query to find the total hours worked by each employee in a given month?
  Required Tables: employeedepart, shift, employee

```
SELECT employee.employee_id,

    employee.name,
```

```
    SUM(shift.hours) AS total_hours_worked
FROM employeedepart
JOIN shift ON employeedepart.shift_id = shift.shift_id
JOIN employee ON employeedepart.employee_id = employee.employee_id
WHERE MONTH(shift.shift_date) = 5 -- Example: May
GROUP BY employee.employee_id;
```

13. Top 5 Highest-Paid Employees:

- Write a query to find the top 5 highest-paid employees based on their salary from the `employee` table.
  Required Tables: employee

```
SELECT employee_id, name, salary
FROM employee
ORDER BY salary DESC
LIMIT 5;
```

14. Average Salary per Department and Comparison with Budget:

- How would you calculate the average salary for each department and compare it with the department's overall compensation budget (if available in the `department` table)?
  Required Tables: employee, employeedepart, department

```
SELECT department.department_name,
    AVG(employee.salary) AS average_salary,
    department.budget AS department_budget
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
JOIN department ON employeedepart.department_id = department.department_id
```

```
GROUP BY department.department_name;
```

15. Employees Assigned to Multiple Departments in the Same Year:

- What SQL query would you use to identify employees who have been assigned to multiple departments in the same year (use the `employeedepart` table)?
  Required Tables: employee, employeedepart

```
SELECT employee.employee_id,

    employee.name

FROM employee

JOIN employeedepart ON employee.employee_id = employeedepart.employee_id

WHERE YEAR(employeedepart.assignment_date) = YEAR(CURDATE())

GROUP BY employee.employee_id

HAVING COUNT(DISTINCT employeedepart.department_id) > 1;
```

16. Total Number of Employees Who Took Sick Leave in the Last Quarter:

- Write a query to find the total number of employees who have taken sick leave in the last quarter, along with the total hours taken.
  Required Tables: employee

```
SELECT COUNT(DISTINCT leave_records.employee_id) AS total_employees,

    SUM(leave_records.hours) AS total_sick_leave_hours

FROM leave_records

JOIN employee ON leave_records.employee_id = employee.employee_id

WHERE leave_records.leave_type = 'Sick'

AND leave_records.leave_date BETWEEN CURDATE() - INTERVAL 3 MONTH AND
CURDATE();
```

17. Average Compensation (Salary, Bonus) per Gender:

- How would you create a report that shows the average compensation (Salary, Bonus) per gender from the `employee` and `salesperson` tables?
  Required Tables: employee, salesperson

```
SELECT employee.gender,

    AVG(employee.salary + IFNULL(salesperson.bonus, 0)) AS average_compensation

FROM employee

LEFT JOIN salesperson ON employee.employee_id = salesperson.employee_id

GROUP BY employee.gender;
```

18. Employees with More Than 5 Years with the Company and Their Compensation:

- Write a query to list the employees who have been with the company for more than 5 years, along with their compensation details.
  Required Tables: employee, employeepayhist

```
SELECT employee.employee_id,

    employee.name,

    employee.salary,

    employeepayhist.bonus

FROM employee

JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id

WHERE employee.hire_date <= CURDATE() - INTERVAL 5 YEAR;
```

19. Percentage Increase in Employee Compensation (Salary, Bonus):

- How can you find the percentage increase in an employee's compensation (salary, bonus) compared to the previous year using the `employeepayhist` table?
  Required Tables: employeepayhist, employee

```
SELECT employee.employee_id,

    employee.name,

    ((SUM(employeepayhist.salary) - LAG(SUM(employeepayhist.salary)) OVER (PARTITION
BY employee.employee_id ORDER BY employeepayhist.pay_date))

    / LAG(SUM(employeepayhist.salary)) OVER (PARTITION BY employee.employee_id
ORDER BY employeepayhist.pay_date)) * 100 AS salary_increase_percentage,

    ((SUM(employeepayhist.bonus) - LAG(SUM(employeepayhist.bonus)) OVER (PARTITION
BY employee.employee_id ORDER BY employeepayhist.pay_date))

    / LAG(SUM(employeepayhist.bonus)) OVER (PARTITION BY employee.employee_id
ORDER BY employeepayhist.pay_date)) * 100 AS bonus_increase_percentage

FROM employee

JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id

GROUP BY employee.employee_id;
```

20. Employees Who Received a Pay Raise in the Last 6 Months:

- Using the `employee` and `employeepayhist` tables, write a query to list the employees who received a pay raise in the last 6 months and the corresponding date of the pay raise.
  Required Tables: employee, employeepayhist

```
SELECT employee.employee_id,

    employee.name,

    employeepayhist.pay_date,

    employeepayhist.salary

FROM employee

JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id

WHERE employeepayhist.pay_date >= CURDATE() - INTERVAL 6 MONTH

AND employeepayhist.salary >
```

```
(SELECT MAX(salary)
 FROM employeepayhist AS subquery
 WHERE subquery.employee_id = employee.employee_id
 AND subquery.pay_date < employeepayhist.pay_date);
```

21. Employee Salary Distribution:

- Write a query to calculate the average, minimum, and maximum salary for each department from the employee and employee department tables. Include only current employees. *Required Tables: employee, employeedepart*

```
SELECT employeedepart.department_id,
    AVG(employee.salary) AS avg_salary,
    MIN(employee.salary) AS min_salary,
    MAX(employee.salary) AS max_salary
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
WHERE employee.current_flag = 'Y' -- Only current employees
GROUP BY employeedepart.department_id;
```

22. Performance Review by Gender:

- Write a query to calculate the average performance rating (if available) of employees grouped by gender. Consider only employees who have a performance rating recorded. *Required Tables: employee*

```
SELECT employee.gender,
    AVG(employee.performance_rating) AS avg_performance_rating
FROM employee
```

```
WHERE employee.performance_rating IS NOT NULL

GROUP BY employee.gender;
```

23. Employee Compensation Trend:

- Write a query to analyze the average salary of employees by their hiring year. Provide the trend of average salaries for employees hired in each year. *Required Tables: employee*

```
SELECT YEAR(employee.hire_date) AS hire_year,

    AVG(employee.salary) AS avg_salary

FROM employee

GROUP BY YEAR(employee.hire_date)

ORDER BY hire_year;
```

24. Top Performers in Sales:

- Using the salesperson and salesorderheader tables, write a query to find the top 5 salespersons based on their total sales (SalesYTD) for the current year. *Required Tables: salesperson, salesorderheader*

```
SELECT salesperson.salesperson_id,

    salesperson.name,

    SUM(salesorderheader.salesytd) AS total_sales

FROM salesperson

JOIN salesorderheader ON salesperson.salesperson_id = salesorderheader.salesperson_id

WHERE YEAR(salesorderheader.sales_order_date) = YEAR(CURDATE())

GROUP BY salesperson.salesperson_id

ORDER BY total_sales DESC

LIMIT 5;
```

25. Employee Job Change Analysis:

- Write a query to find all employees who have changed departments in the last two years. Use the employeedepart table to identify changes in department assignments. *Required Tables: employeedepart*

```
SELECT DISTINCT employee_id,

    employee.name

FROM employeedepart

JOIN employee ON employeedepart.employee_id = employee.employee_id

WHERE employeedepart.assignment_date BETWEEN CURDATE() - INTERVAL 2 YEAR AND
CURDATE();
```

26. Average Vacation Hours by Department:

- Write a query to calculate the average vacation hours used by employees in each department. Consider only employees who are currently active (use the employee table's CurrentFlag). *Required Tables: employee, employeedepart, leave_records*

```
SELECT employeedepart.department_id,

    AVG(leave_records.vacation_hours) AS avg_vacation_hours

FROM leave_records

JOIN employee ON leave_records.employee_id = employee.employee_id

JOIN employeedepart ON employee.employee_id = employeedepart.employee_id

WHERE employee.current_flag = 'Y'

GROUP BY employeedepart.department_id;
```

27. Employee Compensation vs. Experience:

- Write a query to calculate the average salary (from the employeepayhist table) of employees based on their years of experience. Assume that years of experience can be

calculated as the difference between the employee's hire date and the current date. *Required Tables: employeepayhist, employee*

```
SELECT employee.employee_id,
    employee.name,
    TIMESTAMPDIFF(YEAR, employee.hire_date, CURDATE()) AS years_of_experience,
    AVG(employeepayhist.salary) AS avg_salary
FROM employee
JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id
GROUP BY employee.employee_id;
```

28. Employees with Pending Compensation Updates:

- Write a query to find employees whose compensation details have not been updated in the last 6 months. Use the employeepayhist table to check the modification dates. *Required Tables: employeepayhist, employee*

```
SELECT employee.employee_id,
    employee.name
FROM employee
LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id
WHERE employeepayhist.last_update_date < CURDATE() - INTERVAL 6 MONTH
OR employeepayhist.last_update_date IS NULL;
```

29. Salary Comparison by Marital Status:

- Write a query to compare the average salary of employees with different marital statuses. Group the results by marital status. *Required Tables: employee*

```
SELECT employee.marital_status,
    AVG(employee.salary) AS avg_salary
```

```
FROM employee

GROUP BY employee.marital_status;
```

30. Department Performance Analysis:

- Write a query to calculate the total number of employees and the average salary for each department. Also, find the department with the highest average salary. *Required Tables: employee, employeedepart*

```
SELECT employeedepart.department_id,

    COUNT(employee.employee_id) AS total_employees,

    AVG(employee.salary) AS avg_salary

FROM employee

JOIN employeedepart ON employee.employee_id = employeedepart.employee_id

GROUP BY employeedepart.department_id;


-- Find the department with the highest average salary

SELECT department_id

FROM (

  SELECT employeedepart.department_id,

      AVG(employee.salary) AS avg_salary

  FROM employee

  JOIN employeedepart ON employee.employee_id = employeedepart.employee_id

  GROUP BY employeedepart.department_id

) AS dept_avg_salaries

ORDER BY avg_salary DESC

LIMIT 1;
```

31. Find the average salary for employees in each department.

Required Tables: employee, employeedepart

```
SELECT employeedepart.department_id,
    AVG(employee.salary) AS avg_salary
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
GROUP BY employeedepart.department_id;
```

32. Retrieve the top 5 highest-paid employees along with their department, and sort them by salary.

Required Tables: employee, employeedepart

```
SELECT employee.employee_id,
    employee.name,
    employee.salary,
    employeedepart.department_id
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
ORDER BY employee.salary DESC
LIMIT 5;
```

33. Find employees who have taken more than 5 sick leave hours in the past month.

Required Tables: leave_records, employee

```
SELECT employee.employee_id,
```

```
    employee.name,
    SUM(leave_records.sick_leave_hours) AS total_sick_leave
FROM leave_records
JOIN employee ON leave_records.employee_id = employee.employee_id
WHERE leave_records.leave_type = 'Sick'
  AND leave_records.leave_date BETWEEN CURDATE() - INTERVAL 1 MONTH AND
CURDATE()
GROUP BY employee.employee_id
HAVING total_sick_leave > 5;
```

34. List all employees whose salary was last updated within the last 6 months, along with their department name.

Required Tables: employeepayhist, employee, employeedepart

```
SELECT employee.employee_id,
    employee.name,
    employeedepart.department_id,
    employeepayhist.last_update_date
FROM employeepayhist
JOIN employee ON employeepayhist.employee_id = employee.employee_id
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
WHERE employeepayhist.last_update_date > CURDATE() - INTERVAL 6 MONTH;
```

35. Calculate the total bonus payout for each employee, including the employee's name and bonus percentage.

Required Tables: employeepayhist, employee

```
SELECT employee.employee_id,
    employee.name,
    employeepayhist.bonus_percentage,
    (employee.salary * employeepayhist.bonus_percentage / 100) AS total_bonus
FROM employee
JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id;
```

36. Identify the employee with the highest number of vacation hours available and display their department and title.

Required Tables: leave_records, employee, employeedepart

```
SELECT employee.employee_id,
    employee.name,
    employeedepart.department_id,
    employee.job_title,
    MAX(leave_records.vacation_hours) AS max_vacation_hours
FROM leave_records
JOIN employee ON leave_records.employee_id = employee.employee_id
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
GROUP BY employee.employee_id
ORDER BY max_vacation_hours DESC
LIMIT 1;
```

37. Find the employees who have worked the longest (measured by the number of years since their hire date) and their corresponding salary.

Required Tables: employee

```
SELECT employee.employee_id,
    employee.name,
    TIMESTAMPDIFF(YEAR, employee.hire_date, CURDATE()) AS years_of_experience,
    employee.salary
FROM employee
ORDER BY years_of_experience DESC
LIMIT 1;
```

38. Calculate the average pay for employees based on their shift and display the results along with the shift name.

Required Tables: employee, employeedepart

```
SELECT employeedepart.shift,
    AVG(employee.salary) AS avg_salary
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
GROUP BY employeedepart.shift;
```

39. Find employees who have been promoted (by comparing `StartDate` and `EndDate` in `employeedepart`) and their salary change.

Required Tables: employeedepart, employee

```
SELECT employee.employee_id,
    employee.name,
    employeedepart.department_id,
    employeedepart.start_date,
    employeedepart.end_date,
```

```
    (employee.salary - employeedepart.previous_salary) AS salary_change
FROM employeedepart
JOIN employee ON employeedepart.employee_id = employee.employee_id
WHERE employeedepart.end_date IS NOT NULL
  AND employeedepart.start_date < employeedepart.end_date;
```

40. Retrieve all employees whose compensation rate has changed within the past year, along with the new rate and change date.

Required Tables: employeepayhist, employee

```
SELECT employee.employee_id,
    employee.name,
    employeepayhist.new_salary_rate,
    employeepayhist.change_date
FROM employeepayhist
JOIN employee ON employeepayhist.employee_id = employee.employee_id
WHERE employeepayhist.change_date > CURDATE() - INTERVAL 1 YEAR;
```

41. Calculate the total compensation of each employee using their salary and any bonus data.

Required Tables: employee, employeepayhist

```
SELECT employee.employee_id,
    employee.name,
    employee.salary + IFNULL(employeepayhist.bonus_amount, 0) AS total_compensation
```

```
FROM employee
LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id;
```

42. Find the average tenure of employees within each department.

Required Tables: employee, employeedepart

```
SELECT employeedepart.department_id,
    AVG(TIMESTAMPDIFF(YEAR, employee.hire_date, CURDATE())) AS avg_tenure
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
GROUP BY employeedepart.department_id;
```

43. Retrieve the employees who have received the highest bonuses within a specific department.

Required Tables: employee, employeepayhist, employeedepart

```
SELECT employee.employee_id,
    employee.name,
    employeepayhist.bonus_amount,
    employeedepart.department_id
FROM employeepayhist
JOIN employee ON employeepayhist.employee_id = employee.employee_id
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
WHERE employeepayhist.bonus_amount = (
  SELECT MAX(bonus_amount)
  FROM employeepayhist
```

```
    JOIN employee ON employeepayhist.employee_id = employee.employee_id
    WHERE employeedepart.department_id = employeedepart.department_id
);
```

44. Find employees eligible for promotion based on their performance metrics (e.g., SalesYTD, PerformanceRating).

Required Tables: employee, salesdata, employeepayhist

```
SELECT employee.employee_id,
       employee.name,
       salesdata.SalesYTD,
       employee.performance_rating
FROM employee
JOIN salesdata ON employee.employee_id = salesdata.employee_id
WHERE salesdata.SalesYTD > 500000 -- example threshold for sales
  AND employee.performance_rating >= 4; -- assuming 4 is considered high performance
```

45. Calculate the average sick leave taken by employees over the past year.

Required Tables: leave_records, employee

```
SELECT employee.employee_id,
       employee.name,
       AVG(leave_records.sick_leave_hours) AS avg_sick_leave
FROM leave_records
JOIN employee ON leave_records.employee_id = employee.employee_id
WHERE leave_records.leave_type = 'Sick'
```

```
  AND leave_records.leave_date > CURDATE() - INTERVAL 1 YEAR
GROUP BY employee.employee_id;
```

46. Find the top 5 employees who have the highest sales for the current fiscal year.

Required Tables: employee, salesdata

```
SELECT employee.employee_id,
    employee.name,
    SUM(salesdata.SalesYTD) AS total_sales
FROM employee
JOIN salesdata ON employee.employee_id = salesdata.employee_id
WHERE salesdata.year = YEAR(CURDATE()) -- assuming fiscal year
GROUP BY employee.employee_id
ORDER BY total_sales DESC
LIMIT 5;
```

47. Retrieve the total number of employees who have been with the company for less than 5 years and their average performance rating.

Required Tables: employee

```
SELECT COUNT(employee.employee_id) AS num_employees,
    AVG(employee.performance_rating) AS avg_performance_rating
FROM employee
WHERE TIMESTAMPDIFF(YEAR, employee.hire_date, CURDATE()) < 5;
```

48. Identify employees with the highest and lowest salaries in each department, and what SQL joins would you use?

Required Tables: employee, employeedepart

```
SELECT employeedepart.department_id,
    MAX(employee.salary) AS highest_salary,
    MIN(employee.salary) AS lowest_salary
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
GROUP BY employeedepart.department_id;
```

49. Calculate the average salary for each job title in the company.

Required Tables: employee

```
SELECT employee.job_title,
    AVG(employee.salary) AS avg_salary
FROM employee
GROUP BY employee.job_title;
```

50. Join employee compensation data with the department data to analyze how compensation correlates with department size.

Required Tables: employee, employeedepart, department

```
SELECT employeedepart.department_id,
    department.department_name,
    AVG(employee.salary) AS avg_salary,
    COUNT(employee.employee_id) AS department_size
FROM employee
```

```
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
JOIN department ON employeedepart.department_id = department.department_id
GROUP BY employeedepart.department_id;
```

51. Query the average salary by department, using the `employee` and `employeedepart` tables, considering only employees who have been employed for over 2 years.

```
SELECT employeedepart.department_id,
    AVG(employee.salary) AS avg_salary
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
WHERE TIMESTAMPDIFF(YEAR, employee.hire_date, CURDATE()) > 2
GROUP BY employeedepart.department_id;
```

52. Find the employees who have received the highest bonus percentage in each department. Use the `employee`, `employeeaddres`, and `salesperson` tables.

```
SELECT employee.employee_id,
    employee.name,
    employeeaddres.department_id,
    MAX(salesperson.bonus_percentage) AS highest_bonus_percentage
FROM employee
JOIN employeeaddres ON employee.employee_id = employeeaddres.employee_id
```

```
JOIN salesperson ON employee.employee_id = salesperson.employee_id

GROUP BY employeeaddres.department_id;
```

53. Calculate the total compensation (salary + bonuses) for employees over the past year, by joining the `employee` and `employeepayhist` tables.

```
SELECT employee.employee_id,

    employee.name,

    SUM(employee.salary + IFNULL(employeepayhist.bonus_amount, 0)) AS
total_compensation

FROM employee

JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id

WHERE employeepayhist.payment_date > CURDATE() - INTERVAL 1 YEAR

GROUP BY employee.employee_id;
```

54. Identify employees who have not used their sick leave in the past year. Use the `employee` table and a `SickLeaveHours` condition.

```
SELECT employee.employee_id,

    employee.name

FROM employee

LEFT JOIN leave_records ON employee.employee_id = leave_records.employee_id

                AND leave_records.leave_type = 'Sick'

WHERE leave_records.sick_leave_hours IS NULL

  OR leave_records.leave_date < CURDATE() - INTERVAL 1 YEAR;
```

55. Retrieve the list of departments with the highest turnover rates in the last quarter. Use the `employee`, `employeedepart`, and `department` tables.

```
SELECT employeedepart.department_id,
    department.department_name,
    COUNT(employee.employee_id) AS department_turnover
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
JOIN department ON employeedepart.department_id = department.department_id
WHERE employee.hire_date < CURDATE() - INTERVAL 3 MONTH
GROUP BY employeedepart.department_id
ORDER BY department_turnover DESC;
```

56. Determine the number of employees who have reached their target sales quota, joining the `salesperson`, `salespersonquot`, and `salesorderheade` tables.

```
SELECT salesperson.employee_id,
    COUNT(salesorderheade.order_id) AS sales_orders
FROM salesperson
JOIN salespersonquot ON salesperson.employee_id = salespersonquot.employee_id
JOIN salesorderheade ON salesperson.employee_id = salesorderheade.salesperson_id
WHERE salesorderheade.total_amount >= salespersonquot.sales_quota
GROUP BY salesperson.employee_id;
```

57. Retrieve the total number of sick leave hours taken by employees in a specific department in the past year. Join the `employee`, `employeedepart`, and `employeeaddres` tables.

```
SELECT employeedepart.department_id,
    SUM(leave_records.sick_leave_hours) AS total_sick_leave_hours
FROM leave_records
JOIN employee ON leave_records.employee_id = employee.employee_id
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
JOIN employeeaddres ON employee.employee_id = employeeaddres.employee_id
WHERE leave_records.leave_type = 'Sick'
  AND leave_records.leave_date > CURDATE() - INTERVAL 1 YEAR
GROUP BY employeedepart.department_id;
```

58. Calculate the average salary increase for employees who changed their pay rate in the last 6 months, using the `employee` and `employeepayhist` tables.

```
SELECT employee.employee_id,
    AVG(employeepayhist.new_salary - employeepayhist.old_salary) AS avg_salary_increase
FROM employee
JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id
WHERE employeepayhist.change_date > CURDATE() - INTERVAL 6 MONTH
GROUP BY employee.employee_id;
```

59. Find the employees who have the highest number of customer interactions (customer service calls), joining the `employee` and `customer` tables.

```
SELECT employee.employee_id,
    employee.name,
    COUNT(customer.interaction_id) AS customer_interactions
FROM employee
JOIN customer ON employee.employee_id = customer.employee_id
GROUP BY employee.employee_id
ORDER BY customer_interactions DESC;
```

60. Calculate the total annual compensation for each employee, considering base salary and bonuses. Join the `employee`, `employeepayhist`, and `salesperson` tables.

```
SELECT employee.employee_id,
    employee.name,
    employee.salary + IFNULL(SUM(employeepayhist.bonus_amount), 0) AS
total_annual_compensation
FROM employee
LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id
LEFT JOIN salesperson ON employee.employee_id = salesperson.employee_id
GROUP BY employee.employee_id;
```

61. Find the average salary of employees in each department.

```
SELECT employeedepart.department_id,
    AVG(employee.salary) AS avg_salary
FROM employee
```

```
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
GROUP BY employeedepart.department_id;
```

62. Identify the top 5 highest-paid employees along with their department and position.

```
SELECT employee.employee_id,
    employee.name,
    employee.salary,
    employeedepart.department_id,
    department.position
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
JOIN department ON employeedepart.department_id = department.department_id
ORDER BY employee.salary DESC
LIMIT 5;
```

63. Find the total vacation hours taken by employees in the last year.

```
SELECT SUM(vacation_hours) AS total_vacation_hours
FROM leave_records
WHERE leave_records.leave_type = 'Vacation'
  AND leave_records.leave_date > CURDATE() - INTERVAL 1 YEAR;
```

64. Calculate the employee turnover rate for each department over the last 6 months.

```
SELECT employeedepart.department_id,
```

```
    COUNT(employee.employee_id) AS department_turnover
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
WHERE employee.hire_date < CURDATE() - INTERVAL 6 MONTH
GROUP BY employeedepart.department_id;
```

65. Display the total number of employees and the average salary by gender.

```
SELECT employee.gender,
    COUNT(employee.employee_id) AS total_employees,
    AVG(employee.salary) AS avg_salary
FROM employee
GROUP BY employee.gender;
```

66. List employees who have been with the company for more than 5 years and have not taken any sick leave.

```
SELECT employee.employee_id,
    employee.name
FROM employee
LEFT JOIN leave_records ON employee.employee_id = leave_records.employee_id
                AND leave_records.leave_type = 'Sick'
WHERE TIMESTAMPDIFF(YEAR, employee.hire_date, CURDATE()) > 5
  AND leave_records.leave_id IS NULL;
```

67. Find the department with the highest average compensation, including base salary and bonuses.

```
SELECT employeedepart.department_id,

    AVG(employee.salary + IFNULL(employeepayhist.bonus_amount, 0)) AS
avg_compensation

FROM employee

JOIN employeedepart ON employee.employee_id = employeedepart.employee_id

LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id

GROUP BY employeedepart.department_id

ORDER BY avg_compensation DESC

LIMIT 1;
```

68. Calculate the salary growth rate of each employee based on their salary history.

```
SELECT employee.employee_id,

    ((MAX(employeepayhist.salary) - MIN(employeepayhist.salary)) /
MIN(employeepayhist.salary)) * 100 AS salary_growth_rate

FROM employeepayhist

JOIN employee ON employeepayhist.employee_id = employee.employee_id

GROUP BY employee.employee_id;
```

69. Query the employees who have worked in multiple departments during their tenure.

```
SELECT employee.employee_id,

    employee.name

FROM employeedepart
```

```
JOIN employee ON employeedepart.employee_id = employee.employee_id

GROUP BY employee.employee_id

HAVING COUNT(DISTINCT employeedepart.department_id) > 1;
```

70. Show the average employee performance score by department and compare it to their compensation (salary + bonuses).

```
SELECT employeedepart.department_id,

    AVG(employee.performance_score) AS avg_performance_score,

    AVG(employee.salary + IFNULL(employeepayhist.bonus_amount, 0)) AS
avg_compensation

FROM employee

JOIN employeedepart ON employee.employee_id = employeedepart.employee_id

LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id

GROUP BY employeedepart.department_id;
```

71. Calculate the average salary for employees in each department using SQL.

```
SELECT employeedepart.department_id,

    AVG(employee.salary) AS avg_salary

FROM employee

JOIN employeedepart ON employee.employee_id = employeedepart.employee_id

GROUP BY employeedepart.department_id;
```

72. Find the total vacation hours used by each employee for the last 6 months.

```
SELECT employee.employee_id,
    employee.name,
    SUM(vacation_hours) AS total_vacation_hours
FROM leave_records
JOIN employee ON leave_records.employee_id = employee.employee_id
WHERE leave_records.leave_type = 'Vacation'
  AND leave_records.leave_date > CURDATE() - INTERVAL 6 MONTH
GROUP BY employee.employee_id;
```

73. Find the top 5 employees who received the highest bonuses last year.

```
SELECT employee.employee_id,
    employee.name,
    SUM(employeepayhist.bonus_amount) AS total_bonus
FROM employeepayhist
JOIN employee ON employeepayhist.employee_id = employee.employee_id
WHERE employeepayhist.pay_date > CURDATE() - INTERVAL 1 YEAR
GROUP BY employee.employee_id
ORDER BY total_bonus DESC
LIMIT 5;
```

74. Calculate the average performance rating for employees who have been with the company for more than 5 years.

```
SELECT AVG(employee.performance_rating) AS avg_performance_rating
FROM employee
WHERE TIMESTAMPDIFF(YEAR, employee.hire_date, CURDATE()) > 5;
```

75. List all employees who have not received a salary increase in the last 12 months.

```
SELECT employee.employee_id,
    employee.name,
    MAX(employeepayhist.pay_date) AS last_salary_change_date
FROM employeepayhist
JOIN employee ON employeepayhist.employee_id = employee.employee_id
WHERE employeepayhist.pay_date < CURDATE() - INTERVAL 12 MONTH
GROUP BY employee.employee_id
HAVING MAX(employeepayhist.pay_date) < CURDATE() - INTERVAL 12 MONTH;
```

76. Find the number of employees in each shift and their corresponding average pay rate.

```
SELECT employee.shift,
    COUNT(employee.employee_id) AS num_employees,
    AVG(employee.salary) AS avg_salary
FROM employee
GROUP BY employee.shift;
```

77. Identify employees whose compensation is higher than the average compensation for their department.

```
SELECT employee.employee_id,
       employee.name,
       employee.salary + IFNULL(employeepayhist.bonus_amount, 0) AS total_compensation
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id
WHERE (employee.salary + IFNULL(employeepayhist.bonus_amount, 0)) >
      (SELECT AVG(employee.salary + IFNULL(employeepayhist.bonus_amount, 0))
       FROM employee
       LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id
       WHERE employeedepart.department_id = employeedepart.department_id);
```

78. Calculate the total number of sick leave hours taken by employees in the past quarter.

```
SELECT SUM(sick_leave_hours) AS total_sick_leave_hours
FROM leave_records
WHERE leave_records.leave_type = 'Sick'
  AND leave_records.leave_date > CURDATE() - INTERVAL 3 MONTH;
```

79. Find employees who have been assigned to multiple departments and list the start and end dates for each assignment.

```
SELECT employee.employee_id,
       employee.name,
       employeedepart.department_id,
       employeedepart.start_date,
```

```
    employeedepart.end_date
FROM employeedepart
JOIN employee ON employeedepart.employee_id = employee.employee_id
GROUP BY employee.employee_id, employeedepart.department_id
HAVING COUNT(DISTINCT employeedepart.department_id) > 1;
```

80. Get the number of employees who received training last year and their average compensation compared to the rest of the workforce.

```
SELECT
  COUNT(DISTINCT employee.employee_id) AS num_employees_trained,
  AVG(employee.salary + IFNULL(employeepayhist.bonus_amount, 0)) AS
avg_compensation_trained,
  (SELECT AVG(employee.salary + IFNULL(employeepayhist.bonus_amount, 0))
   FROM employee
   LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id)
AS avg_compensation_all
FROM employee
JOIN training_records ON employee.employee_id = training_records.employee_id
LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id
WHERE training_records.training_date > CURDATE() - INTERVAL 1 YEAR;
```

81. Calculate the total salary expense for each department, including only employees who are currently employed (`CurrentFlag = 'Y'`).

```
SELECT employeedepart.department_id,
    SUM(employee.salary) AS total_salary_expense
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
WHERE employee.CurrentFlag = 'Y'
GROUP BY employeedepart.department_id;
```

82. Find the average salary of employees who are on leave (`SickLeaveHours > 0`) within each department.

```
SELECT employeedepart.department_id,
    AVG(employee.salary) AS avg_salary_on_leave
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
JOIN leave_records ON employee.employee_id = leave_records.employee_id
WHERE leave_records.SickLeaveHours > 0
GROUP BY employeedepart.department_id;
```

83. Retrieve the list of employees whose salaries have increased since their last pay history update.

```
SELECT employee.employee_id,
    employee.name,
    employee.salary AS current_salary,
```

```
    MAX(employeepayhist.pay_date) AS last_salary_update
FROM employee
JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id
GROUP BY employee.employee_id
HAVING employee.salary > MAX(employeepayhist.salary);
```

84. Calculate the total compensation (Salary + Bonus) for each employee who has completed more than 5 years of service (`HireDate`).

```
SELECT employee.employee_id,
    employee.name,
    employee.salary + IFNULL(employeepayhist.bonus_amount, 0) AS total_compensation
FROM employee
LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id
WHERE TIMESTAMPDIFF(YEAR, employee.hire_date, CURDATE()) > 5;
```

85. Determine the percentage of employees in each department who are earning above the average salary for that department.

```
SELECT employeedepart.department_id,
    COUNT(CASE WHEN employee.salary > dept_avg.avg_salary THEN 1 END) * 100 /
COUNT(employee.employee_id) AS percentage_above_avg
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
JOIN (SELECT department_id, AVG(salary) AS avg_salary
    FROM employee
    GROUP BY department_id) dept_avg
```

```
ON employeedepart.department_id = dept_avg.department_id

GROUP BY employeedepart.department_id;
```

86. Calculate the total vacation hours used by each employee in the last year, grouping the results by department.

```
SELECT employeedepart.department_id,

    employee.employee_id,

    SUM(leave_records.vacation_hours) AS total_vacation_hours

FROM employee

JOIN employeedepart ON employee.employee_id = employeedepart.employee_id

JOIN leave_records ON employee.employee_id = leave_records.employee_id

WHERE leave_records.leave_type = 'Vacation'

  AND leave_records.leave_date > CURDATE() - INTERVAL 1 YEAR

GROUP BY employeedepart.department_id, employee.employee_id;
```

87. Retrieve a list of employees who have not received any bonuses in the last year (using the employee compensation and performance tables).

```
SELECT employee.employee_id,

    employee.name

FROM employee

LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id

WHERE employeepayhist.bonus_amount IS NULL

  OR employeepayhist.pay_date < CURDATE() - INTERVAL 1 YEAR;
```

88. Calculate the average number of vacation hours used per department, only for employees with over 3 years of service.

```
SELECT employeedepart.department_id,
    AVG(leave_records.vacation_hours) AS avg_vacation_hours
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
JOIN leave_records ON employee.employee_id = leave_records.employee_id
WHERE TIMESTAMPDIFF(YEAR, employee.hire_date, CURDATE()) > 3
GROUP BY employeedepart.department_id;
```

89. Identify the department with the highest average compensation (including salary and bonuses) for the last quarter.

```
SELECT employeedepart.department_id,
    AVG(employee.salary + IFNULL(employeepayhist.bonus_amount, 0)) AS
avg_compensation
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id
WHERE employeepayhist.pay_date > CURDATE() - INTERVAL 3 MONTH
GROUP BY employeedepart.department_id
ORDER BY avg_compensation DESC
LIMIT 1;
```

90. Find the employees who have been promoted (based on job title changes) within the last 6 months, and calculate their average salary before and after the promotion.

```
SELECT employee.employee_id,
    employee.name,
    MIN(employeepayhist.salary) AS avg_salary_before_promotion,
    MAX(employeepayhist.salary) AS avg_salary_after_promotion
FROM employee
JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id
WHERE employeepayhist.pay_date > CURDATE() - INTERVAL 6 MONTH
GROUP BY employee.employee_id
HAVING COUNT(DISTINCT employeepayhist.job_title) > 1;
```

91. Employee Compensation Overview

Find the average compensation of employees in each department, including their bonus and commission.

```
SELECT employeedepart.department_id,
    AVG(employee.salary + IFNULL(employeepayhist.bonus_amount, 0) +
IFNULL(employeepayhist.commission_amount, 0)) AS avg_compensation
FROM employee
JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id
GROUP BY employeedepart.department_id;
```

92. Employee Tenure vs. Compensation

Compare the average salary of employees based on their tenure. Which tenure range has the highest average salary?

```
SELECT

   CASE

      WHEN TIMESTAMPDIFF(YEAR, employee.hire_date, CURDATE()) BETWEEN 0 AND 1
THEN '0-1 year'

      WHEN TIMESTAMPDIFF(YEAR, employee.hire_date, CURDATE()) BETWEEN 2 AND 5
THEN '2-5 years'

      WHEN TIMESTAMPDIFF(YEAR, employee.hire_date, CURDATE()) BETWEEN 6 AND 10
THEN '6-10 years'

      ELSE '10+ years'

   END AS tenure_range,

   AVG(employee.salary) AS avg_salary

FROM employee

GROUP BY tenure_range

ORDER BY avg_salary DESC;
```

93. Salary Changes Over Time

Show salary changes over the last five years, including the rate change and date.

```
SELECT employee.employee_id,

      employee.name,

      employeepayhist.pay_date,

      employeepayhist.salary AS current_salary,

      LAG(employeepayhist.salary) OVER (PARTITION BY employee.employee_id ORDER BY
employeepayhist.pay_date) AS previous_salary,

      (employeepayhist.salary - LAG(employeepayhist.salary) OVER (PARTITION BY
employee.employee_id ORDER BY employeepayhist.pay_date)) /

      LAG(employeepayhist.salary) OVER (PARTITION BY employee.employee_id ORDER BY
employeepayhist.pay_date) * 100 AS salary_change_percentage
```

```
FROM employee

JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id

WHERE employeepayhist.pay_date >= CURDATE() - INTERVAL 5 YEAR;
```

94. Department Performance

Calculate the total compensation (including salary, bonuses, and commissions) for each department. Sort the departments by total compensation in descending order.

```
SELECT employeedepart.department_id,

    SUM(employee.salary + IFNULL(employeepayhist.bonus_amount, 0) +
IFNULL(employeepayhist.commission_amount, 0)) AS total_compensation

FROM employee

JOIN employeedepart ON employee.employee_id = employeedepart.employee_id

LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id

GROUP BY employeedepart.department_id

ORDER BY total_compensation DESC;
```

95. Employee Performance Metrics

Analyze the performance of employees by comparing their commission percentage against their sales quota and bonus in the last quarter.

```
SELECT employee.employee_id,

    employee.name,

    (IFNULL(employeepayhist.commission_amount, 0) / employeepayhist.sales_quota) * 100
AS commission_percentage,

    employeepayhist.bonus_amount

FROM employee

JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id
```

```
WHERE employeepayhist.pay_date > CURDATE() - INTERVAL 3 MONTH;
```

96. Top Performing Employees

Find the top 5 employees with the highest sales performance (based on sales YTD) and their corresponding compensation details.

```
SELECT employee.employee_id,

    employee.name,

    SUM(sales.sales_amount) AS sales_ytd,

    employee.salary + IFNULL(employeepayhist.bonus_amount, 0) +
IFNULL(employeepayhist.commission_amount, 0) AS total_compensation

FROM employee

JOIN sales ON employee.employee_id = sales.employee_id

LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id

WHERE sales.sale_date >= CURDATE() - INTERVAL 1 YEAR

GROUP BY employee.employee_id

ORDER BY sales_ytd DESC

LIMIT 5;
```

97. Employee Shift Analysis

Find the average compensation of employees working in different shifts (using the `ShiftID`). Sort by compensation.

```
SELECT employeedepart.shift_id,

    AVG(employee.salary + IFNULL(employeepayhist.bonus_amount, 0) +
IFNULL(employeepayhist.commission_amount, 0)) AS avg_compensation

FROM employee

JOIN employeedepart ON employee.employee_id = employeedepart.employee_id
```

```
LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id

GROUP BY employeedepart.shift_id

ORDER BY avg_compensation DESC;
```

98. Gender Pay Gap

Determine if there is a gender pay gap by comparing the average salary and total compensation between male and female employees.

```
SELECT employee.gender,

    AVG(employee.salary) AS avg_salary,

    SUM(employee.salary + IFNULL(employeepayhist.bonus_amount, 0) +
IFNULL(employeepayhist.commission_amount, 0)) AS total_compensation

FROM employee

LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id

GROUP BY employee.gender;
```

99. Employee Job Satisfaction

Correlate employee job satisfaction (assumed to be in the database) with their performance and compensation.

```
SELECT employee.employee_id,

    employee.name,

    employee.job_satisfaction,

    AVG(employee.salary + IFNULL(employeepayhist.bonus_amount, 0) +
IFNULL(employeepayhist.commission_amount, 0)) AS avg_compensation,

    AVG(employeepayhist.sales_quota) AS avg_sales_quota

FROM employee

JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id
```

```
GROUP BY employee.employee_id

ORDER BY employee.job_satisfaction DESC;
```

100. Employee Compensation by Location

Analyze the compensation structure (salary, bonuses, commissions) by different store locations. What are the locations with the highest and lowest compensation?

```
SELECT employee.store_location,

    AVG(employee.salary + IFNULL(employeepayhist.bonus_amount, 0) +
IFNULL(employeepayhist.commission_amount, 0)) AS avg_compensation

FROM employee

LEFT JOIN employeepayhist ON employee.employee_id = employeepayhist.employee_id

GROUP BY employee.store_location

ORDER BY avg_compensation DESC;
```