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;
```