```sql
-- 1. Retrieve all columns from the "employees" table.

SELECT * FROM employees;


-- 2. Retrieve only the "employee_id" and "first_name" columns from the "employees" table.

SELECT employee_id, first_name FROM employees;


-- 3. Filter employees who are in the "Sales" department.

SELECT * FROM employees e

JOIN departments d ON e.department_id = d.department_id

WHERE d.department_name = 'Sales';


-- 4. Filter employees who have a salary greater than $50,000.

SELECT * FROM employees WHERE salary > 50000;


-- 5. Sort employees by their hire date in ascending order.

SELECT * FROM employees ORDER BY hire_date ASC;


-- 6. Sort employees by their salary in descending order.

SELECT * FROM employees ORDER BY salary DESC;


-- 7. Calculate the average salary of all employees.

SELECT AVG(salary) AS average_salary FROM employees;


-- 8. Find the highest salary among all employees.

SELECT MAX(salary) AS highest_salary FROM employees;


-- 9. Find the number of employees in the company.

SELECT COUNT(*) AS total_employees FROM employees;


-- 10. Find employees with a first name starting with "J".

SELECT * FROM employees WHERE first_name LIKE 'J%';
```

```
-- 11. Retrieve employees and their corresponding departments using an inner join.

SELECT e.*, d.department_name

FROM employees e

INNER JOIN departments d ON e.department_id = d.department_id;


-- 12. Group employees by department and count the number of employees in each department.

SELECT d.department_name, COUNT(e.employee_id) AS num_employees

FROM employees e

JOIN departments d ON e.department_id = d.department_id

GROUP BY d.department_name;


-- 13. Filter employees who have been hired after January 1, 2020.

SELECT * FROM employees WHERE hire_date > '2020-01-01';


-- 14. Find the oldest employee in the company.

SELECT * FROM employees ORDER BY birth_date ASC LIMIT 1;


-- 15. Find employees with salaries between $40,000 and $60,000.

SELECT * FROM employees WHERE salary BETWEEN 40000 AND 60000;


-- 16. Count the number of employees hired in each year.

SELECT YEAR(hire_date) AS hire_year, COUNT(*) AS num_employees

FROM employees

GROUP BY hire_year;


-- 17. Calculate the total salary expenditure for the company.

SELECT SUM(salary) AS total_salary_expenditure FROM employees;


-- 18. Find employees who have changed their job title more than once using a self-join.

SELECT DISTINCT j1.employee_id
```

```
FROM job_history j1

JOIN job_history j2 ON j1.employee_id = j2.employee_id AND j1.from_date != j2.from_date;


-- 19. Filter employees who were hired in the last 6 months.

SELECT * FROM employees WHERE hire_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH);


-- 20. Find employees with salaries above the average salary.

SELECT * FROM employees WHERE salary > (SELECT AVG(salary) FROM employees);


-- 21. Calculate the average tenure of employees in each department.

SELECT d.department_name, AVG(DATEDIFF(CURDATE(), e.hire_date)) AS avg_tenure

FROM employees e

JOIN departments d ON e.department_id = d.department_id

GROUP BY d.department_name;


-- 22. Retrieve employees and their managers using a self-join.

SELECT e.employee_id, e.first_name, e.last_name, m.employee_id AS manager_id, m.first_name AS
manager_first_name, m.last_name AS manager_last_name

FROM employees e

LEFT JOIN employees m ON e.manager_id = m.employee_id;


-- 23. Find employees with the same first name.

SELECT first_name, COUNT(*) AS num_employees

FROM employees

GROUP BY first_name

HAVING COUNT(*) > 1;


-- 24. Count the number of employees per gender.

SELECT gender, COUNT(*) AS num_employees

FROM employees

GROUP BY gender;
```

```
-- 25. Filter employees who have a salary increase of more than 10% in the last year.
SELECT employee_id
FROM (
  SELECT employee_id,
       MAX(salary) AS current_salary,
       MIN(salary) AS previous_salary,
       (MAX(salary) - MIN(salary)) / MIN(salary) AS salary_increase_percentage
  FROM salaries
  WHERE YEAR(from_date) = YEAR(CURDATE()) - 1
  GROUP BY employee_id
) AS salary_changes
WHERE salary_increase_percentage > 0.10;

-- 26. Calculate the total number of working days for each employee.
SELECT j.employee_id, SUM(DATEDIFF(j.to_date, j.from_date)) AS total_working_days
FROM job_history j
GROUP BY j.employee_id;

-- 27. Find employees who have never changed their department using a join.
SELECT j.employee_id
FROM job_history j
LEFT JOIN departments d ON j.department_id = d.department_id
GROUP BY j.employee_id
HAVING COUNT(DISTINCT j.department_id) = 1;

-- 28. Retrieve employees who have a birthday in the current month.
SELECT * FROM employees WHERE MONTH(birth_date) = MONTH(CURDATE());

-- 29. Find employees who were hired in the same year they were born using a join.
SELECT e.*
```

```
FROM employees e
JOIN (
   SELECT employee_id
   FROM employees
   WHERE YEAR(hire_date) = YEAR(birth_date)
) AS same_year ON e.employee_id = same_year.employee_id;

-- 30. Calculate the total sum of salaries for each department.
SELECT d.department_name, SUM(e.salary) AS total_salary
FROM employees e
JOIN departments d ON e.department_id = d.department_id
GROUP BY d.department_name;

-- 31. Retrieve employees who have been promoted within the company using a self-join.
SELECT DISTINCT j1.employee_id
FROM job_history j1
JOIN job_history j2 ON j1.employee_id = j2.employee_id AND j1.from_date != j2.from_date;

-- 32. Find employees who have the same last name as their manager using a self-join.
SELECT e.employee_id, e.first_name, e.last_name, e.manager_id, m.last_name AS
manager_last_name
FROM employees e
INNER JOIN employees m ON e.manager_id = m.employee_id AND e.last_name = m.last_name;

-- 33. Filter employees who have a salary within 10% of the maximum salary.
SELECT *
FROM employees
WHERE salary BETWEEN (SELECT MAX(salary) * 0.9 FROM employees) AND (SELECT MAX(salary)
FROM employees);

-- 34. Calculate the average salary of male and female employees separately.
SELECT gender, AVG(salary) AS avg_salary
```

```
FROM employees
GROUP BY gender;

-- 35. Retrieve employees who have a salary increase in the last 6 months.
SELECT employee_id
FROM (
   SELECT employee_id,
         MAX(salary) AS current_salary,
         MIN(salary) AS previous_salary
   FROM salaries
   WHERE YEAR(from_date) = YEAR(CURDATE()) - 1
   GROUP BY employee_id
) AS salary_changes
WHERE current_salary > previous_salary;

-- 36. Find employees who have the same hire date.
SELECT hire_date, COUNT(*) AS num_employees
FROM employees
GROUP BY hire_date
HAVING COUNT(*) > 1;

-- 37. Retrieve employees who have not received a salary increase in the last year.
SELECT employee_id
FROM (
   SELECT employee_id,
         MAX(salary) AS current_salary,
         MIN(salary) AS previous_salary
   FROM salaries
   WHERE YEAR(from_date) = YEAR(CURDATE()) - 1
   GROUP BY employee_id
) AS salary_changes
```

```
WHERE current_salary = previous_salary;

-- 38. Calculate the total number of years of service for each employee using a join.
SELECT e.employee_id, DATEDIFF(CURDATE(), MIN(j.hire_date)) / 365 AS years_of_service
FROM employees e
JOIN job_history j ON e.employee_id = j.employee_id
GROUP BY e.employee_id;

-- 39. Find employees who have worked in multiple departments using a join.
SELECT j.employee_id
FROM job_history j
JOIN (
   SELECT employee_id, COUNT(DISTINCT department_id) AS num_departments
   FROM job_history
   GROUP BY employee_id
) AS department_counts ON j.employee_id = department_counts.employee_id
WHERE department_counts.num_departments > 1;

-- 40. Retrieve employees who have a salary below the minimum wage.
SELECT * FROM employees WHERE salary < (SELECT MIN(salary) FROM employees);

-- 41. Filter employees who were born before 1990 and have a salary above $60,000.
SELECT * FROM employees WHERE YEAR(birth_date) < 1990 AND salary > 60000;

-- 42. Retrieve employees who have a birthday on February 29th.
SELECT * FROM employees WHERE MONTH(birth_date) = 2 AND DAY(birth_date) = 29;

-- 43. Find employees who have worked in the company for more than 10 years.
SELECT * FROM employees WHERE DATEDIFF(CURDATE(), hire_date) > 365 * 10;

-- 44. Calculate the average salary for each job title.
```

```
SELECT job_title, AVG(salary) AS avg_salary

FROM employees

GROUP BY job_title;


-- 45. Retrieve employees who have a salary above the 90th percentile.

SELECT *

FROM employees

WHERE salary > (SELECT PERCENTILE_CONT(0.9) WITHIN GROUP (ORDER BY salary) FROM
employees);


-- 46. Find employees who have the same hire date and salary.

SELECT hire_date, salary, COUNT(*) AS num_employees

FROM employees

GROUP BY hire_date, salary

HAVING COUNT(*) > 1;


-- 47. Filter employees who have received a bonus in the last quarter using a join.

SELECT b.employee_id

FROM bonuses b

JOIN employees e ON b.employee_id = e.employee_id

WHERE b.quarter = 'Q4' AND b.year = YEAR(CURDATE());


-- 48. Retrieve employees who have changed their job title more than once in the last year using a
join.

SELECT DISTINCT j1.employee_id

FROM job_history j1

JOIN job_history j2 ON j1.employee_id = j2.employee_id AND j1.from_date != j2.from_date

WHERE YEAR(j1.from_date) = YEAR(CURDATE()) - 1;


-- 49. Find employees who have a salary decrease in the last 6 months.

SELECT employee_id

FROM (
  SELECT employee_id,
        MAX(salary) AS current_salary,
        MIN(salary) AS previous_salary
  FROM salaries
  WHERE YEAR(from_date) = YEAR(CURDATE()) AND MONTH(from_date) >
MONTH(DATE_SUB(CURDATE(), INTERVAL 6 MONTH))
  GROUP BY employee_id
) AS salary_changes
WHERE current_salary < previous_salary;


-- 50. Calculate the total number of months worked for each employee using a join.
SELECT j.employee_id, SUM(DATEDIFF(j.to_date, j.from_date) / 30) AS total_months_worked
FROM job_history j
GROUP BY j.employee_id;
```