1. Retrieve the first name, last name, and salary of employees earning more than the average salary.
2. Count the number of employees in each department where the department name starts with 'M'.
3. Retrieve employees who do not have an email address (email is NULL).
4. Calculate the total salary paid to employees in each department.
5. Update the department of employees whose salary is less than 30,000 to 'Trainee'.
6. Delete employees whose joining date is before January 1, 2020.
7. Retrieve the top 5 highest-paid employees.
8. Find the names of employees who joined in the same month, regardless of the year.
9. Add a new column phone\_number to the employee2 table with a data type of VARCHAR(15).
10. Update the phone\_number of all employees to '1234567890'.
11. Find employees whose first name ends with the letter 'n'.
12. Retrieve employees whose age is between 25 and 35, and salary is greater than 50,000.
13. Add a constraint to ensure that the email column in employee2 is unique.
14. Find employees who are in the 'HR' or 'Finance' department.
15. Retrieve the second highest salary in the employee2 table.
16. Create a view named high\_salary\_employees that retrieves employees with a salary above 70,000.
17. Insert a new record into employee2 with NULL for the age and email columns.
18. Retrieve employees who joined in the last 6 months from today’s date.
19. Update the salary of employees earning the minimum salary to 40,000.
20. Find employees whose last names contain exactly 5 characters.
21. Retrieve employees sorted first by department name (ascending) and then by salary (descending).
22. Find the department with the maximum number of employees.
23. Calculate the difference in days between today’s date and each employee’s joining date.
24. Retrieve employees whose first and last names are identical (e.g., first\_name = last\_name).
25. Create a backup table named employee2\_backup and copy all the data from employee2.
26. Retrieve all employees who do not belong to the 'IT' department.
27. Find employees whose salary is a multiple of 5000.
28. Delete duplicate rows in the employee2 table based on first name and last name.
29. Update the email column of all employees to include their department (e.g., john.doe@it.example.com).
30. Create an index on the salary column to improve query performance for salary-based searches.

-- 1. Retrieve all employees’ first\_name and their departments.

SELECT first\_name, department FROM employees;

-- 2. Update the salary of all employees in the 'IT' department by increasing it by 10%.

UPDATE employees SET salary = salary \* 1.10 WHERE department = 'IT';

-- 3. Delete all employees who are older than 34 years.

DELETE FROM employees WHERE age > 34;

-- 4. Retrieve the names of employees who joined after January 1, 2021.

SELECT first\_name FROM employees WHERE joining\_date > '2021-01-01';

-- 5. Change the data type of the `salary` column to `INTEGER`.

ALTER TABLE employees ALTER COLUMN salary TYPE INTEGER;

-- 6. List all employees with their age and salary in descending order of salary.

SELECT first\_name, age, salary FROM employees ORDER BY salary DESC;

-- 7. Insert a new employee with the following details: (employee\_id, 'Raj', 'Singh', 'Marketing', 60000, '2023-09-15', 30, email).

INSERT INTO employees (employee\_id, first\_name, last\_name, department, salary, joining\_date, age, email)

VALUES (employee\_id, 'Raj', 'Singh', 'Marketing', 60000, '2023-09-15', 30, email);

-- 8. Update age of employee +1 to every employee.

UPDATE employees SET age = age + 1;

-- 9. Find all employees who joined after 2022-01-01.

SELECT \* FROM employees WHERE joining\_date > '2022-01-01';

-- 10. Find the average salary of employees in each department.

SELECT department, AVG(salary) AS average\_salary FROM employees GROUP BY department;

-- 11. Find employees who are older than 30 years.

SELECT \* FROM employees WHERE age > 30;

-- 12. Find the highest salary in the IT department.

SELECT MAX(salary) AS highest\_salary FROM employees WHERE department = 'IT';

-- 13. Update the email column for all employees with appropriate values.

UPDATE employees SET email = CONCAT(first\_name, '.', last\_name, '@example.com');

-- 14. Find the total number of employees in each department.

SELECT department, COUNT(\*) AS total\_employees FROM employees GROUP BY department;

-- 15. Sort employees by their joining\_date from the newest to the oldest.

SELECT \* FROM employees ORDER BY joining\_date DESC;

-- 16. Retrieve employees whose salary is between 50,000 and 70,000.

SELECT \* FROM employees WHERE salary BETWEEN 50000 AND 70000;

-- 17. Find employees who have 'a' in their first name.

SELECT \* FROM employees WHERE first\_name LIKE '%a%';

-- 18. Count the total number of employees in the table.

SELECT COUNT(\*) AS total\_employees FROM employees;

-- 19. Retrieve employees grouped by their department, sorted by department name.

SELECT \* FROM employees ORDER BY department;

-- 20. Find employees who joined in the year 2023.

SELECT \* FROM employees WHERE EXTRACT(YEAR FROM joining\_date) = 2023;

-- 21. Retrieve the minimum salary in the company.

SELECT MIN(salary) AS minimum\_salary FROM employees;

-- 22. Retrieve employees whose age is NULL.

SELECT \* FROM employees WHERE age IS NULL;

-- 23. Retrieve the first name, last name, and salary of employees earning more than the average salary.

SELECT first\_name, last\_name, salary FROM employees

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

-- 24. Count the number of employees in each department where the department name starts with 'M'.

SELECT department, COUNT(\*) AS total\_employees FROM employees

WHERE department LIKE 'M%' GROUP BY department;

-- 25. Retrieve employees who do not have an email address (email is NULL).

SELECT \* FROM employees WHERE email IS NULL;

-- 26. Calculate the total salary paid to employees in each department.

SELECT department, SUM(salary) AS total\_salary FROM employees GROUP BY department;

-- 27. Update the department of employees whose salary is less than 30,000 to 'Trainee'.

UPDATE employees SET department = 'Trainee' WHERE salary < 30000;

-- 28. Delete employees whose joining date is before January 1, 2020.

DELETE FROM employees WHERE joining\_date < '2020-01-01';

-- 29. Retrieve the top 5 highest-paid employees.

SELECT \* FROM employees ORDER BY salary DESC LIMIT 5;

-- 30. Find the names of employees who joined in the same month, regardless of the year.

SELECT first\_name, last\_name FROM employees

WHERE EXTRACT(MONTH FROM joining\_date) = EXTRACT(MONTH FROM CURRENT\_DATE);

-- 31. Add a new column phone\_number to the employee2 table with a data type of VARCHAR(15).

ALTER TABLE employee2 ADD COLUMN phone\_number VARCHAR(15);

-- 32. Update the phone\_number of all employees to '1234567890'.

UPDATE employee2 SET phone\_number = '1234567890';

-- 33. Find employees whose first name ends with the letter 'n'.

SELECT \* FROM employees WHERE first\_name LIKE '%n';

-- 34. Retrieve employees whose age is between 25 and 35, and salary is greater than 50,000.

SELECT \* FROM employees WHERE age BETWEEN 25 AND 35 AND salary > 50000;

-- 35. Add a constraint to ensure that the email column in employee2 is unique.

ALTER TABLE employee2 ADD CONSTRAINT unique\_email UNIQUE (email);

-- 36. Find employees who are in the 'HR' or 'Finance' department.

SELECT \* FROM employees WHERE department IN ('HR', 'Finance');

-- 37. Retrieve the second highest salary in the employee2 table.

SELECT MAX(salary) AS second\_highest\_salary FROM employee2

WHERE salary < (SELECT MAX(salary) FROM employee2);

-- 38. Create a view named high\_salary\_employees that retrieves employees with a salary above 70,000.

CREATE VIEW high\_salary\_employees AS

SELECT \* FROM employees WHERE salary > 70000;

-- 39. Insert a new record into employee2 with NULL for the age and email columns.

INSERT INTO employee2 (employee\_id, first\_name, last\_name, department, salary, joining\_date, age, email)

VALUES (employee\_id, 'First', 'Last', 'Department', 50000, '2025-01-01', NULL, NULL);

-- 40. Retrieve employees who joined in the last 6 months from today’s date.

SELECT \* FROM employees WHERE joining\_date >= CURRENT\_DATE - INTERVAL '6 months';

-- 41. Update the salary of employees earning the minimum salary to 40,000.

UPDATE employees SET salary = 40000 WHERE salary = (SELECT MIN(salary) FROM employees);

-- 42. Find employees whose last names contain exactly 5 characters.

SELECT \* FROM employees WHERE LENGTH(last\_name) = 5;

-- 43. Retrieve employees sorted first by department name (ascending) and then by salary (descending).

SELECT \* FROM employees ORDER BY department ASC, salary DESC;

-- 44. Find the department with the maximum number of employees.

SELECT department, COUNT(\*) AS total\_employees FROM employees

GROUP BY department ORDER BY total\_employees DESC LIMIT 1;

-- 45. Calculate the difference in days between today’s date and each employee’s joining date.

SELECT first\_name, last\_name, (CURRENT\_DATE - joining\_date) AS days\_with\_company FROM employees;

-- 46. Retrieve employees whose first and last names are identical (e.g., first\_name = last\_name).

SELECT \* FROM employees WHERE first\_name = last\_name;

-- 47. Create a backup table named employee2\_backup and copy all the data from employee2.

CREATE TABLE employee2\_backup AS TABLE employee2;

-- 48. Retrieve all employees who do not belong to the 'IT' department.

SELECT \* FROM employees WHERE department != 'IT';

-- 49. Find employees whose salary is a multiple of 5000.

SELECT \* FROM employees WHERE salary % 5000 = 0;

-- 50. Delete duplicate rows in the employee2 table based on first name and last name.

DELETE FROM employee2 WHERE id NOT IN (

  SELECT MIN(id) FROM employee2 GROUP BY first\_name, last\_name

);

-- 51. Update the email column of all employees to include their department (e.g., john.doe@it.example.com).

UPDATE employees SET email = CONCAT(LOWER(first\_name), '.', LOWER(last\_name), '@', LOWER(department), '.example.com');

-- 52. Create an index on the salary column to improve query performance for salary-based searches.

CREATE INDEX idx\_salary ON employees (salary);