SECTION A - BASIC CONCEPTS

-- 1. Write a command to create a database named company\_db.

CREATE DATABASE company\_db;

-- 2. Create a table employees & departments with the following structure:

CREATE TABLE departments (

dept\_id SERIAL PRIMARY KEY,

dept\_name VARCHAR(100) NOT NULL

);

CREATE TABLE employees (

employee\_id SERIAL PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

salary DECIMAL(10,2) CHECK (salary > 0),

hire\_date DATE NOT NULL,

dept\_id INT REFERENCES departments(dept\_id) ON DELETE SET NULL

);

-- SECTION B - DATA MANIPULATION

-- 1. Insert the following records into the employee’s table:

-- imported the data by import feature

-- 2. Write a query to increase the salary of all employees by 10%.

UPDATE employees

SET salary = salary \* 1.10;

-- 3. Write a query to delete employees hired before 2022.

DELETE FROM employees

WHERE hire\_date < '2022-01-01';

-- 4. Retrieve all employees who earn between 50000 and 80000.

SELECT \* FROM employees

WHERE salary BETWEEN 50000 AND 80000;

-- 5. Select employees whose first name starts with 'J' using the LIKE operator.

SELECT \* FROM employees

WHERE first\_name LIKE 'J%';

-- SECTION C: SORTING & AGGREGATION

-- 1. Retrieve all employees and sort them by salary in descending order.

SELECT \* FROM employees

ORDER BY salary DESC;

-- 2. Retrieve the top 3 highest-paid employees.

SELECT employee\_id, first\_name, last\_name, salary

FROM employees

ORDER BY salary DESC

LIMIT 3;

-- 3. Find the total salary expense of the company.

SELECT SUM(salary) AS total\_salary\_expense

FROM employees;

-- 4. Find the average salary of employees and filter only those with an

-- average salary greater than 70000 using the HAVING clause.

SELECT dept\_id, AVG(salary) AS avg\_salary

FROM employees

GROUP BY dept\_id

HAVING AVG(salary) > 70000;

-- Section D: STRING & DATE FUNCTIONS

-- 1. Concatenate the first\_name and last\_name with a space in between.

SELECT CONCAT(first\_name, ' ', last\_name) AS full\_name

FROM employees;

-- 2. Extract the year from the hire\_date column.

SELECT employee\_id, first\_name, last\_name, EXTRACT(YEAR FROM hire\_date) AS hire\_year

FROM employees;

-- 3. Convert all first\_name values to uppercase.

SELECT employee\_id, UPPER(first\_name) AS first\_name\_upper, last\_name, salary, hire\_date, dept\_id

FROM employees;

-- 4. Find the difference in years between the current date and hire\_date.

SELECT employee\_id, first\_name, last\_name, hire\_date,

AGE(CURRENT\_DATE, hire\_date) AS experience

FROM employees;

-- 5. Use DATE\_TRUNC to round off hire\_date to the nearest month.

SELECT employee\_id, first\_name, last\_name, hire\_date,

DATE\_TRUNC('month', hire\_date) AS rounded\_hire\_date

FROM employees;

-- SECTION E: ADVANCED FILTERING & CONDITIONAL LOGIC

-- 1. Use COALESCE to replace NULL salaries with 50000.

SELECT employee\_id, first\_name, last\_name,

COALESCE(salary, 50000) AS updated\_salary

FROM employees;

-- 2. Find the highest and lowest salaries using GREATEST and LEAST.

SELECT

GREATEST(MAX(salary)) AS highest\_salary,

LEAST(MIN(salary)) AS lowest\_salary

FROM employees;

-- 3. Use NULLIF to prevent division by zero when calculating salary percentages.

SELECT

employee\_id,

first\_name,

last\_name,

salary,

(salary / NULLIF((SELECT SUM(salary) FROM employees), 0)) \* 100 AS salary\_percentage

FROM employees;

-- SECTION F: JOINS & SET OPERATIONS

-- 1. Given a departments table with dept\_id and dept\_name, write a

-- query to join employees and departments on dept\_id using an

-- INNER JOIN.

SELECT

e.employee\_id,

e.first\_name,

e.last\_name,

e.salary,

e.hire\_date,

d.dept\_id,

d.dept\_name

FROM employees e

INNER JOIN departments d ON e.dept\_id = d.dept\_id;

-- 2. Retrieve all employees, ensuring that those without a department

-- are also included (use LEFT JOIN).

SELECT

e.employee\_id,

e.first\_name,

e.last\_name,

e.salary,

e.hire\_date,

d.dept\_id,

d.dept\_name

FROM employees e

LEFT JOIN departments d ON e.dept\_id = d.dept\_id;

-- 3. Find employees who are not present in another table using EXCEPT.

SELECT employee\_id, first\_name, last\_name, dept\_id

FROM employees

EXCEPT

SELECT employee\_id, first\_name, last\_name, dept\_id

FROM employees

WHERE dept\_id IN (SELECT dept\_id FROM departments);