LabWork 4

-- Task 1.1

SELECT

first\_name || ' ' || last\_name AS full\_name,

department,

salary

FROM employees;

-- Task 1.2

SELECT DISTINCT department

FROM employees;

-- Task 1.3

SELECT

project\_name,

budget,

CASE

WHEN budget > 150000 THEN 'Large'

WHEN budget BETWEEN 100000 AND 150000 THEN 'Medium'

ELSE 'Small'

END AS budget\_category

FROM projects;

-- Task 1.4

SELECT

first\_name || ' ' || last\_name AS full\_name,

COALESCE(email, 'No email provided') AS email\_display

FROM employees;

-- Task 2.1

SELECT \* FROM employees

WHERE hire\_date > '2020-01-01';

-- Task 2.2

SELECT \* FROM employees

WHERE salary BETWEEN 60000 AND 70000;

-- Task 2.3

SELECT \* FROM employees

WHERE last\_name LIKE 'S%' OR last\_name LIKE 'J%';

-- Task 2.4

SELECT \* FROM employees

WHERE manager\_id IS NOT NULL AND department = 'IT';

-- Task 3.1

SELECT

UPPER(first\_name || ' ' || last\_name) AS upper\_name,

LENGTH(last\_name) AS last\_name\_length,

SUBSTRING(email FROM 1 FOR 3) AS email\_prefix

FROM employees;

-- Task 3.2

SELECT

first\_name || ' ' || last\_name AS full\_name,

salary \* 12 AS annual\_salary,

ROUND(salary / 12, 2) AS monthly\_salary,

salary \* 0.10 AS raise\_amount

FROM employees;

-- Task 3.3

SELECT

FORMAT('Project: %s - Budget: $%s - Status: %s', project\_name, budget, status) AS project\_info

FROM projects;

-- Task 3.4

SELECT

first\_name || ' ' || last\_name AS full\_name,

DATE\_PART('year', AGE(CURRENT\_DATE, hire\_date)) AS years\_with\_company

FROM employees;

-- Task 4.1

SELECT department, AVG(salary) AS avg\_salary

FROM employees

GROUP BY department;

-- Task 4.2

SELECT

p.project\_name,

SUM(a.hours\_worked) AS total\_hours

FROM assignments a

JOIN projects p ON a.project\_id = p.project\_id

GROUP BY p.project\_name;

-- Task 4.3

SELECT

department,

COUNT(\*) AS employee\_count

FROM employees

GROUP BY department

HAVING COUNT(\*) > 1;

-- Task 4.4

SELECT

MAX(salary) AS max\_salary,

MIN(salary) AS min\_salary,

SUM(salary) AS total\_payroll

FROM employees;

-- Task 5.1

(SELECT employee\_id, first\_name || ' ' || last\_name AS full\_name, salary

FROM employees

WHERE salary > 65000)

UNION

(SELECT employee\_id, first\_name || ' ' || last\_name AS full\_name, salary

FROM employees

WHERE hire\_date > '2020-01-01');

-- Task 5.2

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

FROM employees

WHERE department = 'IT'

INTERSECT

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

FROM employees

WHERE salary > 65000;

-- Task 5.3

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

FROM employees

EXCEPT

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

FROM employees e

JOIN assignments a ON e.employee\_id = a.employee\_id;

-- Task 6.1

SELECT \*

FROM employees e

WHERE EXISTS (

SELECT 1 FROM assignments a

WHERE a.employee\_id = e.employee\_id

);

-- Task 6.2

SELECT \*

FROM employees

WHERE employee\_id IN (

SELECT a.employee\_id

FROM assignments a

JOIN projects p ON a.project\_id = p.project\_id

WHERE p.status = 'Active'

);

-- Task 6.3

SELECT \*

FROM employees

WHERE salary > ANY (

SELECT salary FROM employees WHERE department = 'Sales'

);

-- Task 7.1

SELECT

e.first\_name || ' ' || e.last\_name AS employee\_name,

e.department,

AVG(a.hours\_worked) AS avg\_hours,

e.salary,

RANK() OVER (PARTITION BY e.department ORDER BY e.salary DESC) AS dept\_rank

FROM employees e

LEFT JOIN assignments a ON e.employee\_id = a.employee\_id

GROUP BY e.employee\_id, e.first\_name, e.last\_name, e.department, e.salary;

-- Task 7.2

SELECT

p.project\_name,

SUM(a.hours\_worked) AS total\_hours,

COUNT(DISTINCT a.employee\_id) AS employee\_count

FROM projects p

JOIN assignments a ON p.project\_id = a.project\_id

GROUP BY p.project\_name

HAVING SUM(a.hours\_worked) > 150;

-- Task 7.3

SELECT

e.department,

COUNT(e.employee\_id) AS total\_employees,

AVG(e.salary) AS avg\_salary,

(SELECT first\_name || ' ' || last\_name

FROM employees e2

WHERE e2.department = e.department

ORDER BY e2.salary DESC

LIMIT 1) AS highest\_paid\_employee,

GREATEST(MAX(salary), MIN(salary)) AS salary\_range\_top,

LEAST(MAX(salary), MIN(salary)) AS salary\_range\_bottom

FROM employees e

GROUP BY e.department;