DROP TABLE IF EXISTS assignments, employees, projects CASCADE;  
  
  
CREATE TABLE employees  
(  
 employee\_id SERIAL PRIMARY KEY,  
 first\_name VARCHAR(50),  
 last\_name VARCHAR(50),  
 department VARCHAR(50),  
 salary NUMERIC(10, 2),  
 hire\_date DATE,  
 manager\_id INTEGER,  
 email VARCHAR(100)  
);  
  
CREATE TABLE projects  
(  
 project\_id SERIAL PRIMARY KEY,  
 project\_name VARCHAR(100),  
 budget NUMERIC(12, 2),  
 start\_date DATE,  
 end\_date DATE,  
 status VARCHAR(20)  
);  
  
CREATE TABLE assignments  
(  
 assignment\_id SERIAL PRIMARY KEY,  
 employee\_id INTEGER REFERENCES employees (employee\_id),  
 project\_id INTEGER REFERENCES projects (project\_id),  
 hours\_worked NUMERIC(5, 1),  
 assignment\_date DATE  
);  
  
INSERT INTO employees (first\_name, last\_name, department, salary, hire\_date, manager\_id, email) VALUES  
('John', 'Smith', 'IT', 75000, '2020-01-15', NULL, 'john.smith@company.com'),  
('Sarah', 'Johnson', 'IT', 65000, '2020-03-20', 1, 'sarah.j@company.com'),  
('Michael', 'Brown', 'Sales', 55000, '2019-06-10', NULL, 'mbrown@company.com'),  
('Emily', 'Davis', 'HR', 60000, '2021-02-01', NULL, 'emily.davis@company.com'),  
('Robert', 'Wilson', 'IT', 70000, '2020-08-15', 1, NULL),  
('Lisa', 'Anderson', 'Sales', 58000, '2021-05-20', 3, 'lisa.a@company.com');  
  
INSERT INTO projects (project\_name, budget, start\_date, end\_date, status) VALUES  
('Website Redesign', 150000, '2024-01-01', '2024-06-30', 'Active'),  
('CRM Implementation', 200000, '2024-02-15', '2024-12-31', 'Active'),  
('Marketing Campaign', 80000, '2024-03-01', '2024-05-31', 'Completed'),  
('Database Migration', 120000, '2024-01-10', NULL, 'Active');  
  
INSERT INTO assignments (employee\_id, project\_id, hours\_worked, assignment\_date) VALUES  
(1, 1, 120.5, '2024-01-15'),  
(2, 1, 95.0, '2024-01-20'),  
(1, 4, 80.0, '2024-02-01'),  
(3, 3, 60.0, '2024-03-05'),  
(5, 2, 110.0, '2024-02-20'),  
(6, 3, 75.5, '2024-03-10');  
  
-- 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 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 uppercase\_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 AS annual\_salary, *ROUND*(salary / 12, 2) AS monthly\_salary, salary \* 0.1 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, *EXTRACT*(YEAR FROM *AGE*(*CURRENT\_DATE*, hire\_date)) AS years\_with\_company FROM employees;

-- Task 4.1  
SELECT department, *ROUND*(*AVG*(salary), 2) AS average\_salary FROM employees GROUP BY department;

-- Task 4.2  
SELECT p.project\_name, *SUM*(a.hours\_worked) AS total\_hours\_worked FROM projects p JOIN assignments a ON p.project\_id = a.project\_id GROUP BY p.project\_id, 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 DISTINCT 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, *ROUND*(*AVG*(a.hours\_worked), 2) AS avg\_hours\_worked, e.salary, (SELECT *COUNT*(\*) + 1 FROM employees e2 WHERE e2.department = e.department AND e2.salary > e.salary) AS salary\_rank\_in\_dept 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 ORDER BY e.department, salary\_rank\_in\_dept;

-- Task 7.2  
SELECT p.project\_name, *SUM*(a.hours\_worked) AS total\_hours, *COUNT*(DISTINCT a.employee\_id) AS employees\_assigned FROM projects p JOIN assignments a ON p.project\_id = a.project\_id GROUP BY p.project\_id, p.project\_name HAVING *SUM*(a.hours\_worked) > 150;

-- Task 7.3  
SELECT e.department, *COUNT*(\*) AS total\_employees, *ROUND*(*AVG*(e.salary), 2) AS average\_salary, *MAX*(e.salary) AS highest\_salary, (SELECT first\_name || ' ' || last\_name FROM employees WHERE department = e.department ORDER BY salary DESC LIMIT 1) AS highest\_paid\_employee, *GREATEST*(*MAX*(e.salary), 100000) AS adjusted\_max\_salary, *LEAST*(*MIN*(e.salary), 50000) AS adjusted\_min\_salary FROM employees e GROUP BY e.department;





























