Laboratory Work 4

Lab Assignment: SQL Queries, Functions, and Operators

Objective

This laboratory work will help you practice writing SQL queries using various clauses, operators, and functions covered in the lectures. You'll work with a sample database and progressively build more complex queries.

Database Schema

For this lab, we'll use the following tables:

```
-- Create tables
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 sample data
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');
```

Part 1: Basic SELECT Queries

Task 1.1: Write a query to select all employees, displaying their full name (concatenated first and last name), department, and salary.

Task 1.2: Use `SELECT DISTINCT` to find all unique departments in the company.

Task 1.3: Select all projects with their names and budgets, and create a new column called `budget_category` using a CASE expression:

- 'Large' if budget > 150000

- 'Medium' if budget between 100000 and 150000
- 'Small' otherwise

Task 1.4: Write a query using `COALESCE` to display employee names and their emails. If email is NULL, display 'No email provided'.

Part 2: WHERE Clause and Comparison Operators

Task 2.1: Find all employees hired after January 1, 2020.

Task 2.2: Find all employees whose salary is between 60000 and 70000 (use the `BETWEEN` operator).

Task 2.3: Find all employees whose last name starts with 'S' or 'J' (use the `LIKE` operator).

Task 2.4: Find all employees who have a manager (manager_id IS NOT NULL) and work in the IT department.

Part 3: String and Mathematical Functions

Task 3.1: Create a query that displays:

- Employee names in uppercase
- Length of their last names
- First 3 characters of their email address (use `substring`)

Task 3.2: Calculate the following for each employee:

- Annual salary
- Monthly salary (rounded to 2 decimal places)
- A 10% raise amount (use mathematical operators)

Task 3.3: Use the `format()` function to create a formatted string for each project: `"Project: [name] - Budget: $[budget] - Status: [status]"`

Task 3.4: Calculate how many years each employee has been with the company (use date functions and the current date).

Part 4: Aggregate Functions and GROUP BY

Task 4.1: Calculate the average salary for each department.

Task 4.2: Find the total hours worked on each project, including the project name.

Task 4.3: Count the number of employees in each department. Only show departments with more than 1 employee (use `HAVING`).

Task 4.4: Find the maximum and minimum salary in the company, along with the total payroll (sum of all salaries).

Part 5: Set Operations

Task 5.1: Write two queries and combine them using `UNION`:

- Query 1: Employees with salary > 65000
- Query 2: Employees hired after 2020-01-01 Display employee_id, full name, and salary.

Task 5.2: Use `INTERSECT` to find employees who work in IT AND have a salary greater than 65000.

Task 5.3: Use `EXCEPT` to find all employees who are NOT assigned to any projects.

Part 6: Subqueries

Task 6.1: Use `EXISTS` to find all employees who have at least one project assignment.

Task 6.2: Use `IN` with a subquery to find all employees working on projects with status 'Active'.

Task 6.3: Use `ANY` to find employees whose salary is greater than ANY employee in the Sales department.

Part 7: Complex Queries

Task 7.1: Create a query that shows:

- Employee name
- Their department
- Average hours worked across all their assignments
- Their rank within their department by salary (use ORDER BY if window functions not covered)

Task 7.2: Find projects where the total hours worked exceeds 150 hours. Display project name, total hours, and number of employees assigned.

Task 7.3: Create a report showing departments with their:

- Total number of employees
- Average salary
- Highest paid employee name Use `GREATEST` and `LEAST` functions somewhere in this query.

Submission Requirements

1. Create a single `.sql` file with all your queries
2. Each query should be clearly labeled with comments (e.g., `-- Task 1.1`)
3. Test all queries to ensure they execute without errors

4. Upload your .sql file to your GitHub repo.

Good luck with your laboratory work!