# SQL Practice Tasks - From Basic to Advanced

## Sample Tables

```sql
-- Employees table
CREATE TABLE employees (
    emp_id INT,
    name VARCHAR(50),
    salary DECIMAL(10,2),
    department_id INT,
    manager_id INT,
    hire_date DATE
);

-- Departments table
CREATE TABLE departments (
    dept_id INT,
    dept_name VARCHAR(50),
    location VARCHAR(50)
);

-- Projects table
CREATE TABLE projects (
    project_id INT,
    project_name VARCHAR(50),
    budget DECIMAL(10,2),
    dept_id INT
);

-- Employee_Projects table
CREATE TABLE employee_projects (
    emp_id INT,
    project_id INT,
    hours_worked DECIMAL(10,2)
);
```

## Task 1 - Basic COUNT and NULL Handling (Easy)

**Task:** Find the total number of employees and how many employees have a salary value.

```sql
SELECT
    COUNT(*) as total_employees,
    COUNT(salary) as employees_with_salary
FROM employees;
```

**Skills:** Basic COUNT, understanding NULL

### Task 2 - Simple Arithmetic and ROUND (Easy)

**Task:** Calculate the monthly salary (annual salary divided by 12) for each employee, rounded to 2 decimal places.

```sql
SELECT
    name,
    salary as annual_salary,
    ROUND(salary/12, 2) as monthly_salary
FROM employees;
```

**Skills:** ROUND, basic arithmetic

### Task 3 - Basic JOIN and GROUP BY (Easy-Medium)

**Task:** Show the number of employees in each department, including department name.

```sql
SELECT
    d.dept_name,
    COUNT(e.emp_id) as employee_count
FROM departments d
LEFT JOIN employees e ON d.dept_id = e.department_id
GROUP BY d.dept_name;
```

**Skills:** LEFT JOIN, GROUP BY, COUNT

### Task 4 - Multiple JOINs with NULL Handling (Medium)

**Task:** Find all projects and their assigned employees. For projects with no employees or employees with no hours logged, show 0.

```sql
SELECT
    p.project_name,
    e.name,
    COALESCE(ep.hours_worked, 0) as hours_worked
FROM projects p
LEFT JOIN employee_projects ep ON p.project_id = ep.project_id
LEFT JOIN employees e ON ep.emp_id = e.emp_id;
```

**Skills:** Multiple LEFT JOINs, COALESCE

### Task 5 - Self Join with Aggregation (Medium)

**Task:** For each employee, show their name, their manager's name, and their manager's total number of direct reports.

```sql
SELECT
    e1.name as employee_name,
    e2.name as manager_name,
    COUNT(e3.emp_id) as manager_direct_reports
FROM employees e1
LEFT JOIN employees e2 ON e1.manager_id = e2.emp_id
LEFT JOIN employees e3 ON e2.emp_id = e3.manager_id
GROUP BY e1.name, e2.name;
```

**Skills:** Self JOIN, Multiple JOINs, GROUP BY

### Task 6 - Subquery with Set Theory (Medium-Hard)

**Task:** Find all departments that have employees but no active projects.

```sql
SELECT DISTINCT d.dept_name
FROM departments d
INNER JOIN employees e ON d.dept_id = e.department_id
WHERE d.dept_id NOT IN (
    SELECT dept_id
    FROM projects
    WHERE budget > 0
);
```

**Skills:** Subquery, NOT IN, DISTINCT

### Task 7 - Complex Aggregation with Multiple JOINs (Hard)

**Task:** Calculate the percentage of total project budget consumed by each department, considering employee hours worked and their hourly rate (salary/2080).

```
WITH employee_costs AS (
    SELECT
        p.project_id,
        p.dept_id,
        SUM(ep.hours_worked * (e.salary/2080)) as labor_cost
    FROM projects p
    JOIN employee_projects ep ON p.project_id = ep.project_id
    JOIN employees e ON ep.emp_id = e.emp_id
    GROUP BY p.project_id, p.dept_id
)
SELECT
    d.dept_name,
    ROUND(SUM(ec.labor_cost) * 100.0 / SUM(p.budget), 2) as
budget_utilization_percentage
FROM departments d
JOIN projects p ON d.dept_id = p.dept_id
JOIN employee_costs ec ON p.project_id = ec.project_id
GROUP BY d.dept_name;
```

**Skills:** CTEs, Complex calculations, Multiple JOINs, Window functions

### Task 8 - Semi Join with Complex Conditions (Hard)

**Task:** Find employees who have worked on all projects in their department.

```
SELECT e.name
FROM employees e
WHERE NOT EXISTS (
    SELECT p.project_id
    FROM projects p
    WHERE p.dept_id = e.department_id
    AND NOT EXISTS (
        SELECT 1
        FROM employee_projects ep
        WHERE ep.project_id = p.project_id
```

```
        AND ep.emp_id = e.emp_id
    )
);
```

**Skills:** Semi joins, Nested NOT EXISTS

### Task 9 - Full Join with Conditional Aggregation (Hard)

**Task:** Create a department summary showing total salary budget, total project budget, number of employees, and number of projects, including departments with no employees or projects.

```
SELECT
    d.dept_name,
    COUNT(DISTINCT e.emp_id) as employee_count,
    COUNT(DISTINCT p.project_id) as project_count,
    COALESCE(SUM(e.salary), 0) as salary_budget,
    COALESCE(SUM(p.budget), 0) as project_budget
FROM departments d
FULL JOIN employees e ON d.dept_id = e.department_id
FULL JOIN projects p ON d.dept_id = p.dept_id
GROUP BY d.dept_name;
```

**Skills:** FULL JOIN, Conditional aggregation, COALESCE

### Task 10 - Advanced Analysis with All Concepts (Very Hard)

**Task:** Find departments where the average employee salary is higher than the department's project budget per employee, and show the salary to budget ratio, but only include departments where all employees are assigned to at least 2 projects.

```
WITH dept_metrics AS (
    SELECT
        d.dept_id,
        d.dept_name,
        AVG(e.salary) as avg_salary,
        SUM(p.budget) / NULLIF(COUNT(DISTINCT e.emp_id), 0) as
budget_per_employee,
        COUNT(DISTINCT e.emp_id) as emp_count
    FROM departments d
```

```
    JOIN employees e ON d.dept_id = e.department_id
    LEFT JOIN projects p ON d.dept_id = p.dept_id
    GROUP BY d.dept_id, d.dept_name
),
employee_project_counts AS (
    SELECT
        e.department_id,
        e.emp_id,
        COUNT(DISTINCT ep.project_id) as project_count
    FROM employees e
    LEFT JOIN employee_projects ep ON e.emp_id = ep.emp_id
    GROUP BY e.department_id, e.emp_id
)
SELECT
    dm.dept_name,
    ROUND(dm.avg_salary, 2) as avg_salary,
    ROUND(dm.budget_per_employee, 2) as budget_per_employee,
    ROUND(dm.avg_salary / NULLIF(dm.budget_per_employee, 0), 2) as
salary_to_budget_ratio
FROM dept_metrics dm
WHERE NOT EXISTS (
    SELECT 1
    FROM employee_project_counts epc
    WHERE epc.department_id = dm.dept_id
    AND epc.project_count < 2
)
AND dm.avg_salary > dm.budget_per_employee
ORDER BY salary_to_budget_ratio DESC;
```

**Skills:** CTEs, Complex calculations, Multiple JOINs, Subqueries, NULL handling, Advanced aggregation

Each task builds upon the previous ones, incorporating more complex concepts and combining multiple SQL features. The final task brings together almost all the concepts in a real-world analytical scenario.