

EXERCISE-9

Sub queries

Objectives

After completing this lesson, you should be able to do the following:

- Define subqueries
- Describe the types of problems that subqueries can solve
- List the types of subqueries
- Write single-row and multiple-row subqueries

Using a Subquery to Solve a Problem

Who has a salary greater than Abel's?

Main query:

Which employees have salaries greater than Abel's salary?

Subquery:

What is Abel's salary?

Subquery Syntax

`SELECT select_list FROM table WHERE expr operator (SELECT select_list FROM table);`

- The subquery (inner query) executes once before the main query (outer query).
- The result of the subquery is used by the main query.

A subquery is a SELECT statement that is embedded in a clause of another SELECT statement. You can build powerful statements out of simple ones by using subqueries. They can be very useful when you need to select rows from a table with a condition that depends on the data in the table itself.

You can place the subquery in a number of SQL clauses, including the following:

- WHERE clause
- HAVING clause
- FROM clause

In the syntax:

operator includes a comparison condition such as >, =, or IN

Note: Comparison conditions fall into two classes: single-row operators (>, =, >=, <, <=, <>) and multiple-row operators (IN, ANY, ALL). The subquery generally executes first, and its output is used to complete the query condition for the main (or outer) query

Using a Subquery

`SELECT last_name FROM employees WHERE salary > (SELECT salary FROM employees WHERE last_name = 'Abel');`

Displays employees who are not IT programmers and whose salary is less than that of any IT programmer. The maximum salary that a programmer earns is \$9,000.

< ANY means less than the maximum. > ANY means more than the minimum. = ANY is equivalent to IN.

Using the ALL Operator in Multiple-Row Subqueries

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary < ALL (SELECT salary FROM employees WHERE job_id = 'IT_PROG')
AND job_id <> 'IT_PROG';
```

Displays employees whose salary is less than the salary of all employees with a job ID of IT_PROG and whose job is not IT_PROG.

> ALL means more than the maximum, and < ALL means less than the minimum.

The NOT operator can be used with IN, ANY, and ALL operators.

Null Values in a Subquery

```
SELECT emp.last_name FROM employees emp
WHERE emp.employee_id NOT IN (SELECT mgr.manager_id FROM employees mgr);
```

Notice that the null value as part of the results set of a subquery is not a problem if you use the IN operator. The IN operator is equivalent to = ANY. For example, to display the employees who have subordinates, use the following SQL statement:

```
SELECT emp.last_name
FROM employees emp
WHERE emp.employee_id IN (SELECT mgr.manager_id FROM employees mgr);
```

Display all employees who do not have any subordinates:

```
SELECT last_name FROM employees
WHERE employee_id NOT IN (SELECT manager_id FROM employees WHERE manager_id IS
NOT NULL);
```

Find the Solution for the following:

1. The HR department needs a query that prompts the user for an employee last name. The query then displays the last name and hire date of any employee in the same department as the employee whose name they supply (excluding that employee). For example, if the user enters Zlotkey, find all employees who work with Zlotkey (excluding Zlotkey).

```
SELECT last_name, hire_date FROM employees WHERE department_id =
(SELECT department_id FROM employees WHERE last_name = '&name')
AND last_name <> '&name';
```

2. Create a report that displays the employee number, last name, and salary of all employees who earn more than the average salary. Sort the results in order of ascending salary.

```
SELECT employee_id, last_name, salary FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees)
ORDER BY salary;
```


3. Write a query that displays the employee number and last name of all employees who work in a department with any employee whose last name contains a *u*.

```
SELECT employee-id, last-name FROM employees WHERE  
department-id IN (SELECT department-id FROM employees WHERE  
last-name LIKE '%u%');
```

4. The HR department needs a report that displays the last name, department number, and job ID of all employees whose department location ID is 1700.

```
SELECT last-name, department-id, job-id FROM employees  
WHERE department-id IN (SELECT department-id FROM  
departments WHERE location-id = 1700);
```

5. Create a report for HR that displays the last name and salary of every employee who reports to King.

```
SELECT last-name, salary FROM employees WHERE  
manager-id IN (SELECT employee-id FROM employees  
WHERE manager-id IS NULL);
```

6. Create a report for HR that displays the department number, last name, and job ID for every employee in the Executive department.

```
SELECT department-id, last-name, job-id FROM employees  
WHERE department-id = (SELECT department-id FROM departments  
WHERE department-name = 'Executive');
```

7. Modify the query 3 to display the employee number, last name, and salary of all employees who earn more than the average salary and who work in a department with any employee whose last name contains a *u*.

```
SELECT employee-id, last-name, salary FROM employees  
WHERE salary > AVG(salary) AND department-id IN  
(SELECT department-id FROM employees WHERE  
last-name LIKE '%u%');
```

C. P. 57 k
2/9/25