6

Subqueries

Objectives

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

- Describe the types of problem that subqueries can solve
- Define subqueries
- 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.
- The result of the subquery is used by the main query (outer query).

Using a Subquery

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

11000

LAST_NAME
King
Kochhar
De Haan
Hartstein
Higgins

Guidelines for Using Subqueries

- Enclose subqueries in parentheses.
- Place subqueries on the right side of the comparison condition.
- The `ORDER BY` clause in the subquery is not needed unless you are performing Top-N analysis.
- Use single-row operators with single-row subqueries and use multiple-row operators with multiple-row subqueries.

Types of Subqueries

- Single-row subquery

- Multiple-row subquery

Single-Row Subqueries

- Return only one row
- Use single-row comparison operators

Operator	Meaning

Example

Display the employees whose job ID is the same as that of employee 141.

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
                (SELECT job_id
                 FROM   employees
                 WHERE  employee_id = 141);
```

Executing Single-Row Subqueries

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id =
                (SELECT job_id
                 FROM   employees
                 WHERE  employee_id = 141)
AND    salary >
                (SELECT salary
                 FROM   employees
                 WHERE  employee_id = 143);
```

ST_CLERK

2600

LAST_NAME	JOB_ID	SALARY
Rajs	ST_CLERK	3500
Davies	ST_CLERK	3100

Using Group Functions in a Subquery

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary =
                (SELECT MIN(salary)
                 FROM   employees);
```

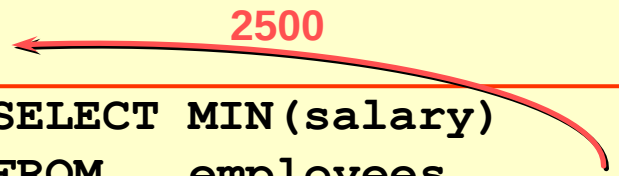
2500

LAST_NAME	JOB_ID	SALARY
Vargas	ST_CLERK	2500

The HAVING Clause with Subqueries

- The Oracle server executes subqueries first.
- The Oracle server returns results into the HAVING clause of the main query.

```
SELECT    department_id, MIN(salary)
FROM      employees
GROUP BY  department_id
HAVING    MIN(salary) >
          (SELECT MIN(salary)
           FROM   employees
           WHERE  department_id = 50);
```



What is Wrong with this Statement?

```
SELECT employee_id, last_name
FROM   employees
WHERE  salary =
                (SELECT   MIN(salary)
                 FROM     employees
                 GROUP BY department_id);
```

```
ERROR at line 4:
ORA-01427: single-row subquery returns more than
one row
```

Single-row operator with multiple-row subquery

Will this Statement Return Rows?

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
                (SELECT job_id
                 FROM   employees
                 WHERE  last_name = 'Haas');
```

```
no rows selected
```

Subquery returns no values

Multiple-Row Subqueries

```
SELECT last_name, salary, department_id
  FROM   employees
  WHERE  salary IN (SELECT   MIN(salary)
                    FROM     employees
                    GROUP BY department_id);
```

Example

Find the employees who earn the same salary as the minimum salary for each department.

The inner query is executed first, producing a query result. The main query block is then processed and uses the values returned by the inner query to complete its search condition. In fact, the main query would appear to the Oracle server as follows:

```
   SELECT last_name, salary, department_id
   FROM   employees
   WHERE  salary IN (2500, 4200, 4400, 6000, 7000,
8300, 8600, 17000);
```

Using the ANY Operator in Multiple-Row Subqueries

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

9000, 6000, 4200

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
124	Mourgos	ST_MAN	5800
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

...

10 rows selected.

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';
```

9000, 6000, 4200

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

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);

no rows selected
```

Summary

In this lesson, you should have learned how to:

- Identify when a subquery can help solve a question
- Write subqueries when a query is based on unknown values

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

Practice 6 Overview

This practice covers the following topics:

- Creating subqueries to query values based on unknown criteria
- Using subqueries to find out which values exist in one set of data and not in another