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

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

<table>
<thead>
<tr>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>King</td>
</tr>
<tr>
<td>Kochhar</td>
</tr>
<tr>
<td>De Haan</td>
</tr>
<tr>
<td>Hartstein</td>
</tr>
<tr>
<td>Higgins</td>
</tr>
</tbody>
</table>
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

  Main query
  \[\text{Subquery} \rightarrow \text{ST\_CLERK}\]

- Multiple-row subquery

  Main query
  \[\text{Subquery} \rightarrow \text{ST\_CLERK, SA\_MAN}\]
Single-Row Subqueries

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

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>
Executing Single-Row Subqueries

```sql
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);
```

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>JOB_ID</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rajs</td>
<td>ST_CLERK</td>
<td>3500</td>
</tr>
<tr>
<td>Davias</td>
<td>ST_CLERK</td>
<td>3100</td>
</tr>
</tbody>
</table>
Using Group Functions in a Subquery

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

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>JOB_ID</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vargas</td>
<td>ST_CLERK</td>
<td>2500</td>
</tr>
</tbody>
</table>
The **HAVING** Clause with Subqueries

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

```sql
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');
```

Subquery returns no values
Multiple-Row Subqueries

- Return more than one row
- Use multiple-row comparison operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>Equal to any member in the list</td>
</tr>
<tr>
<td>ANY</td>
<td>Compare value to each value returned by the subquery</td>
</tr>
<tr>
<td>ALL</td>
<td>Compare value to every value returned by the subquery</td>
</tr>
</tbody>
</table>
Using the **ANY** Operator in Multiple-Row Subqueries

```sql
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';
```

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>JOB_ID</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>124</td>
<td>Murgos</td>
<td>ST_MAN</td>
<td>6800</td>
</tr>
<tr>
<td>141</td>
<td>Raj</td>
<td>ST_CLERK</td>
<td>3500</td>
</tr>
<tr>
<td>142</td>
<td>Davies</td>
<td>ST_CLERK</td>
<td>3100</td>
</tr>
<tr>
<td>143</td>
<td>Matos</td>
<td>ST_CLERK</td>
<td>2600</td>
</tr>
<tr>
<td>144</td>
<td>Vargas</td>
<td>ST_CLERK</td>
<td>2500</td>
</tr>
</tbody>
</table>

10 rows selected.
Using the **ALL** Operator in Multiple-Row Subqueries

```sql
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';
```

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>JOB_ID</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>141</td>
<td>Raja</td>
<td>ST_CLERK</td>
<td>3500</td>
</tr>
<tr>
<td>142</td>
<td>Davies</td>
<td>ST_CLERK</td>
<td>3100</td>
</tr>
<tr>
<td>143</td>
<td>Matos</td>
<td>ST_CLERK</td>
<td>2600</td>
</tr>
<tr>
<td>144</td>
<td>Vargas</td>
<td>ST_CLERK</td>
<td>2500</td>
</tr>
</tbody>
</table>
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`
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