**DP Section 10.1 Fundamentals of Subqueries**

## Objectives

This lesson covers the following objectives:
•Define and explain the purpose of subqueries for retrieving data
•Construct and execute a single-row subquery in the WHERE clause
•Distinguish between single-row and multiple-row subqueries

## Purpose

•Has a friend asked you to go to a movie, but before you could answer "yes" or "no", you first had to check with your parents?
•Has someone asked you the answer to a math problem, but before you can give the answer, you had to do the problem yourself?
•Asking parents, or doing the math problem, are examples of subqueries.
•In SQL, subqueries enable us to find the information we need so that we can get the information we want.

## Subquery Overview

•Throughout this course, you have written queries to extract data from a database.
•What if you wanted to write a query, only to find out you didn't have all the information you needed to construct it?
•You can solve this problem by nesting queries—placing one query inside the other query.
•The inner query is called a "subquery."

Subquery Overview
•The subquery executes to find the information you don't know.
•The outer query uses that information to find out what you need to know.
•Being able to combine two queries into one 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.

Subquery Overview
•A subquery is a SELECT statement that is embedded in a clause of another SELECT statement.
•A subquery executes once before the main query.
•The result of the subquery is used by the main or outer query.
•Subqueries can be placed in a number of SQL clauses, including the WHERE clause, the HAVING clause, and the FROM clause.
•The subquery syntax is:

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

The SELECT statement in parentheses is the inner query or 'subquery'.
It executes first, before the outer query.

## Guidelines for Using Subqueries

•Guidelines:
–The subquery is enclosed in parentheses.
–The subquery is placed on the right side of the comparison condition.
–The outer and inner queries can get data from different tables.
–Only one ORDER BY clause can be used for a SELECT statement; if used, it must be the last clause in the outer query.
–A subquery cannot have its own ORDER BY clause.
–The only limit on the number of subqueries is the buffer size the query uses.

## Two Types of Subqueries

•The two types of subqueries are:
–Single-row subqueries that use single-row operators (>, =, >=, <, <>, <=) and return only one row from the inner query.

–Multiple-row subqueries that use multiple-row operators (IN, ANY, ALL) and return more than one row from the inner query.

**Subquery Example**
•What if you wanted to find out the names of the employees that were hired after Peter Vargas?
•The first thing you need to know is the answer to the question, "When was Peter Vargas hired?"
•Once you know his hire date, then you can select those employees whose hire dates are after his.

```
SELECT first_name, last_name,
hire_date
FROM employees
WHERE hire_date>
        (SELECT hire_date
        FROM employees
        WHERE last_name= 'Vargas');
```

**Subquery and Null**
•If a subquery returns a null value or no rows, the outer query takes the results of the subquery (null) and uses this result in its WHERE clause.
•The outer query will then return no rows, because comparing any value with a null always yields a null.

```
SELECT last_name
FROM employees
WHERE department_id=
        (SELECT department_id
        FROM employees
        WHERE last_name= 'Grant');
```

| FIRST_NAME | LAST_NAME | HIRE_DATE |
|---|---|---|
| Eleni | Zlotkey | 29-Jan-2000 |
| Kimberely | Grant | 24-May-1999 |
| Kevin | Mourgos | 16-Nov-1999 |
| Diana | Lorentz | 07-Feb-1999 |

**Subquery and Null**
•Who works in the same department as Grant?
•Grant's department_idis null, so the subquery returns NULL.
•The outer query then substitutes this value in the WHERE clause (WHERE department_id= NULL).
•The outer query returns no rows, because comparing anything with a null returns a null.

```
SELECT last_name
FROM employees
WHERE department_id=
        (SELECT department_id
        FROM employees
        WHERE last_name= 'Grant');
no data found
```

**Terminology**
Key terms used in this lesson included:
•Subquery
•Inner query
•Outer query
•Single-row subquery
•Multiple-row subquery
**Summary**
In this lesson, you should have learned how to:

•Define and explain the purpose of subqueries for retrieving data
•Construct and execute a single-row subquery in the WHERE clause
•Distinguish between single-row and multiple-row subqueries

## DP Section 10.2 / **Single-Row Subqueries**

Objectives
This lesson covers the following objectives:
•Construct and execute a single-row subquery in the WHERE clause or HAVING clause
•Construct and execute a SELECT statement using more than one subquery
•Construct and execute a SELECT statement using a group function in the subquery

Purpose
•As you have probably realized, subqueries are a lot like Internet search engines.
•They are great at locating the information needed to accomplish another task.
•In this lesson, you will learn how to create even more complicated tasks for subqueries to do for you.
•Keep in mind that subqueries save time in that you can accomplish two tasks in one statement.

**Facts About Single-row Subqueries**
•They:
–Return only one row
–Use single-row comparison operators (=, >,>=, <, <=, <>)
•Always:
–Enclose the subquery in parentheses.
–Place the subquery on the right hand side of the comparison condition.

**Additional Subquery Facts**
•The outer and inner queries can get data from different tables.
•Only one ORDER BY clause can be used for a SELECT statement, and if specified, it must be the last clause in the main SELECT statement.
•The only limit on the number of subqueries is the buffer size that the query uses.

**Subqueries from Different Tables**
•The outer and inner queries can get data from different tables.
•Who works in the Marketing department?

```
SELECT last_name, job_id, department_id
FROM employees
WHERE department_id=
        (SELECT department_id FROM departments WHERE department_name= 'Marketing')
ORDER BY job_id;
```

| LAST_NAME | JOB_ID | DEPARTMENT_ID |
|---|---|---|
| Hartstein | MK_MAN | 20 |
| Fay | MK_REP | 20 |

**Result of subquery**

| DEPARTMENT_ID |
|---|
| 20 |

Subqueries from Different Tables
•More than one subquery can return information to the outer query.

```
SELECT last_name, job_id, salary, department_id
FROM employees
WHERE job_id=
        (SELECT job_id FROM employees WHERE employee_id= 141)
AND department_id=
        (SELECT department_id FROM departments WHERE location_id= 1500);
```

Result of 1stsubquery

| JOB_ID |
| --- |
| ST_CLERK |

Result of 2ndsubquery

| DEPARTMENT_ID |
| --- |
| 50 |

| LAST_NAME | JOB_ID | SALARY | DEPARTMENT_ID |
| --- | --- | --- | --- |
| Rajs | ST_CLERK | 3500 | 50 |
| Davies | ST_CLERK | 3100 | 50 |
| Matos | ST_CLERK | 2600 | 50 |
| Vargas | ST_CLERK | 2500 | 50 |

**Group Functions in Subqueries**

•Group functions can be used in subqueries.
•A group function without a GROUP BY clause in the subquery returns a single row.
•The query on the next slide answers the question, "Which employees earn less than the average salary?"

Group Functions in Subqueries

•The subquery first finds the average salary for all employees, the outer query then returns employees with a salary of less than the average.

```
SELECT last_name, salary FROM employees
WHERE salary < (SELECT AVG(salary) FROM employees);
```

**Result of subquery**

| AVG(SALARY) |
| --- |
| 8775 |

| LAST_NAME | SALARY |
| --- | --- |
| Whalen | 4400 |
| Gietz | 8300 |
| Taylor | 8600 |
| Grant | 7000 |
| Mourgos | 5800 |
| Rajs | 3500 |
| Davies | 3100 |
| Matos | 2600 |
| Vargas | 2500 |
| Ernst | 6000 |
| Lorentz | 4200 |
| Fay | 6000 |

**Subqueries in the HAVING Clause**

•Subqueries can also be placed in the HAVING clause.
•Remember that the HAVING clause is similar to the WHERE clause, except that the HAVING clause is used to restrict groups and always includes a group function such as MIN, MAX, or AVG.
•Because the HAVING clause always includes a group function, the subquery will nearly always include a group function as well.

**Subquery Example**

•Which departments have a lowest salary that is greater than the lowest salary in department 50?
•In this example, the subquery selects and returns the lowest salary in department 50.

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

**Result of subquery**

| MIN(SALARY) |
|---|
| 2500 |

| DEPARTMENT_ID | MIN(SALARY) |
|---|---|
| - | 7000 |
| 90 | 17000 |
| 20 | 6000 |
| 110 | 8300 |
| 80 | 8600 |
| 10 | 4400 |
| 60 | 4200 |

Subquery Example

•The outer query uses this value to select the department ID and lowest salaries of all the departments whose lowest salary is greater than that number.

•The HAVING clause eliminated those departments whose MIN salary was less than department 50's MIN salary.

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

**Result of subquery**

| MIN(SALARY) |
|---|
| 2500 |

| DEPARTMENT_ID | MIN(SALARY) |
|---|---|
| - | 7000 |
| 90 | 17000 |
| 20 | 6000 |
| 110 | 8300 |
| 80 | 8600 |
| 10 | 4400 |
| 60 | 4200 |

Summary

In this lesson, you should have learned how to:

•Construct and execute a single-row subquery in the WHERE clause or HAVING clause

•Construct and execute a SELECT statement using more than one subquery

•Construct and execute a SELECT statement using a group function in the subquery

## DP Section 10.3 Multiple-Row Subqueries

Objectives

This lesson covers the following objectives:

• Correctly use the comparison operators IN, ANY, and ALL in multiple-row subqueries

• Construct and execute a multiple-row subquery in the WHERE clause or HAVING clause

• Describe what happens if a multiple-row subquery returns a null value

• Understand when multiple-row subqueries should be used, and when it is safe to use a single-row subquery

• Distinguish between pair-wise and non-pair-wise subqueries

Purpose

• A subquery is designed to find information you don't know so that you can find information you want to know.

• However, single-row subqueries can return only one row. What if you need to find information based on several rows and several values?
• The subquery will need to return several rows.
• We achieve this using multiple-row subqueries and the three comparison operators: IN, ANY, and ALL.

**Query Comparison**
• Whose salary is equal to the salary of an employee in department 20 ?
• This example returns an error because more than one employee exists in department 20, the subquery returns multiple rows.
• We call this a multiple-row subquery.

```
SELECT first_name, last_name
FROM employees
WHERE salary =
   (SELECT salary FROM employees WHERE department_id= 20);
```

ORA-01427: single-row subqueryreturns more than one row

| LAST_NAME | DEPT_ID | SALARY |
|---|---|---|
| Hartstein | 20 | 13000 |
| Fay | 20 | 6000 |

Query Comparison
• The problem is the equal sign (=) in the WHERE clause of the outer query.
• How can one value be equal to (or not equal to) more than one value at a time?
• It's a silly question, isn't it?

```
SELECT first_name, last_name
FROM employees
WHERE salary =
      (SELECT salary FROM employees WHERE department_id= 20);
```

ORA-01427: single-row subqueryreturns more than one row

IN, ANY, and ALL
• Subqueries that return more than one value are called multiple-row subqueries.
• Because we cannot use the single-row comparison operators (=, <, and so on), we need different comparison operators for multiple-row subqueries.
• The multiple-row operators are:
– IN,
– ANY
– ALL
• The NOT operator can be used with any of these three operators.

IN
• The IN operator is used within the outer query WHERE clause to select only those rows which are IN the list of values returned from the inner query.
• For example, we are interested in all the employees that were hired the same year as an employee in department 90.

```
SELECT last_name, hire_date
FROM employees
WHERE EXTRACT(YEAR FROM hire_date) IN
     (SELECT EXTRACT(YEAR FROM hire_date) FROM employees WHERE department_id=90);
```

LAST_NAME HIRE_DATE

| | |
|---|---|
| King | 17-Jun-1987 |
| Kochhar | 21-Sep-1989 |
| De Haan | 13-Jan-1993 |
| Whalen | 17-Sep-1987 |

4 rows returned

OR:

```
SELECT last_name, hire_date, TO_CHAR(hire_date, 'YYYY')
FROM employees
WHERE TO_CHAR(hire_date, 'YYYY') IN
        (SELECT TO_CHAR(hire_date, 'YYYY') FROM employees WHERE department_id=90);
```

| LAST_NAME | HIRE_DATE | YEAR |
|---|---|---|
| King | 17-Jun-1987 | 1987 |
| Kochhar | 21-Sep-1989 | 1989 |
| De Haan | 13-Jan-1993 | 1993 |
| Whalen | 17-Sep-1987 | 1987 |

4 rows returned

IN

- The inner query will return a list of the years that employees in department 90 were hired.
- The outer query will then return any employee that was hired the same year as any year in the inner query list.

```
SELECT last_name, hire_date
FROM employees
WHERE EXTRACT(YEAR FROM hire_date) IN
    (SELECT EXTRACT(YEAR FROM hire_date) FROM employees WHERE department_id=90);
```

| LAST_NAME | HIRE_DATE |
|---|---|
| King | 17-Jun-1987 |
| Kochhar | 21-Sep-1989 |
| De Haan | 13-Jan-1993 |
| Whalen | 17-Sep-1987 |

## ANY

The ANY operator is used when we want the outer-query WHERE clause to select the rows which match the criteria (<, >, =, etc.) of at leastone value in the subquery result set.

- The example shown will return any employee whose year hired is less than at least one year hired of employees in department 90.

```
SELECT last_name, hire_date
FROM employees
WHERE EXTRACT(YEAR FROM hire_date) < ANY
    (SELECT EXTRACT(YEAR FROM hire_date) FROM employees WHERE department_id=90);
```

| Year Hired |
|---|
| 1987 |
| 1989 |
| 1993 |

| LAST_NAME | HIRE_DATE |
|---|---|
| King | 17-Jun-1987 |
| Kochhar | 21-Sep-1989 |
| Whalen | 17-Sep-1987 |
| Hunold | 03-Jan-1990 |

Ernst 21-May-1991
5 rows returned

OR:
```
SELECT last_name, hire_date, TO_CHAR(hire_date, 'YYYY') YEAR
FROM employees
WHERE TO_CHAR(hire_date, 'YYYY') < ANY
        (SELECT TO_CHAR(hire_date, 'YYYY') FROM employees WHERE department_id=90);
```

| LAST_NAME | HIRE_DATE | YEAR |
|---|---|---|
| King | 17-Jun-1987 | 1987 |
| Kochhar | 21-Sep-1989 | 1989 |
| Whalen | 17-Sep-1987 | 1987 |
| Hunold | 03-Jan-1990 | 1990 |
| Ernst | 21-May-1991 | 1991 |

5 rows returned

## ALL

```
SELECT last_name, hire_date
FROM employees
WHERE EXTRACT(YEAR FROM hire_date) < ALL
(SELECT EXTRACT(YEAR FROM hire_date) FROM employees WHERE department_id=90);
```
no data found

```
SELECT last_name, hire_date, TO_CHAR(hire_date, 'YYYY') YEAR
FROM employees
WHERE TO_CHAR(hire_date, 'YYYY') < ALL
        (SELECT TO_CHAR(hire_date, 'YYYY') FROM employees WHERE department_id=110);
```

| LAST_NAME | HIRE_DATE | YEAR |
|---|---|---|
| King | 17-Jun-1987 | 1987 |
| Kochhar | 21-Sep-1989 | 1989 |
| De Haan | 13-Jan-1993 | 1993 |
| Whalen | 17-Sep-1987 | 1987 |
| Hunold | 03-Jan-1990 | 1990 |
| Ernst | 21-May-1991 | 1991 |

6 rows returned

## NULL Values

```
SELECT last_name, manager_id FROM employees
where manager_id is null
```

| LAST_NAME | MANAGER_ID |
|---|---|
| King | - |

1 rows returned
(because of NULL manager_id):

```
SELECT last_name, employee_id FROM employees
WHERE employee_id <= ALL
        (SELECT manager_id FROM employees);
```
no data found

```
SELECT last_name, employee_id FROM employees
WHERE employee_id <= ALL
        (SELECT manager_id FROM employees);
```

| LAST_NAME | EMPLOYEE_ID |
|---|---|

| | |
|---|---|
| King | 100 |

1 rows returned

## GROUP BY and HAVING

```
SELECT department_id, MIN(salary) FROM employees
GROUP BY department_id
```

| DEPARTMENT_ID | MIN(SALARY) |
|---|---|
| - | 7000 |
| 90 | 17000 |
| 20 | 6000 |
| 110 | 8300 |
| 80 | 8600 |
| 50 | 2500 |
| 10 | 4400 |
| 60 | 4200 |

8 rows returned

```
SELECT department_id, salary FROM employees
WHERE department_id IN (10,20)
ORDER BY department_id
```

| DEPARTMENT_ID | SALARY |
|---|---|
| 10 | 4400 |
| 20 | 13000 |
| 20 | 6000 |

3 rows returned

```
SELECT department_id, MIN(salary) FROM employees
GROUP BY department_id
HAVING MIN(salary) < ANY
        (SELECT salary FROM employees WHERE department_id IN (10,20))
ORDER BY department_id;
```

| DEPARTMENT_ID | MIN(SALARY) |
|---|---|
| 10 | 4400 |
| 20 | 6000 |
| 50 | 2500 |
| 60 | 4200 |
| 80 | 8600 |
| 110 | 8300 |
| - | 7000 |

7 rows returned (17000 does not come because > 13000).

**Multiple-Column Subqueries**

```
SELECT employee_id, manager_id, department_id FROM employees
WHERE(manager_id,department_id) IN
        (SELECT manager_id,department_id FROM employees WHERE employee_id IN (149,174))
AND employee_id NOT IN (149,174)
```

| EMPLOYEE_ID | MANAGER_ID | DEPARTMENT_ID |
|---|---|---|
| 176 | 149 | 80 |

1 rows returned

```
SELECT employee_id, manager_id, department_id FROM employees
WHERE manager_id IN
```

```
        (SELECT manager_id FROM employees WHERE employee_id IN (149,174))
AND department_id IN
        (SELECT department_id FROM employees WHERE employee_id IN (149,174))
AND employee_id NOT IN(149,174);
```

| EMPLOYEE_ID | MANAGER_ID | DEPARTMENT_ID |
|---|---|---|
| 176 | 149 | 80 |

1 rows returned

**EXISTS & NOT EXISTS in Subqueries**

| | |
|---|---|
| SELECT last_name AS "Not a Manager"<br>FROM employees emp<br>WHERE employee_id NOT IN<br>(SELECT manager_id FROM employees<br>where manager_id is not null); | SELECT last_name AS "Not a Manager"<br>FROM employees emp<br>WHERE NOT EXISTS<br>(SELECT * FROM employees mgr<br>WHERE mgr.manager_id = emp.employee_id); |
| **Not a Manager**<br>Whalen<br>Gietz<br>Abel<br>Taylor<br>Grant<br>Rajs<br>Davies<br>Matos<br>Vargas<br>Ernst<br>Lorentz<br>Fay<br>12 rows returned | **Not a Manager**<br>Whalen<br>Gietz<br>Abel<br>Taylor<br>Grant<br>Rajs<br>Davies<br>Matos<br>Vargas<br>Ernst<br>Lorentz<br>Fay<br>12 rows returned |

```
SELECT last_name AS "Not a Manager" FROM employees emp
WHERE emp.employee_id NOT IN
        (SELECT mgr.manager_id FROM employees mgr);
```

no data found

| | |
|---|---|
| SELECT last_name AS "Managers"<br>FROM employees emp<br>WHERE EXISTS<br>(SELECT * FROM employees mgr<br>WHERE mgr.manager_id = emp.employee_id); | SELECT last_name AS "Managers"<br>FROM employees<br>WHERE employee_id IN<br>(SELECT manager_id FROM employees<br>WHERE manager_id is not null); |
| **Managers**<br>King<br>Kochhar<br>De Haan<br>Higgins<br>Zlotkey<br>Mourgos<br>Hunold<br>Hartstein<br>8 rows returned | **Managers**<br>King<br>Kochhar<br>De Haan<br>Higgins<br>Zlotkey<br>Mourgos<br>Hunold<br>Hartstein<br>8 rows returned |

| | |
|---|---|
| SELECT SUBSTR(first_name, 1, 1) \|\| ' ' \|\| last_name AS "Employee", salary AS "Salary", department_name AS "Department Name" FROM employees e JOIN departments d ON e.department_id = d.department_id WHERE salary > (SELECT AVG(salary) FROM employees sqe WHERE sqe.department_id = e.department_id); | SELECT o.first_name, i.department_name o.last_name, o.salary FROM employees o WHERE o.salary > (SELECT AVG(i.salary) FROM employees i WHERE i.department_id = o.department_id); |