

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');`

The inner query determines the salary of employee Abel. The outer query takes the result of the inner query and uses this result to display all the employees who earn more than this amount.

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.

Type of Subqueries

- Single-row subqueries: Queries that return only one row from the inner SELECT statement.
- Multiple-row subqueries: Queries that return more than one row from the inner SELECT statement.

Single-Row Subqueries

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

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);
```

Displays employees whose job ID is the same as that of employee 141 and whose salary is greater than that of employee 143.

```
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);
```

Using Group Functions in a Subquery

Displays the employee last name, job ID, and salary of all employees whose salary is equal to the minimum salary. The MIN group function returns a single value (2500) to the outer query.

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

The HAVING Clause with Subqueries

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

Displays all the departments that have a minimum salary greater than that of 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);
```

Example

Find the job with the lowest average salary.

```
SELECT job_id, AVG(salary)
FROM employees
GROUP BY job_id
HAVING AVG(salary) = (SELECT MIN(AVG(salary))
FROM employees
GROUP BY job_id);
```

What Is Wrong in this Statements?

```
SELECT employee_id, last_name
FROM employees
WHERE salary = (SELECT MIN(salary) FROM employees GROUP BY department_id);
Will This Statement Return Rows?
SELECT last_name, job_id
FROM employees
WHERE job_id = (SELECT job_id FROM employees WHERE last_name = 'Haas');
```

Multiple-Row Subqueries

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

Example

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

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

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';
```

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:

- 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 lname, hire_date from emp where dept_id =
(select dept_id from emp l_name = 'Zlotkey') and l_name <
'Zlotkey';

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 emp-id, l-name, salary from emp where salary > (Select avg(salary) from emp) order by salary asc;

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 emp-id, l-name from emp where dep-id in (Select dep-id from emp where l-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 l-name, dep-id, job-id from emp where job-id in (Select job-id from emp where department id in (Select dept-id from dept where loc-id = 1700));

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

Select l-name, salary from emp where manager-id = (Select emp-id from emp where l-name = 'King');

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

Select dept-id, l-name, job-id from emp where dept-id

= (Select dept-id from dept where dept-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 emp-id, l-name, Sal from emp where Sal >

Select avg (Sal) from emp) and dept-id in (Select

distinct dept-id from emp where l-name like '%u%');

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	5
Total (15)	15
Faculty Signature	C BPL

Practice Questions

1. Ellen Abel is an employee who has received a \$2,000 raise. Display her first name and last name, her current salary, and her new salary. Display both salaries with a \$ and two decimal places. Label her new salary column AS New_Salary.

Select first_name, l_name, Concat('\$', format(Salary, 2)) AS Current_Salary, Concat('\$', format(Salary + 2000, 2)) AS 'New_Salary'
From emp where first_name = 'Ellen' and l_name = 'Abel';

2. On what day of the week and date did Global Fast Foods promotional code 110 Valentine's Special begin?

Select promo_code, Start_date Day Name(Start_date) AS day_of_week
from promotion where promo_code = 110;

3. Create one query that will convert 27-Oct-2004 into each of the following (you will have to convert 27-Oct-2004 to a date and then to character data):

December 25th, 2004
DECEMBER 25TH 2004
25th December, 2004

Select To_Char(to_date('27-OCT-2004', 'DD-MON-YYYY'), 'Current Date')
AS "Format 1"; To_Char(to_date('27-OCT-2004', 'DD-MON-YYYY')) AS
'MONTH DD TH, YYYY') AS "Format 2";

Select To_Char(to_date('27-OCT-2004', 'DD-MON-YYYY')) AS "Format"
Format dual;

4. Create a query that will format the Dis on Demand d_packages columns, low range and high range package costs, in the format \$2500.00.

Select To_Char(low_range, '\$99,999.00') AS "Low Range Cost",
To_Char(high_range, '\$99,999.00') AS "High Range Cost" From
d_packages;

5. Convert JUNE 19 2004 to a date using the to_date model.

Select To_DATE('JUNE 19 2004', 'Month DD YYYY') AS "Convert Date"
From dual;

6. What is the distinction between implicit and explicit datatype conversion? Give an example of each.

Select * FROM employees where empbyt-id = '10';

Select * FROM employees where hire-date > TO_DATE ('01-JAN-2023',
'DD-MON-XXXX')

7. Why is it important from a business perspective to have datatype conversions?

~~Batch Integration:~~ Business collects data from many sources. This data rarely arrives in the correct format. Conversions are necessary to extract, transform and load this data.

For example a web API might send sales figure as a string which must be explicitly converted to a number before it can be stored and used in a database.

Evaluation Procedure	Marks awarded
Practice Evaluation (5)	5
Viva(5)	5
Total (10)	10
Faculty Signature	