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). statement. 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.

Types 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 employeesWHERE 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:

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 e.last_name, e.hire_date FROM employees e
WHERE e.department_id = ( SELECT department_id FROM
employees WHERE last_name = '&last_name').
AND e.last_name <> '&last_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 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 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 = (SELECT employee_id FROM
employees WHERE last_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 department_id, last_name, job_id FROM
employees WHERE department_id = ( SELECT
department_id = ( SELECT department_id FROM
department 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 >
(SELECT AVG(salary)
FROM employees)
AND department_id IN
(SELECT department_id
 FROM employees
 WHERE last_name
 LIKE '%u%');
```

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

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, last_name, TO_CHAR(salary,
'$999,999.00') AS current_salary, TO_CHAR(salary+2000,
'$999,999.00') AS New_Salary FROM employees WHERE
first_name = 'Ellen' AND last_name = 'Abel';
```

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

```
SELECT promo_id, promo_name, TO_CHAR(promo_begin_date,
'Day, DD-Mon-YYYY') AS Start_Date FROM promotions
WHERE promo_id = 110;
```

3. Create one query that will convert 25-Dec-2004 into each of the following (you will have to convert 25-Dec-2004 to a date and then to character data):

```
SELECT TO_CHAR(TO_DATE('25-Dec-2004', 'DD-Mon-YYYY'), 'MM/DD/YYYY')
AS Format1, TO_CHAR(TO_DATE('25-Dec-2004', 'DD-Mon-YYYY'),
'Month DD, YYYY') AS Format2, TO_CHAR(TO_DATE('25-Dec-2004',
'DD-Mon-YYYY'), 'DY, DDTH Month YYYY') AS Format3 FROM dual;
```

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

```
SELECT package_id, TO_CHAR(low_range, '$99999.99') AS
Low_Range, TO_CHAR(high_range, '$9999.99') AS High_Range
FROM d_packages;
```

5. Convert JUNE192004 to a date using the fx format model.

```
SELECT TO_DATE('JUNE 192004', 'FXMonthDDYYYY') AS
converted_date FROM dual;
```

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

Implicit conversion:
Oracle automatically converts data types when needed.
SELECT 'The salary is ' || salary FROM employees;

Explicit conversion: You tell oracle exactly how to convert
SELECT TO_CHAR(salary, '$999,999.00') FROM employees;

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

- Ensures data consistency across systems e.g., currency.
- Prevent errors during calculations or comparisons.
- Enables reporting & formatting for business users.
- Support integration with other apps.

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