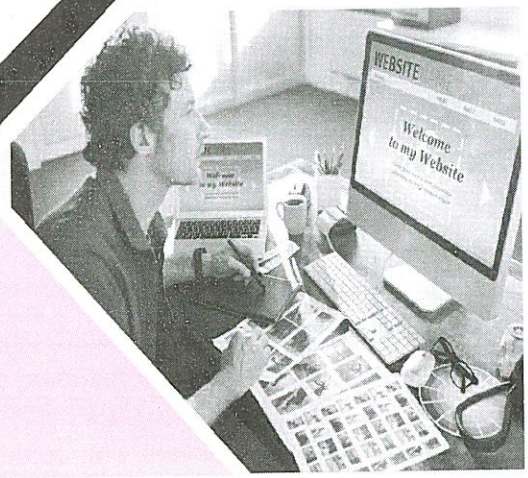
CHAPTER 27



SUB QUERIES

CHAPTER OBJECTIVES

In this chapter you will learn:

- Need of sub queries
- Single-row sub queries
- Multiple-row sub queries
- Multiple-column sub queries
- Correlated Sub-queries
- Usage of special operators in Sub-queries

27.1 USING A SUB-QUERY TO SOLVE A PROBLEM

A sub-query is a SELECT statement that is embedded in a clause of another SELECT statement. In order to understand the use of sub-query, consider a case to find a person who earns a salary greater than Jones salary. To solve this problem, you need two queries: one query to find what Jones earns and a second query to find who earns more than that amount.

We can solve this problem by combining the two queries, placing one query inside the other query. The inner query or the sub-query returns a value that is used by the outer query or the main query. Using a sub-query is equivalent to performing two sequential queries and using the result of the first query as the search value in the second query.

We can build powerful statements out of simple ones by using sub queries. They can be very useful when we need to select rows from a table with a condition that depends on the data in the table itself.

We can place the sub-query in a number of SQL clauses:

- WHERE clause

- HAVING clause
- FROM clause

Syntax:

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

The sub-query is often referred to as a nested SELECT, sub-SELECT, or inner SELECT statement. The sub-query generally executes first, and its output is used to complete the query condition for the main or outer query.

Consider the following query:

```
SQL> SELECT ename FROM emp
    WHERE sal> (SELECT sal FROM emp WHERE empno=7566);
```

The output is:

ENAME
KING
FORD
SCOTT

In the above query, the inner query determines the salary of employee 7566. 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.

27.2 TYPES OF SUBQUERIES

Following table shows the types of sub queries and their description:

Type	Description
Single-row sub queries	Queries that return only one row from the inner SELECT statement
Multiple-row sub queries	Queries that return more than one row from the inner SELECT statement
Multiple-column sub queries	Queries that return more than one column from the inner SELECT statement.

27.2.1 Single-Row Sub queries

A single-row sub-query is one that returns one row from the inner SELECT statement. This type of the sub-query uses a single-row operator.

Example

- Display the employees whose job title is the same as that of employee 7369.

```
SQL> SELECT ename, job FROM emp
WHERE job=(SELECT job FROM emp WHERE empno=7369);
```

The output is:

ENAME	Job
JAMES	CLERK
SMITH	CLERK
ADAMS	CLERK
MILLER	CLERK

Using Group functions in a sub-query

We can display data from a main query by using a group function in a sub-query to return a single row. The sub-query is in paranthesis and is placed after the comparison operator.

For example: In order to display the employee name whose salary is equal to the minimum salary. The MIN group function returns a single value to the outer query.

```
SQL>SELECT ename FROM emp WHERE sal = (SELECT MIN(sal) FROM emp);
```

- To display name of top five highest paid employees from emp table.

```
SQL>SELECT E.ENAME, E.SAL FROM EMP E WHERE 5>
(SELECT COUNT(*) FROM EMP WHERE E.SAL<EMP.SAL)
ORDER BY SAL DESC;
```

- To display name of five employees who are getting minimum pay from emp table

```
SQL>SELECT E.ENAME,E.SAL FROM EMP E WHERE 5>
(SELECT COUNT(*) FROM EMP WHERE E.SAL>EMP.SAL)
ORDER BY SAL DESC
```

HAVING Clause with Sub queries

We can use sub queries not only in the WHERE clause, but also in the Having clause. The Oracle Server executes the sub-query, and the results are returned into the Having clause of the main query.

For example:

To display all the departments that have a minimum salary greater than that of department 20.

```
SQL> SELECT deptno, MIN(sal) FROM emp GROUP BY deptno
     Having MIN(sal)>(Select MIN(sal) from emp where deptno=20);
```

The output is:

DEPTNO	MIN(SAL)
10	1300
30	950

27.2.2 Multiple-Row Sub-queries

Sub-queries that return more than one row are called multiple row sub queries. We use a multiple-row operator, instead of a single-row operator, with a multiple-row sub-query. The multiple-row operator expects one or more values.

There are following multi-row operators:

Operator	Meaning
IN	Equal to any member in the list
ANY	Compare value to each value returned by the sub-query
ALL	Compare value to every value returned by the sub-query

Example

- Find the employees who earn the same salary as the minimum salary for departments.

```
SQL> SELECT ename, sal, deptno FROM emp
WHERE sal IN (select MIN(sal) FROM emp Group By deptno);
```

The inner query is executed first, producing a query result containing three rows: 800,950, 1300. 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 look like the following to the Oracle Server.

```
SQL> SELECT ename, sal, deptno FROM emp WHERE sal IN(800, 950,1300);
```

Using ANY Operator in Multiple-Row Sub queries

The ANY operator (and its synonym SOME operator) compares a value to each value returned by a sub-query.

Example

- Display employees whose salary is less than any clerk and who are not clerks.

  The maximum salary that a clerk earns is ₹ 1300. The SQL statement displays all the employees who are not clerks but earn less than ₹ 1300.

Here **<ANY:** means less than the maximum

>ANY: means more than the minimum

=ANY: is equivalent to IN.

```
SQL > SELECT empno, ename, job
FROM emp
WHERE sal < ANY(SELECT sal FROM emp WHERE job='CLERK') AND
job<>'CLERK';
```

The output is:

EMPNO	ENAME	JOB
7654	MARTIN	SALESMAN
7521	WARD	SALESMAN

Using ALL Operator in Multiple-Row Sub queries

The ALL operator compares a value to every value returned by a sub-query.

Example

- Display employees whose salary is greater than the average salaries of all the departments.

The highest average salary of a department is ₹ 2916.66, so the query returns those employees whose salary is greater than ₹ 2916.66

```
SQL> SELECT empno, ename, job FROM emp
        WHERE sal>ALL (SELECT avg(sal) FROM emp GROUP BY deptno);
```

The output is:

EMPNO	ENAME	JOB
7839	KING	PRESIDENT
7566	JONES	MANAGER
7902	FORD	ANALYST
7788	SCOTT	ANALYST

Here >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.

27.2.3 Multiple-column Subqueries

So far we have written single-row sub queries where only one column was compared in the WHERE clause or HAVING clause of the SELECT statement. If you want to compare two or more columns, we must write a compound WHERE clause using logical operators. Multiple-column sub queries enable we to combine duplicate WHERE conditions into a single WHERE clause.

Syntax

```
SELECT column, column,.....
   FROM table
      WHERE (column, column, .....) IN
         (SELECT column, column, .....
            FROM table
               WHERE condition);
```

Example

- To display the order id, product id and quantity of items in the item table that match both the product id and quantity of an item in order 605.

```
SQL> SELECT ordid, prodid,qty FROM item
   WHERE (prodid, qty) IN
      (SELECT prodid, qty FROM item
         WHERE ordid=605) AND ordid<>605;
```

When the above SQL statement is executed, the Oracle server compares the values in both the PRODID and QTY columns and returns those orders where the product number and quantity for that product match both the product number and quantity for an item in order 605.

The output of the SQL statement is:

ORDID	PRODID	QTY
617	100861	100
617	100870	500
616	102130	10

The output shows that there are three items in other orders that contain the same product number and quantity as an item in order 605. Oracle first executes the sub-query to see the PRODID and QTY values for each item in order 605.

27.3 CORRELATED SUB-QUERIES

Oracle performs a correlated sub-query when the sub-query references a column from a table referred to in the parent statement. A correlated sub-query is evaluated once for each row processed by the parent statement. The parent statement can be a SELECT, UPDATE, or DELETE statement.

Example:

- Select the highest-paid employee of each department.

The sub-query is executed for each row returned in the parent query. Notice that the parent table column is used with the alias name inside the sub-query.

```
SQL> SELECT deptno, ename, salary
    FROM emp e1
        WHERE salary = (SELECT MAX(salary) FROM emp
            WHERE deptno = e1.deptno) ORDER BY deptno;
```

The output is:

DEPTNO	ENAME	SALARY
10	A_EDWARD	5000
20	SCOTT	3000
20	FORD	3000
30	K_BLAKE	2850

27.4 USING SPECIAL OPERATORS IN SUB-QUERIES

Some special operators used in sub queries are:

- EXISTS
- SOME
- ANY
- ALL

27.4.1 EXISTS Operator

The EXISTS operator is frequently used with correlated sub queries. It tests whether a value is there (NOT EXISTS ensure for nonexistence of values). If the value exists it

returns TRUE, if it does not exists it returns FALSE. NOT EXISTS operator is more reliable if the sub-query returns any NULL values.

Examples:

- List all the employees who have at least one person reporting to them.

```
SQL>SELECT empno, ename, job from emp WHERE exists (SELECT empno
from emp
    WHERE emp.mgr=e.empno) ORDER BY empno;
```

- List the employee details if and only if more than 10 employees are present in department number is:

```
SQL>SELECT * FROM emp WHERE DEPTNO=10 AND
    EXISTS (SELECT COUNT(*) FROM emp WHERE deptno=10
    GROUP BY DEPTNO HAVING COUNT(*) >10);
```

- List all the employees details who do not manage any one.

```
SQL>SELECT ename, job FROM emp e
    WHERE NOT EXISTS (SELECT * FROM emp WHERE e.mgrno=e.empno);
```

27.4.2 ANY Operator

The ANY operator compares the lowest value from the set.

Examples:

- List the employee names whose salary is greater than the lowest salary of employee belonging to department number 20;

```
SQL>SELECT ename FROM emp
    WHERE sal>ANY (SELECT sal FROM emp WHERE deptno=20);
```

- List the employee details of those employees whose salary is greater than any of the managers:

```
SQL>SELECT empno, ename, sal FROM emp
    WHERE sal>ANY(SELECT sal FROM emp WHERE job='MANAGER');
```

27.4.3 ALL Operator

In case of ALL operator the predicate is true if every value selected by the sub-query satisfies the condition in the predicate of the outer query.

Examples

- List the employee names whose salary is greater than the highest salary of all employees belonging to department number 20;

```
SQL>SELECT ename FROM emp
    WHERE sal> ALL (SELECT sal FROM emp WHERE deptno=20);
```

The inner query returns salary of all employees who belong to department number 20. The outer query selects employee name of that employee whose salary is greater than the employees' salary who belong to department number 20.

- List the details of the employee earning more than the highest paid MANAGER;

```
SQL> SELECT empno, ename, sal FROM emp
     WHERE sal>ALL (SELECT sal FROM emp WHERE job='MANAGER');
```

FLASH BACK

A sub-query is a SELECT statement that is embedded in a clause of another SELECT statement. In order to understand the use of sub-query, consider a case to find a person who earns a salary greater than Jones salary. To solve this problem, you need two queries: one query to find what Jones earns and a second query to find who earns more than that amount.

Types of Sub-queries

Type	Description
Single-row sub queries	Queries that return only one row from the inner SELECT statement
Multiple-row sub queries	Queries that return more than one row from the inner SELECT statement
Multiple-column sub queries	Queries that return more than one column from the inner SELECT statement.

Oracle performs a correlated sub-query when the sub-query references a column from a table referred to in the parent statement. A correlated sub-query is evaluated once for each row processed by the parent statement Some special operators used in sub queries are: EXISTS, ANY, SOME and ALL.

HANDS ON SESSION

Lab Assignmnet-6

1. You need to find the salaries for all the employees who have a higher salary than the Vice President of a company 'ABC'.Which of the following queries will give you the required result? (Consider the table structure as given)

```
SQL> DESC employees
```

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)

COMMISSION_PCT	NUMBER(2,2)
MANAGER_ID	NUMBER(6)
DEPARTMENT_ID	NUMBER(4)

(a)
```
SELECT first_name, last_name, salary
FROM employees
WHERE salary > (SELECT salary
FROM employees
WHERE job_id = 'VICE-PRESIDENT');
```

(b)
```
SELECT first_name, last_name, salary
FROM employees
WHERE salary = (SELECT salary
FROM employees
WHERE job_id = 'VICE-PRESIDENT');
```

(c)
```
SELECT first_name, last_name, salary
FROM employees
WHERE job_id = 'VICE-PRESIDENT');
```

(d) None of the above

2. What will be the outcome of the following query? (Consider the given table structure)

SQL> DESC employees

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

```
SELECT first_name, last_name, salary
FROM employees
WHERE salary ANY (SELECT salary FROM employees);
```

(a) It executes successfully giving the desired results

(b) It executes successfully but does not give the desired results

(c) It throws an ORA error

(d) It executes successfully and gives two values for each row obtained in the result set

Examine the exhibit and answer the questions 3 to 6 that follow.

EMPLOYEES		
Name	Null?	Type
EMP_ID	NOT NULL	NUMBER(4)
FIRST_NAME		VARCHAR2(10)
LAST_NAME		VARCHAR2(10)
JOB		VARCHAR2(9)
MGR		VARCHAR2(4)
HIRE_DATE		DATE
SALARY	NOT NULL	NUMBER
COMM_PCT		NUMBER(7,2)
DEPT_ID		NUMBER(2)

DEPARTMENTS		
Name	Null?	Type
DEPT_ID		NUMBER
DEPT_NAME		VARCHAR2(20)
DEPT_LOC		VARCHAR2(20)

3. You need to find out the names of all employees who belong to the same department as the employee 'Jessica Butcher' who is in department 100 and has an employee ID 40. Which of the following queries will be correct?

(a)
```
SELECT first_name, last_name
FROM employees
WHERE last_name = 'Butcher'
And first_name = 'Jessica';
```

(b)
```
SELECT first_name, last_name
FROM employees
WHERE department =100;
```

(c)
```
SELECT first_name, last_name
FROM employees
WHERE department = (SELECT department
FROM employees
WHERE first_name = 'Jessica'
AND last_name = 'Butcher');
```

(d)
```
SELECT first_name, last_name
FROM employees
WHERE department = (SELECT department
FROM employees
WHERE first_name = 'Jessica'
AND last_name = 'Butcher'
AND department = 100
AND employee_id = 40);
```

4. You need to find out the employees which belong to the department of 'Jessica Butcher' and have salary greater than the salary of 'Jessica Butcher' who has an employee ID of 40. Which of the following queries will work?

(a)
```
SELECT first_name, last_name
FROM employees
WHERE last_name = 'Butcher'
```

```
AND first_name = 'Jessica'
AND salary > 10000;
```

(b)
```
SELECT first_name, last_name
FROM employees
WHERE department = 100;
```

(c)
```
SELECT first_name, last_name
FROM employees
WHERE department = (SELECT department
                    FROM employees
                    WHERE first_name = 'Jessica'
                    AND last_name = 'Butcher'
                    AND employee_id = 40)
AND salary > (SELECT salary
              FROM employees
              WHERE first_name = 'Jessica'
              AND last_name = 'Butcher'
              AND employee_id = 40);
```

(d)
```
SELECT first_name, last_name
FROM employees
WHERE department = (SELECT department
                    FROM employees
                    WHERE first_name = 'Jessica'
                    AND last_name = 'Butcher'
                    AND department = 100);
```

5. Based on the answers for questions 3rd and 4th, what type of sub-queries is used by them?

(a) Single row sub-query (b) Multiple row sub-query

(c) Both A and B (d) Inline sub-query

6. Consider two statements about outer and inner queries in context of SQL sub-queries?

(i) The inner queries can get data from only one table

(ii) The inner queries can get data from more than one table

Which of the above statements are true?

(a) (i) (b) (ii) (c) Both (i) and (ii) (d) Neither (i) nor (ii)

Examine the table structure as follows and answer the questions 7 to 12 that follow:

SQL> DESC employees

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)

EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

7. What will be the outcome of the following query? (Choose the most appropriate answer)

SQL> DESC employees

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

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

(a) It executes successfully and gives the employees who have salaries equal to the max salary.

(b) It executes successfully but does not give the required results

(c) It throws an error as a group function is used in the sub-query

(d) It throws an error as a single row sub-query should contain a multi-row operator

8. What will be the outcome of the query that follows?

```
SELECT first_name, last_name, min(salary)
FROM employees
GROUP BY department_id
HAVING MIN(salary) >
                    (SELECT min(salary)
                    FROM employees
                    WHERE department_id = 100);
```

(a) It executes successfully and gives the names and minimum salary greater than department 100 of all employees

(b) It executes successfully and gives the salaries of the employees in department 100

(c) It executes successfully and gives the names and minimum salaries of all the employees.

(d) It throws an error.

9. You need to find the job which has a maximum average salary.Which of the following queries will give you the required results?

(a)
```
SELECT job_id, avg(salary)
FROM employees
GROUP BY job_id;
```

(b)
```
SELECT job_id, avg(salary)
FROM employees
GROUP BY job_id
HAVING job_id in (SELECT max(avg(salary) FROM employees);
```

(c)
```
SELECT job_id, avg(salary)
FROM employees
GROUP BY job_id
HAVING max(avg(salary) in (SELECT max(avg(salary) FROM employees);
```

(d)
```
SELECT job_id, avg(salary)
FROM employees
GROUP BY job_id
HAVING avg(salary) in (SELECT max(avg(salary) FROM employees GROUP BY job_id);
```

10. The following query throws an error. Choose the correct reason for the error as given in the options.

```
SELECT first_name, last_name
FROM employees
WHERE commission_pct = (SELECT min(commission_pct)
    FROM employees
    GROUP BY department_id);
```

(a) The GROUP BY clause is not required in the sub-query

(b) A function cannot be used in a sub-query SELECT statement

(c) The single row sub-query gives multiple records

(d) The use of "=" operator is invalid; an IN operator will work correctly

11. Consider the query given below.How many records will be returned as a result of the above query? (Assuming the no employee with job id XX exists in the company)

```
SELECT first_name, last_name
FROM employees
WHERE salary = (SELECT salary
                FROM employees
                WHERE job_id = 'XX');
```

(a) 1 (b) NULL (c) 0

(d) The query raises ORA error because sub-query is invalid.

12. What happens if the WHERE condition in the query given in question 26 is replaced with a new one (WHERE job_id IS NOT NULL)? (Assume the number of records in 'employees' table is 14).

(a) 1 (b) 14 (c) 0 (d) ORA error

Solution Keys

1. (a) In the option 'A', the inner sub-query gives the VP's salary as a result to the outer query.

2. (c) Multi-row operators cannot be used in single-row sub-queries and vice versa.

3. (d) 'D' is more appropriate than 'C' because it filters on employee id which is unique and ensures that the sub-query will return single row only. 'C' can fail if there are more than one employee with the same first and last name.

4. (c) More than one sub-query can be written in one SQL statement to add more than one condition.

5. (a) The questions 3rd and 4th given above demonstrate the usage sub-queries in a SELECT statement.

6. (b) Sub-queries can fetch data from more than one table.

7. (a) A group function can be used within a sub-query.

8. (a) HAVING clause can be used in sub-queries as shown

9. (d) Sub-queries can make use of group functions and HAVING clause to restrict the groups.

10. (c, d) The GROUP BY clause gives the minimum commission_pct for each department and hence multiple results are fetched to the main query giving an error.

11. (c) Since there is no employee with job_id "XX" in the company, the sub-query returns no result, which when equated to job_id in the main query gives a 0.

12. (d) The query execution raises the exception "ORA-01427: single-row subquery returns more than one row".

○○○○○