SUBQUERIES TO SOLVE QUERIES

A subquery is best defined as a query within a query. Subqueries enable you to write queries that select data rows for criteria that are actually developed while the query is executing at run time. More formally, it is the use of a SELECT statement inside one of the clauses of another SELECT statement. n fact, a subquery can be contained inside another subquery, which is inside another subquery, and so forth. A subquery can also be nested inside INSERT, UPDATE, and DELETE statements. Subqueries must be enclosed within parentheses.

A subquery can be used any place where an expression is allowed providing it returns a single value. This means that a subquery that returns a single value can also be listed as an object in a FROM clause listing. This is termed an inline view because when a subquery is used as part of a FROM clause, it is treated like a virtual table or view. Subquery can be placed either in FROM clause, WHERE clause or HAVING clause of the main query.

Oracle allows a maximum nesting of 255 subquery levels in a WHERE clause. There is no limit for nesting subqueries expressed in a FROM clause.In practice, the limit of 255 levels is not really a limit at all because it is rare to encounter subqueries nested beyond three or four levels.

A subquery SELECT statement is very similar to the SELECT statement used to begin a regular or outer query.The complete syntax of a subquery is:

```
( SELECT [DISTINCT] subquery_select_parameter
  FROM {table_name | view_name}
               {table_name | view_name} ...
  [WHERE search_conditions]
  [GROUP BY column_name [,column_name ] ...]
  [HAVING search_conditions] )
```

Types of Subqueries

Single Row Sub Query: Sub query which returns single row output. They mark the usage of single row comparison operators, when used in WHERE conditions.

Multiple row sub query: Sub query returning multiple row output. They make use of multiple row comparison operators like IN, ANY, ALL. There can be sub queries returning multiple columns also.

Correlated Sub Query: Correlated subqueries depend on data provided by the outer query.This type of subquery also includes subqueries that use the EXISTS operator to test the existence of data rows satisfying specified criteria.

Single Row Sub Query

A single-row subquery is used when the outer query's results are based on a single, unknown value. Although this query type is formally called "single-row," the name implies that the query returns multiple columns-but only one row of results. However, a single-row subquery can return only one row of results consisting of only one column to the outer query.

In the below SELECT query, inner SQL returns only one row i.e. the minimum salary for the company. It in turn uses this value to compare salary of all the employees and displays only those, whose salary is equal to minimum salary.

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

A HAVING clause is used when the group results of a query need to be restricted based on some condition. If a subquery's result must be compared with a group function, you must nest the inner query in the outer query's HAVING clause.

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

Multiple Row Sub Query

Multiple-row subqueries are nested queries that can return more than one row of results to the parent query. Multiple-row subqueries are used most commonly in WHERE and HAVING clauses. Since it returns multiple rows,it must be handled by set comparison operators *IN*, *ALL*, *ANY*.While IN operator holds the same meaning as discussed in earlier chapter, ANY operator compares a specified value to each value returned by the sub query while ALL compares a value to every value returned by a sub query.

Below query shows the error when single row sub query returns multiple rows.

```
SELECT first_name, department_id
FROM employees
WHERE department_id = (SELECT department_id
   FROM employees
   WHERE LOCATION_ID = 100)
department_id = (select
                 *
ERROR at line 4:
ORA-01427: single-row subquery returns more than one row
```

Usage of Multiple Row operators

- [> ALL] More than the highest value returned by the subquery
- [< ALL] Less than the lowest value returned by the subquery
- [< ANY] Less than the highest value returned by the subquery
- [> ANY] More than the lowest value returned by the subquery
- [= ANY] Equal to any value returned by the subquery *sameasIN*

Above SQL can be rewritten using IN operator like below.

```
SELECT first_name, department_id
FROM employees
WHERE department_id IN (SELECT department_id
                        FROM departments
                        WHERE LOCATION_ID = 100)
```

Note in the above query, IN matches department ids returned from the sub query,compares it with that in the main query and returns employee's name who satisfy the condition.

A join would be better solution for above query, but for purpose of illustration, sub query has been used in it.

Correlated Sub Query

As opposed to a regular subquery, where the outer query depends on values provided by the inner query,a correlated subquery is one where the inner query depends on values provided by the outer query. This means that in a correlated subquery,the inner query is executed repeatedly, once for each row that might be selected by the outer query.

Correlated subqueries can produce result tables that answer complex management questions.

Consider the below SELECT query. Unlike the subqueries previously considered, the subquery in this SELECT statement cannot be resolved independently of the main query. Notice that the outer query specifies that rows are selected from the employee table with an alias name of e1. The inner query compares the employee department number column *DepartmentNumber* of the employee table

with alias e2 to the same column for the alias table name e1.

```
SELECT EMPLOYEE_ID, salary, department_id
FROM   employees E
WHERE salary > (SELECT AVG(salary)
                FROM   EMP T
                WHERE E.department_id = T.department_id)
```

Multiple Column Sub Query

A multiple-column subquery returns more than one column to the outer query and can be listed in the outer query's FROM, WHERE, or HAVING clause. For example, the below query shows the employee's historical details for the ones whose current salary is in range of 1000 and 2000 and working in department 10 or 20.

```
SELECT first_name, job_id, salary
FROM emp_history
WHERE (salary, department_id) in (SELECT salary, department_id
      FROM employees
       WHERE salary BETWEEN 1000 and 2000
      AND department_id BETWEEN 10 and 20)
ORDER BY first_name;
```

When a multiple-column subquery is used in the outer query's FROM clause, it creates a temporary table that can be referenced by other clauses of the outer query. This temporary table is more formally called an inline view. The subquery's results are treated like any other table in the FROM clause. If the temporary table contains grouped data, the grouped subsets are treated as separate rows of data in a table. Consider the FROM clause in the below query. The inline view formed by the subquery is the data source for the main query.

```
SELECT *
FROM (SELECT salary, department_id
 FROM employees
  WHERE salary BETWEEN 1000 and 2000);
```