

MULTIPLE-COLUMN SUBQUERIES -- CHAPTER 7 --

- Multiple –Column Subqueries: So we have written single-row subqueries where only one column was compared in the WHERE clause or HAVING clauses of the SELECT statement. If you want to compare two or more columns, you must write a compound WHERE clause using logical operators.
- Let’s take a look at the syntax:

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

- Check out the following multiple-column query with subquery...

```
SQL> r
1 select empno, ename, job, sal, deptno
2 from emp
3 where (sal, deptno) in
4 (select sal, deptno
5 from emp
6 where sal between 1000 and 2000 and
7      deptno between 20 and 40)
8* order by empno
```

EMPNO	ENAME	JOB	SAL	DEPTNO
7369	SMITH	ANALYST	1500	20
7499	ALLEN	SALESMAN	1000	30
7521	WARD	SALESMAN	1000	30
7566	JONES	MANAGER	2000	20
7698	BLAKE	MANAGER	2000	30
7788	SCOTT	ANALYST	2000	20
7844	TURNER	SALESMAN	1000	30
7876	ADAMS	ANALYST	1500	20

- In this case, we selected empno, ename, job, sal, and deptno from the ‘emp’ table such that all those employees whose salary was between 1000 and 2000 and their deptno was between 20 and 40. A glance at the data allows us to verify that all records returned by the query satisfy those conditions. Note also that the records are ordered by the empno as dictated by the last line of the subquery.
- In the following example, we add a couple of more lines to the previous query to further restrict the search, and thereby eliminate the records for SMITH and JONES...

```
SQL> r
1 select empno, ename, job, sal, deptno
```

```

2  from emp
3  where (sal, deptno) in
4  (select sal, deptno
5  from emp
6  where sal between 1000 and 2000 and
7  deptno between 20 and 40)
8  and ename <> 'SMITH'
9  and ename <> 'JONES'
10* order by empno

```

EMPNO	ENAME	JOB	SAL	DEPTNO
7499	ALLEN	SALESMAN	1000	30
7521	WARD	SALESMAN	1000	30
7698	BLAKE	MANAGER	2000	30
7788	SCOTT	ANALYST	2000	20
7844	TURNER	SALESMAN	1000	30
7876	ADAMS	ANALYST	1500	20

6 rows selected.

- Column Comparisons: Pairwise Versus Nonpairwise Comparisons
- The above examples are called 'Pairwise Comparisons' since in order to meet the criteria, the records being compared had to meet two conditions simultaneously...i.e., their both their sal 'AND' their deptno had to be in a certain range. If you want a nonpairwise comparison (a cross product), you must use a WHERE clause with multiple conditions. A candidate row must match the multiple conditions in the WHERE clause but the values are compared individually.

```

SQL> r
1  select empno, ename, job, sal, deptno
2  from emp
3  where sal in
4  (select sal
5  from emp
6  where sal between 1000 and 2000)
7  and empno in
8  (select empno
9  from emp
10 where deptno between 20 and 40)
11* order by empno

```

EMPNO	ENAME	JOB	SAL	DEPTNO
7369	SMITH	ANALYST	1500	20
7499	ALLEN	SALESMAN	1000	30
7521	WARD	SALESMAN	1000	30
7566	JONES	MANAGER	2000	20

7698	BLAKE	MANAGER	2000	30
7788	SCOTT	ANALYST	2000	20
7844	TURNER	SALESMAN	1000	30
7876	ADAMS	ANALYST	1500	20

8 rows selected.

- Unfortunately, this is not the best example since it yielded the same results as the pairwise comparison, but theoretically, this is just a coincidence...other times the results may differ between these two types of comparisons.
- NULL values in a subquery. When SQL encounters a NULL value in a comparison of some sort, SQL will simply not return any rows as output. All conditions that compare a null value result in a null. So, whenever null values are likely to be part of the resultant set of a subquery, do not use the NOT IN operator. As an example, compare the following two queries...

```
SQL> select employee.ename
2  from emp employee
3  where employee.empno NOT IN
4  (select manager.mgr
5  from emp manager);
```

no rows selected

Compared to this one...

```
SQL> select employee.ename
2  from emp employee
3  where employee.empno IN
4  (select manager.mgr
5  from emp manager);
```

```
ENAME
-----
JONES
BLAKE
CLARK
SCOTT
KING
REDCORNER
```

6 rows selected.

- Notice that the null value as part of the resultant set of a subquery will not be a problem if you are using the IN operator.
- Finally, let's look at how we can use a subquery in the FROM clause...

```
SQL> r
```

```

1  select a.ename, a.sal, a.deptno, b.salavg
2  from emp a, (select deptno, avg(sal) salavg
3               from emp
4               group by deptno) b
5  where a.deptno = b.deptno
6* and a.sal > b.salavg

```

ENAME	SAL	DEPTNO	SALAVG
KING	3000	10	2000
SCOTT	2000	20	1750
JONES	2000	20	1750
BLAKE	2000	30	1250
SHANK	5500	40	5250
BOBBY	3000	50	1978.8947
"HELLO"	2000	50	1978.8947
BUSH	2000	50	1978.8947
HOMER	2000	50	1978.8947
RED	2000	50	1978.8947
REDCORNER	2000	50	1978.8947
ALBERT	4000	50	1978.8947
TESTTIME	2000	50	1978.8947
NKLGNFGLKB	2000	50	1978.8947
HOLLYO	2000	50	1978.8947
GREEN	2000	50	1978.8947
	2000	50	1978.8947
KWIATKI	2000	50	1978.8947
FREY	2000	50	1978.8947
GORDON	2000	50	1978.8947
DUTT	2000	50	1978.8947

21 rows selected.

- You can use a subquery in the FROM clause of a SELECT statement, which is very similar to how views are used. A subquery in the FROM clause of a SELECT statement defines a data source for that particular SELECT statement, and only that SELECT statement.