**Subqueries**

**Please be aware that MS Access ‘Date’ outputs are formatted slightly differently ‘01/02/2020’ compared to Oracle ’01-FEB-20’**

**Aims of this Section**

Now we shall cover some advanced features of the SELECT statement - queries contained in a predicate clause of another SQL statement.

**Nested Subqueries**

A subquery is a SELECT statement that is nested within another SELECT statement.

For example :

SELECT column1, column2, ......

FROM table

WHERE column = (SELECT column

FROM table

WHERE condition);

The subquery is often referred to as a sub-select or inner select; it generally executes first and its output is used to complete the query condition for the main or outer query.

Using subqueries allows a developer to build powerful commands out of simple ones. The nested subquery 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.

**Single Row Subqueries**

To find the employee who earns the minimum salary in the company (minimum salary is an unknown quantity), two steps must be taken:

1. Find the minimum salary:

SELECT MIN(SAL)

FROM EMP ;

**MIN(SAL)**

**--------**

**800**

2. Find the employee who earns the minimum salary:

SELECT ENAME, JOB, SAL

FROM EMP

WHERE SAL = (lowest salary which is 'unknown')

We can combine the two commands as a nested subquery:

SELECT ENAME, JOB, SAL

FROM EMP

WHERE SAL = (SELECT MIN(SAL)

FROM EMP);

**ENAME JOB SAL**

**----- ----- ---**

**SMITH CLERK 800**

**How Are Subqueries Processed?**

A select statement can be considered as a query block. The example above consists of two query blocks - the outer query and the inner query.

The inner select statement or query block is executed first, producing a query result : 800. The outer query block is then processed, using the value returned by the inner query to complete its search condition. In effect, the outer query would finally look like this:

SELECT ENAME, SAL, DEPTNO

FROM EMP

WHERE SAL = 800;

In the above example, the 800 is a single value. The subquery that returned the value of 800 is called a **single row subquery**. When a subquery returns only one row, a single row comparison or logical operator should be used. For example : =, <, >, <=, etc.

To find all employees who have the same job as BLAKE we would enter:

SELECT ENAME, JOB

FROM EMP

WHERE JOB = (SELECT JOB

FROM EMP

WHERE ENAME = 'BLAKE');

**ENAME JOB**

**----- -------**

**JONES MANAGER**

**BLAKE MANAGER**

**CLARK MANAGER**

The inner query return's Blake's job title (MANAGER) which is then used on the right hand side of the outer query's WHERE clause (after the comparison operator).

**Subqueries That Return More Than One Row**

The following query searches for those employees who earn the lowest salary in each department:

SELECT ENAME, SAL, DEPTNO

FROM EMP

WHERE SAL IN (SELECT MIN(SAL)

FROM EMP

GROUP BY DEPTNO);

**ENAME SAL DEPTNO**

**------ ---- ------**

**SMITH 800 20**

**JAMES 950 30**

**MILLER 1300 10**

Notice that the inner query has a GROUP BY clause. This means that it may return more than one value. We therefore need to use a multi-row comparison operator. In this case, the IN operator MUST be used because it expects a list of values.

The result obtained above does not show the department in which the qualifying employees work. In addition, since we are only comparing salary values, the inner query could return a value simply because it matches the lowest salary for one of the departments, *not necessarily the employee's own department.* Therefore, the query should be rewritten in order to match the combination of employee's salary and department number with the minimum salary and department number.

SELECT ENAME, SAL, DEPTNO

FROM EMP

WHERE (SAL, DEPTNO ) IN

(SELECT MIN(SAL), DEPTNO

FROM EMP

GROUP BY DEPTNO) ;

**With MS Access, we need to concatenate the values**

SELECT ENAME, SAL, DEPTNO

FROM EMP

WHERE SAL & DEPTNO IN

(SELECT MIN(SAL) & DEPTNO

FROM EMP

GROUP BY DEPTNO) ;

**ENAME SAL DEPTNO**

**-------- ---- ------**

**JAMES 950 30**

**SMITH 800 20**

**MILLER 1300 10**

**Subqueries That Return More Than One Row...contd**

The corrected query shown overleaf compares a pair of columns.

Notice that the columns on the LEFT of the search condition are in parentheses and that each column is separated with a comma.

Columns listed on the SELECT clause of the subquery MUST BE IN THE SAME ORDER as the bracketed column list on the WHERE clause of the outer query.

Columns returned by the inner query must also match in number and datatype the columns to which they are compared in the outer query.

Hence, for example:

...WHERE (numcolumn, charcolumn) =

(SELECT datecolumn,

numcolumn,

charcolumn);

is illegal.

When a subquery returns more than one row and a single row comparison operator is used, SQL\*Plus issues the following error message:

SELECT ENAME, SAL, DEPTNO

FROM EMP

WHERE SAL = (SELECT MIN(SAL)

FROM EMP

GROUP BY DEPTNO);

**ERROR : ORA-01427 : single-row subquery returns more than one row**

**no records selected**

MS Access error: At most one record can be returned by this subquery

If the inner query returns no rows, you get the following error message:

SELECT ENAME, JOB

FROM EMP

WHERE JOB = (SELECT JOB

FROM EMP

WHERE ENAME='SMYTHE');

**ERROR : ORA-01426 : single-row subquery returns no rows**

**no records selected**

**ANY or ALL Operators**

The ANY or ALL operators may be used for subqueries that return more than one row. They are used on the WHERE or HAVING clause in conjunction with the logical operators.

ANY compares a value to *EACH* value returned by a subquery.

To display employees who earn more than the lowest salary in Department 30, enter:

SELECT ENAME, SAL, JOB, DEPTNO

FROM EMP

WHERE SAL > ANY (SELECT DISTINCT SAL

FROM EMP

WHERE DEPTNO = 30)

ORDER BY SAL DESC;

**ENAME SAL JOB DEPTNO**

**------ ---- --------- ------**

**KING 5000 PRESIDENT 10**

**FORD 3000 ANALYST 20**

**SCOTT 3000 ANALYST 20**

**JONES 2975 MANAGER 20**

**BLAKE 2850 MANAGER 30**

**CLARK 2450 MANAGER 10**

**ALLEN 1600 SALESMAN 30**

**TURNER 1500 SALESMAN 30**

**MILLER 1300 CLERK 10**

**WARD 1250 SALESMAN 30**

**MARTIN 1250 SALESMAN 30**

**ADAMS 1100 CLERK 20**

The lowest salary in Department 30 is 950 (James). The outer query has returned employees from all departments who earn a salary in excess of this lowest (minimum) salary in department 30. So '> ANY' means more than the minimum. '= ANY' is equivalent to IN.

When using ANY, the DISTINCT keyword is often used to prevent rows being selected several times.

**ANY or ALL Operators...contd**

ALL compares a value to *EVERY* value returned by a subquery.

The following query finds employees who earn more than every employee in Department 30.

SELECT ENAME, SAL, JOB, DEPTNO

FROM EMP

WHERE SAL > ALL (SELECT DISTINCT SAL

FROM EMP

WHERE DEPTNO = 30)

ORDER BY SAL DESC;

**ENAME SAL JOB DEPTNO**

**------ ----- --------- -----**

**KING 5000 PRESIDENT 10**

**SCOTT 3000 ANALYST 20**

**FORD 3000 ANALYST 20**

**JONES 2975 MANAGER 20**

The highest salary in Department 30 is 2850 (Blake), so the query has returned those employees whose salary is higher than 2850. That is, greater than the highest (maximum) salary for Department 30, and consequently more than every salary in that Department.

The NOT operator can be used with IN, ANY or ALL.

**HAVING Clause with Nested Subqueries**

Nested subqueries can also be used in the HAVING clause. However, before we look at their use it may be useful to remind ourselves that WHERE refers to SINGLE rows and HAVING to GROUPS of rows specified in the GROUP BY clause.

For example, to display the department(s) which have an average salary bill greater than Dept. 30, enter:

SELECT DEPTNO, AVG(SAL)

FROM EMP

HAVING AVG(SAL) > (SELECT AVG(SAL)

FROM EMP

WHERE DEPTNO = 30)

GROUP BY DEPTNO;

**MS Access:**

SELECT DEPTNO, AVG(SAL)

FROM EMP

GROUP BY DEPTNO

HAVING AVG(SAL) > (SELECT AVG(SAL)

FROM EMP

WHERE DEPTNO = 30) ;

**DEPTNO AVG(SAL)**

**------ ----------**

**10 2916.66667**

**20 2175**

To construct a query which finds the job with the highest average salary, enter :

SELECT JOB, AVG(SAL)

FROM EMP

HAVING AVG(SAL) = (SELECT MAX(AVG(SAL))

FROM EMP

GROUP BY JOB )

GROUP BY JOB ;

**Nested Aggregate functions not possible in MS Access**

**JOB AVG (SAL)**

**--------- ---------**

**PRESIDENT 5000**

The inner query first finds the average salary for each different job group, and the MAX function picks the highest average salary. That value (5000) is used in the HAVING clause. The GROUP BY clause in the outer query is needed because the outer query's SELECT list contains both an aggregate and non-aggregate column.

**Ordering Data with Subqueries**

The ORDER BY clause is illegal in a subquery.

The rule remains that you can have *ONLY ONE*  ORDER BY clause for a select statement and, if specified, it must be the last clause in the select command.

**Nesting Subqueries**

Subqueries may be nested (used within another subquery). For example, if we wish to display the name, job, and hiredate for employees whose salary is greater than the highest salary in the SALES department, enter:

SELECT ENAME, JOB, HIREDATE, SAL

FROM EMP

WHERE SAL > (SELECT MAX(SAL)

FROM EMP

WHERE DEPTNO = (SELECT DEPTNO

FROM DEPT

WHERE DNAME='SALES'));

**ENAME JOB HIREDATE SAL**

**------ --------- ---------- --------**

**JONES MANAGER 31-OCT-13 2975**

**SCOTT ANALYST 05-MAR-14 3000**

**KING PRESIDENT 09-JUL-14 5000**

**FORD ANALYST 05-DEC-13 3000**

Nesting limits are:

An outer query may have up to 16 subqueries at level 1

Within any level 1 nesting you can nest up to 255 levels of depth:

**Level 1 Level 2 Level 3......255**

**- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -**

SELECT

FROM

WHERE (SELECT

FROM

WHERE (SELECT

FROM

WHERE ))

SELECT

FROM

WHERE

.

up to 16

**Guidelines on Subqueries**

* + The inner query must be enclosed in parentheses, and must be on the right hand side of the condition.
  + The subquery may not have an ORDER BY clause.
  + The ORDER BY clause appears at the end of the main select statement.
  + Multiple columns on the select list of the inner query must be in the same order as the columns appearing on the condition clause of the main query. The datatype and number of columns must also correspond.
  + Set operators may be used in a subquery.
  + Subqueries are always executed from the most deeply nested to the least deeply nested, unless they are correlated subqueries, (discussed later).
  + Logical and SQL operators may be used as well as ANY and ALL.
  + **Subqueries can :**

**-** Return one or more rows

- Return one or more columns

- Use group by or group functions

- Be used in multiple AND or OR predicates of the same outer query

- Join Tables

- Retrieve from a different table than the outer query

- Appear in SELECT, UPDATE, DELETE, INSERT and CREATE TABLE statements

- Correlate with an outer query

**Correlated Subqueries**

A Correlated Subquery is a nested subquery which is executed once for each "candidate row" considered by the main query and which on execution uses a value from a column in the outer query. This causes the correlated subquery to be processed in a different way to the ordinary Nested Subquery.

A Correlated Subquery is identified by the use of an outer query's column in the inner query's predicate clause.

With a normal nested subquery, the inner select runs first and it executes once, returning values to be used by the main query. A Correlated Subquery, on the other hand, executes once for each row (candidate row) considered by the outer query. The inner query is driven by the outer query.

Steps to execute a Correlated Subquery:

1. Get candidate row (collected by outer query).

2. Execute inner query using candidate row's value.

3. Use value(s) resulting from inner query to qualify or disqualify candidate.

4. Repeat until no candidate row remains.

**Why Use a Correlated Subquery ?**

The Correlated Subquery is one way of 'reading' every row in the table, and comparing values in each row against related data. It is used whenever a subquery must return a different result or set of results for each candidate row considered by the main query.

The inner select is normally executed once for each candidate row.

**Correlated Subqueries - A Worked Example**

To find employees who earn a salary greater than the average salary for their department, enter :

SELECT EMPNO, ENAME, SAL, DEPTNO

FROM EMP E

WHERE SAL > (SELECT AVG(SAL)

FROM EMP

WHERE DEPTNO = E. DEPTNO)

ORDER BY DEPTNO ;

**EMPNO ENAME SAL DEPTNO**

**----- ----- ---- ------**

**7839 KING 5000 10**

**7566 JONES 2975 20**

**7788 SCOTT 3000 20**

**7902 FORD 3000 20**

**7499 ALLEN 1600 30**

**7698 BLAKE 2850 30**

Note that the alias is necessary only to avoid ambiguity in column names.

Let us now look in detail at the way the above query uses the EMP table.

**The Main Query**

1. Select first candidate row - Smith in department 20 earning 800.

2. EMP in FROM clause has alias E which qualifies DEPTNO column referenced in inner query's WHERE clause.

3. WHERE clause compares 800 against value returned by inner query.

**The Inner Query**

4. Computes AVG(SAL) for employee's department.

5. WHERE department value is candidate's department (E. DEPTNO) value passed into inner query from outer query's DEPTNO column.

6. AVG(SAL) for Smith's department - 20 - is 2175.

7. Candidate row does not meet condition, so discard.

8. Repeat from step 1 above for next candidate row; ALLEN in department 30 earning 1600.

The selection of candidate rows continues with those meeting the condition appearing in the query result.

**Correlated Subqueries - A Worked Example...contd**

Note :

A Correlated subquery is signaled by a column name, a table name or table alias that refers to the value of a column in each candidate row of the outer select. Also, the Correlated Subquery executes repeatedly for each candidate row in the main query.

**Operators**

When you are nesting select statements the logical operators are all valid as well as ANY and ALL. In addition, the EXISTS operator may be used.

**EXISTS Operator**

The EXISTS operator is often used with Correlated Subqueries. It tests whether a value is there (NOT EXISTS checks that something is not there). If the value exists it returns TRUE, if it does not exist FALSE is flagged.

To find employees who have at least one person reporting to them, enter :

SELECT EMPNO, ENAME, JOB, DEPTNO

FROM EMP E

WHERE EXISTS (SELECT EMPNO

FROM EMP

WHERE EMP. MGR = E.EMPNO)

ORDER BY EMPNO ;

**EMPNO ENAME JOB DEPTNO**

**----- ------- --------- -------**

**7566 JONES MANAGER 20**

**7698 BLAKE MANAGER 30**

**7782 CLARK MANAGER 10**

**7788 SCOTT ANALYST 20**

**7839 KING PRESIDENT 10**

**7902 FORD ANALYST 20**

To find all employees whose department is not in the DEPT table, enter :

SELECT EMPNO, ENAME, DEPTNO

FROM EMP E

WHERE NOT EXISTS (SELECT DEPTNO

FROM DEPT D

WHERE D.DEPTNO=E.DEPTNO)

**no records selected.**

**Tutorial Exercises**

**Please be aware that MS Access ‘Date’ outputs are formatted slightly differently ‘01/02/2020’ compared to Oracle ’01-FEB-20’**

This exercise allows you to write complex queries using nested selects and correlated selects.

**Workshop**

1. Find the employee(s) who earn the highest salary in each job type. Sort in descending salary order.

**JOB ENAME SAL**

**--------- ------ ----**

**PRESIDENT KING 5000**

**ANALYST SCOTT 3000**

**ANALYST FORD 3000**

**MANAGER JONES 2975**

**SALESMAN ALLEN 1600**

**CLERK MILLER 1300**

**6 records selected.**

2. Find the employees who earn the minimum salary for their job. Display the result in ascending order of salary.

**ENAME JOB SAL**

**------ --- ----**

**SMITH CLERK 800**

**MARTIN SALESMAN 1250**

**WARD SALESMAN 1250**

**CLARK MANAGER 2450**

**SCOTT ANALYST 3000**

**FORD ANALYST 3000**

**KING PRESIDENT 5000**

**7 records selected.**

3. Find the most recently hired employees in each department. Sort on hiredate

**DEPTNO ENAME HIREDATE**

**------ ----- ---------**

**20 ADAMS 04-JUN-14**

**10 KING 09-JUL-14**

**30 JAMES 23-JUL-14**

4. Correlated subquery - Show the following details for any employee who earns a salary greater than the average for their department. Sort in department number order*.*

**ENAME SALARY DEPTNO**

**----- ------ ------**

**KING 5000 10**

**JONES 2975 20**

**FORD 3000 20**

**SCOTT 3000 20**

**ALLEN 1600 30**

**BLAKE 2850 30**

**6 records selected.**

5. List all the departments where there are no employees. Use a subquery!

**DEPTNO DNAME**

**------ ----------**

**40 OPERATIONS**