SQL Select Queries - the Beginning

by Clark Anderson

SQL is the acronym for Structured Query Language. SQL has become the language to communicate with many different kinds of relational databases on many kinds of computers, Micro to Mainframe. There are variations in the SQL language. I will discuss the SQL dialect familiar to Microsoft Access users. SQL is used for several primary tasks with databases. The task that is probably performed the most is retrieving data. That is performed by the SELECT query.

The SELECT query can be very simple: **SELECT * FROM MyTable** This will retrieve all of the data from "MyTable" In its simplicity, there is nothing in the query to limit which records (or rows) that are retrieved. The asterisk "*" indicates: retrieve all fields (or columns). There are many ways to limit the records retrieved. One way is to specify conditions that must be met with a WHERE clause: **SELECT * FROM MyTable WHERE State = 'CT'**

GivenName	FamilyName	Shop	City	State	Price of Coffee	Avg_Cans_Per_Day
Abraham	Benson	Moe's	Bloomfield	CT	$5.25	2345
Abraham	Benson	Moe's	Bloomfield	CT	$5.75	1212
Abraham	Benson	Moe's	Bloomfield	CT	$5.25	3121
Abraham	Benson	Moe's	Bloomfield	CT	$5.25	1345

WHERE clauses can specify more details: **SELECT * FROM MyTable WHERE [Price of Coffee]<5.5 AND (State="CO" OR State="CT").** Note that when the field name is more than a single word with spaces in it, it must be specified in SQL with surrounding square brackets, (e.g.: [Price of Coffee]) The square brackets may be used for any name. They help the SQL interpreter clearly understand that anything between a pair of square brackets is a name of a field or table or another query. This example illustrates the use of Boolean logic with its combinations of ANDs and ORs. Careful use of parenthesis is also important to help you and the interpreter clearly understand which records are requested. As in any use of Boolean logic there if often room for optimization: **SELECT * FROM MyTable WHERE (([Price of Coffee]<5.5) AND (State="CO")) OR (([Price of Coffee]<5.5) AND (State="CT")).** Honest, both of these queries obtain the same results.

GivenName	FamilyName	Shop	City	State	Price of Coffee	Avg_Cans_Per_Day
Abraham	Benson	Moe's	Bloomfield	CT	$5.25	2345
Aaron	Barber	Cafe Luna	Longmont	CO	$5.15	1468
Abraham	Benson	Moe's	Bloomfield	CT	$5.25	3121
Abraham	Benson	Moe's	Bloomfield	CT	$5.25	1345

Another neat feature available in the WHERE clause is the IN reserved word: **SELECT * FROM MyTable WHERE [Price of Coffee]<5.5 AND State IN("CO","CT").** This query also returns the same records. The BETWEEN reserved word is also useful: **SELECT * FROM MyTable WHERE [Price of Coffee] BETWEEN 5.25 AND 5.75 AND State IN("CO","CT").** This query returns a slightly different set of records with prices in the range of 5.25 through 5.75.

GivenName	FamilyName	Shop	City	State	Price of Coffee	Avg_Cans_Per_Day
Abraham	Benson	Moe's	Bloomfield	CT	$5.25	2345
Abraham	Benson	Moe's	Bloomfield	CT	$5.75	1212
Abraham	Benson	Moe's	Bloomfield	CT	$5.25	3121
Abraham	Benson	Moe's	Bloomfield	CT	$5.25	1345

In many situations, only some of the fields in a table are wanted. Then it is appropriate to replace the asterisk with a list of the desired fields: **SELECT City, State, [Price of Coffee], Avg_Cans_Per_Day FROM MyTable WHERE State IN ("CO","CT") AND [Price of Coffee] BETWEEN 5.25 AND 5.75.** This query will return only the values in columns named: City, State, [Price of Coffee] and Avg_Cans_Per_Day. It is usually best to request only what you need.

City	State	Price of Coffee	Avg_Cans_Per_Day
Bloomfield	CT	$5.25	2345
Bloomfield	CT	$5.75	1212
Bloomfield	CT	$5.25	3121
Bloomfield	CT	$5.25	1345

You can also control the order of the data records returned: **SELECT City, State, [Price of Coffee], Avg_Cans_Per_Day FROM MyTable WHERE State IN ("CO","CT") ORDER BY [Price of Coffee]** will return the records ordered by ascending Price of Coffee. **SELECT City, State, [Price of Coffee], Avg_Cans_Per_Day FROM MyTable WHERE State IN ("CO","CT") ORDER BY [Price of Coffee] DESC** will list the data ordered by descending Price of Coffee.

City	State	Price of Coffee	Avg_Cans_Per_Day
Boulder	CO	$5.99	1001
Bloomfield	CT	$5.75	1212
Bloomfield	CT	$5.25	1345
Bloomfield	CT	$5.25	3121
Bloomfield	CT	$5.25	2345
Longmont	CO	$5.15	1468

The default sort order is ASC, ascending, (A-Z, 0-9). You can include additional fields in the ORDER BY clause. Records are sorted first by the first field listed after ORDER BY. Records that have equal values in that field are then sorted by the value in the second field listed, and so on.

Additional calculated fields can be derived from the data in the tables: **SELECT City & ", " & State AS City_State, [Price of Coffee], Avg_Cans_Per_Day, [Price of Coffee] * Avg_Cans_Per_Day AS Avg_Dollars_Per_Day FROM MyTable WHERE State In ("CO","CT") ORDER BY [Price of Coffee] DESC , Avg_Cans_Per_Day ASC**. In this query, each record's City text value is concatenated with a constant string and with the State text value. The resulting string is named City_State. Also In this query, each record's value for [Price of Coffee] is multiplied by its value for Avg_Cans_Per_Day and the product is named Avg_Dollars_Per_Day.

City_State	Price of Coffee	Avg_Cans_Per_Day	Avg_Dollars_Per_Day
Boulder, CO	$5.99	1001	$5,995.99
Bloomfield, CT	$5.75	1212	$6,969.00
Bloomfield, CT	$5.25	1345	$7,061.25
Bloomfield, CT	$5.25	2345	$12,311.25
Bloomfield, CT	$5.25	3121	$16,385.25
Longmont, CO	$5.15	1468	$7,560.20

This is a good start. Next time I will begin discussing other types of SQL queries.