MaryJo Sylwester
The Center for Public Integrity
msylwester@publicintegrity.org
NICAR Conference-Philadelphia, March 2002

The Basics of SQL

Structured Query Language (SQL): A computer language designed for communicating with databases; created by IBM in the early 1970s. Unlike other languages, SQL is made up of very few words. Most database managers support SQL, although there are some slight variations in syntax.

Do I need it?	**In the SQL View:**	**In the Grid View:**
Required	SELECT	Field:
Required	FROM	Table:
Optional	WHERE	Criteria:
When Necessary	GROUP BY	Total:
Optional	ORDER BY	Sort:

SELECT: This is where you tell the computer what pieces of information you want to show in your answer ("vertical" filtering). Use an asterisk (*) to return all fields. Otherwise list selected fields in whatever order you like, separated by commas. Use the word "AS" to rename a column (e.g. "sum(amount) AS total")

FROM: This is where you specify which table the information is coming from. You will only list one table, unless you are JOINING tables. If you are joining, you would list each table, separated by commas.

WHERE: This allows you to do "horizontal" filtering. In other words, you can choose to show only certain records in your answer. For example, you might want to only see records where the address is listed as being in Hawaii. Then you would say Where state = "HI". This uses Boolean logic (and, or, not) and can include many different criteria. You can add on to the previous example and say Where state= "HI" or state= "GA". This would give you records from both states. Reminder: you must repeat the name of the field for each criteria. Also don't forget that you can specify a criteria in the where line, even if that field does not appear in the select line. Text values must be in quotes; dates surrounded by pound signs (#)

GROUP BY: This is only used when you are summarizing (or looking for trends). With this line, you will tell the computer to put your data in piles based on a criteria that you assign on the GROUP BY line. For example, if you're working with campaign contributions, you could ask the computer to lump the individual contributions together by the industry of the contributor. The group by line would say something like this…. "GROUP BY occupation". Your answer will give you one line for each occupation found in the data. Rule of Thumb: whatever criteria you group by, must be included in the Select line. Then you can build on this and ask for further information in the SELECT line (e.g. sum the amount of money from each occupation or count the number of records for each occupation). Word of warning: when you're using Group By, your answer will always be some type of summary. You can't get detail at the same time.

ORDER BY: This puts your answer in an order based on whichever field you want. You do this, by telling it which column in your SELECT line to sort it by, then saying whether you want it descending (DESC) or ascending (leave it blank). This can be done with a shortcut…referring to each column as "1", "2", and so on. So if you want to list occupations, in the above example, going from the largest total contributions to the smallest, you would order it by the second column. Your order by line would say: ORDER BY 2 desc. You can also sort by a column that does not appear in the Select line.

MORE ADVANCED STUFF:

HAVING: This is another line of SQL that comes after GROUP BY and before ORDER BY. It is used to dictate what results of the query will be shown. It's most often used to find the top 10 or top 25 of an answer. For example, you could use the following query to return only the list of occupations that show up more than 50 times in the data:

```
SELECT occupation, count(*)
FROM tablename
GROUP BY occupation
HAVING count(*)>50
ORDER BY count(*) desc
```

JOINING TABLES: The primary way to join tables is in the WHERE clause. Using this method will return only those records where the specified criteria match in both tables. (It's possible to do a more advanced join where you get all records from one table and only those that match from another….see OUTER JOIN in the Access Help menu). It's often necessary to list the name of the table in front of the field names referred to in your SQL, especially if there are fields named the same in both tables. If there is more than one join criteria use the "AND" operator.

```
SELECT table1.fieldname1, table2.fieldname1, table1.fieldname2, table1.fieldname3
FROM table1, table2
WHERE table1.joinfield=table2.jo infield
```

MATHEMATICAL FUNCTIONS:

COUNT(*): This counts how many occurrences of something are in the data. You can determine what it counts through GROUP BY and/or WHERE clauses. Using the asterisk will return NULL or blank values. If you put a field name in the parentheses, it will not show you the blanks (plus it takes more typing!)

SUM(fieldname): This can only be used on numeric fields. You must include the name of that field storing the numeric information in the parentheses.

AVG(fieldname): This can only be used on numeric fields. You must include the name of that field in the parentheses. For example, you would use this if you want to find the average age of the victims in a hunting database. Without a group by, the answer would reveal the average age of all victims. Including a group by (e.g. on the county field), your answer would show the average victim age in each county.

MIN(fieldname): Returns the lowest value, if used with a numeric field; or returns the last value (end of the alphabet) if it's alphanumeric.

MAX(fieldname): Returns the highest value, if used with a numeric field; or returns the first value (start of the alphabet) if it's alphanumeric.

DATE FUNCTIONS:

YEAR(datefield): Returns only the year portion of a date.

MONTH(datefield): Returns only the month portion of a date. Result will be numbers, e.g. 12 for the month of December.

DAY(datefield): Returns only the day portion of date in numeric value, (e.g. 22 for 3/22/99)

FORMAT(datefield,"ddd"): Returns the day of the week in the three-letter abbreviation (e.g. Tue, Wed.).

FORMAT (datefield, "mmm"): Returns the month in a three-letter abbreviation (e.g. Oct.)

CLEAN UP:

TRIM(fieldname): Use this in an Update Query to trim leading and trailing spaces in a certain field. (be sure to put your results in a new field, and not over the old data in case something goes wrong!). You might also want to use this when joining tables to help it work better (e.g. trim(field1)=trim(field2)). In FoxPro, use ALLTRIM to trim both sides.

UPPER(fieldname): Use this in an Update Query to convert all words to uppercase. This is not as crucial in Access, because it isn't case sensitive. Other database managers (such as FoxPro) think DEMOCRAT and Democrat are two different words.

CHANGING DATA:

Access has built-in queries for the major SQL commands used to change data: Append, Update and Delete. (In Fox Pro, Replace All is used instead of Update.) Here's the SQL behind them.

UPDATE: Use to make changes to existing data (words of wisdom…create a new field and make the changes there). The basic syntax is:

```
UPDATE tablename SET fieldname =
```

After the equal sign you specify what you want filled into that column. This could be one value that will appear in all records. (e.g. UPDATE tablename SET fieldname = "STATE"). Or it could be an exact copy of another field (e.g. UPDATE tablename SET fieldname = fieldname2). Or you can use string functions to take portions of another field, or concatenate to combine two fields.

You can add a where clause to the end of the command to limit the update to only certain records.

```
UPDATE tablename SET fieldname = fieldname1 where state= "MN"
```

DELETE: Use to delete existing data. Use the where clause to specify the criteria used to determine what data to delete. For example, if you want to delete all the records where the state is listed as "MN" your query would look like this:

```
DELETE *
FROM TENNGIVE
WHERE STATE="MN"
```

APPEND: Use when you have two or more tables – typically with the same fields of information – that you want to merge together. The result will be that the records from the first table will be followed by the records in the second table. You must tell Access which fields should be matched between the two tables.

MORE INFORMATION:

"SAMS Teach Yourself SQL in 10 Minutes", by Ben Forta. This is designed to encompass all database manager types, but the majority is written for SQL Server or Oracle. However, it's still the most highly recommended SQL book on the market.

Introduction to Structured Query Language, web site by James Hoffman:
http://www.geocities.com/SiliconValley/Vista/2207/sql1.html

SQL: The Universal Database Language:
http://www.zdnet.com/pcmag/pctech/content/17/19/tf1719.001.html