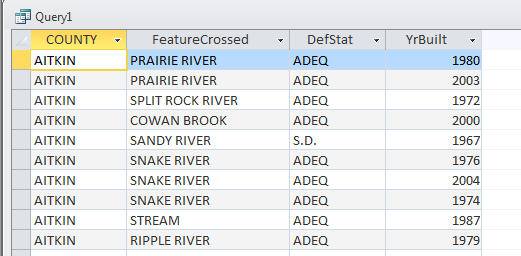
Secrets to SQL query construction

The fields that you list in the SELECT line will be what fields are displayed in your answer

SELECT COUNTY, FeatureCrossed, DefStat, YrBuilt

FROM Bridges2009



You can add an ORDER BY if you want to sort the fields. You can either name the field specifically or you can use a number to specify the column you want to sort by (in this case the county field is column 1, FeatureCrossed is column 2, etc. It’s just based on the columns that you are displaying in your SELECT line). You can add “desc” to have it go largest to smallest (or newest to oldest). The default is ascending, so you don’t need to specify that. So in this case if we want to sort by the YrBuilt, here are two ways to do that query:

SELECT COUNTY, FeatureCrossed, DefStat, YrBuilt

FROM Bridges2009  
ORDER BY YrBuilt

Or

SELECT COUNTY, FeatureCrossed, DefStat, YrBuilt

FROM Bridges2009

ORDER BY 4 desc

Only use the WHERE line if you want to limit which records will be included in your answer. For example, if you only want to see the bridges in Hennepin County:

SELECT COUNTY, FeatureCrossed, DefStat, YrBuilt  
FROM Bridges2009  
where county="HENNEPIN"

WHERE is very much like searching in Google. You can use Boolean operators to specify what to include or what to avoid. Keep in mind that the more detailed your WHERE line is, the greater likelihood that you might miss records you want.

The queries listed above are essentially just filtering the data and allowing you to see various subsets (only certain fields or only certain records or both). It is possible to count things that way, but only one thing at a time. So for example, if we want to find out how many bridges are structurally deficient, we could run this query, and we can see from the record count that we have 1,170 structurally deficient bridges in Minnesota.

SELECT COUNTY, FeatureCrossed, DefStat, YrBuilt

FROM Bridges2009

where defstat="S.D."

However, there are a couple flaws in this approach. The first is that we are assuming that all records in our dataset have “S.D.” in the DefStat field when it is structurally deficient. What if there is one record where it doesn’t have the periods and it instead says “SD”? The query above will miss that one. The second, is that most of the time we want to put that count (in this case, the number of deficient bridges) into context – so we’d want to see how many total bridges there are, as well.

A better approach is to summarize all the records, based on what is stored in a specific field. So for example, if we want to see how many structurally deficient bridges there are – the first step you do is figure out which field(s) identify a bridges’ rating. In this case, that information is housed in the DefStat field. So we want to count up each of the rating types and see how many bridges there are for each. This requires a summary – or “Group by” – query.

SELECT DefStat, count(\*)  
From Bridges2009  
Group by DefStat

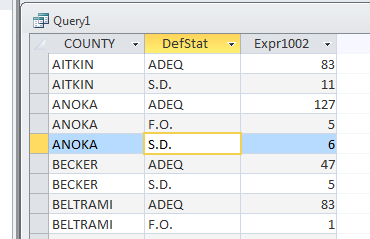
Key point: Group by queries tend to be very sparse in the SQL. Think about what your answer is going to be – you want to see a list of each DefStat category in one column and a second column showing the number of bridges in each group. That means you will only have 2 items in your SELECT line. The Group By line is required to tell Access how to “group” the records – imagine Access sending all the records with “S.D.” into one corner of a gymnasium, and then all the records with “F.O.” into another corner, and all the records with “ADEQ” into another corner. That’s what this line of SQL is doing.

In this case, we don’t want a WHERE line because we want it to tally ALL the records in our database. You CAN include a WHERE line if you want to limit what’s included. For example, you might want to know what’s going on with bridges just in Hennepin County….

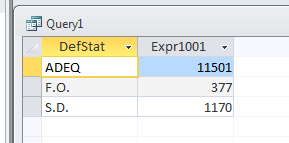
SELECT DefStat, count(\*)  
From Bridges2009  
WHERE county=”Hennepin”  
Group by DefStat

Beginners often worry that they didn’t get the right answer in their group by query. Or they want to see more information – like they want to know what county each deficient bridge is located in. The problem is that once you start adding more information into a Group by query’s SELECT line, you start mucking with the groups.

For example…. If we add the county field to our first group by query. Remember, the rule is that whatever fields are in your SELECT line (except the math field), needs to be in your GROUP BY line as well.

SELECT county, defstat, count(\*)  
FROM bridges2009  
GROUP BY county, defstat  
  
Now you end up with multiple records for each county because we told Access to make the groups (think of the corners of the gymnasium) based on two variables – what county it’s in and what it’s DefStat rating is.

A better way to do something like this is a Crosstab query – which we’ll get to a bit later.

Let’s talk about some ways to check your work. So remember the query that tallied up the number of bridges based on the DefStat field?

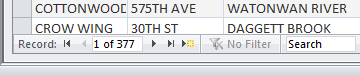
SELECT DefStat, count(\*)  
From Bridges2009  
Group by DefStat

We have 377 “F.O” (functionally obsolete) bridges. Let’s go see if we can replicate that number using a more basic query:

Select \*  
FROM bridges2009  
where DefStat like “F\*”

This query is going to return all the columns/fields (that’s what the asterisk does in the SELECT line) and only the records/rows that have a value in the DefStat field that starts with the letter F (the asterisk in the where line is a wildcard)

Check the record count (lower left corner of the table) and you should have the same number of records – 377.



COUNT, SUM and AVG functions

Our group by queries above, used COUNT(\*) to tally up the number of bridges. This function merely counts the records that are each group (each corner of the gymnasium). The asterisk in there is kind of like a way of saying “it doesn’t matter what field you use to do the counting.” It is possible to put a field name in there, instead of the asterisk. But I don’t recommend it. The reason is that if you count on a specific field – and if that field has any records with a NULL (blank) value – your answer will not include that record. And this is, unfortunately, way too common with datasets. This bridges data is a rare one in that all the fields are filled out.

The vast, vast majority of time you will be using the COUNT function for your group by queries.

The SUM function will add together values that are stored in a field in your table. It REQUIRES you to specify a field. And the SUM function only works on numeric fields. So in the cases of our bridges data, we have two numeric fields – the SuffRate and the Length. Neither one make sense to use SUM on. Why we would we want to add up all the lengths of all the bridges in each county, for example?

Most of the time you will use SUM if you have a currency/dollar field (i.e. campaign finance donations data) or if you have a field that lists a number of something (i.e. school data showing total enrollment by school and you want to get total enrollment for all the schools in each district)

AVG (average) also requires you to use a specific field and it also must be a numeric field. The two numeric fields in the Bridges data are perfect examples for when you might use this. This query below will tell you the average length of the bridges in each county.

SELECT county, avg(length)  
FROM Bridges2009  
group by county  
order by 2 desc

There are some more advanced types of queries you can do. One that you might find particularly useful is a Crosstab query. Remember we wanted to see the deficiency categories by county? That concept of “crossing” two variables – x by x – is when you use Crosstab.

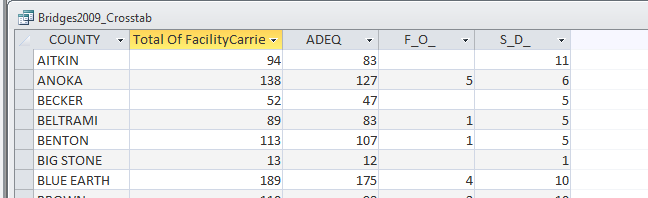
Access has a “query wizard” that helps you set up a Crosstab query, so you don’t have to write the SQL.

Under the “Create” ribbon, choose “Query Wizard” and then in the dialog box that pops up, choose “Crosstab Query Wizard” and push OK.

The first step is going to ask you what table or query you want to use.

The second step is going to ask what you want each ROW in your answer to represent. I’d recommend picking the one that has more variables. In this case we have 87 counties and our DefStat field has 3 values. So let’s go with County for the rows.

The third step asks what you want the columns to be – here’s where you put your second variable; in this case DefStat

And the fourth step asks you to specify the “data” that is going to be summarized. In this case we want to count the records – so you pick Count from the list of functions. And it requires you to pick a field to count – just make sure to pick one that doesn’t have any NULL or blank values.

It will automatically add a “total” column for you, unless you uncheck the box on this screen.