# Fun with th HAVING Clause

Posted At : October 23, 2006 1:59 PM | Posted By : Mark Kruger
Related Categories: SQL tips

You probably already know about the GROUP BY clause in SQL - but have you ever tried "HAVING"? The group by clause is great for getting aggregate groups of information together. Let's say you want a count of how many times a keyword appears in a table. You have the words car, auto, van and bus all in a keywords table.

## sample data

| id | keyword |
|----|---------|
| 1  | auto    |
| 2  | auto    |
| 3  | auto    |
| 4  | van     |
| 5  | bus     |
| 6  | truck   |
| 7  | taxi    |
| 8  | car     |
| 9  | car     |

## A count of the totals is easy using group by:

```
<cfquery name="getkeyword" datasource="#mydsn#">
    SELECT  count(id) AS tot, keyword
    FROM     keywordTest
    GROUP    BY keyword
    ORDER BY tot desc
</cfquery>
```

**query - Top 6 of 6 Rows**

|   | KEYWORD | TOT |
|---|---------|-----|
| 1 | bus     | 1   |
| 2 | taxi    | 1   |
| 3 | truck   | 1   |
| 4 | van     | 1   |
| 5 | car     | 2   |
| 6 | auto    | 3   |

But what if you get tired of looking at all the single search words? What if you wanted to get at only the words where there are more than 3 entries? That's where "HAVING" comes in.

## GROUP By With HAVING

The *HAVING* clause works *after the fact*. In other words, it doesn't do it's thing until *after* the result set is built. That is how it works its magic. If we change the above query to:

```
<cfquery name="getkeyword" datasource="#mydsn#">
    SELECT count(id) AS tot, keyword
    FROM     keywordTest
    GROUP    BY keyword
    HAVING count(id) > 2
    ORDER BY tot desc
</cfquery>
```

We will end up with just 1 result - "auto" with a total of 3 entries. One word of caution. "HAVING" makes your database work. Remember it must first retrieve the results, then filter them, then order them. So it should be used with care.

## Another Clever Use for HAVING

A client recently asked for help with a complicated query using keywords. In his case he had pictures tied to keywords with a one-to-many relationship. In other words, a single picture could match many keywords. When a user searched for "auto and taxi" the client wanted the result set to contain only items that had *both auto and taxi* attached as keywords. Having gave us the ability to do it. Here's the query (with the names changed to protect the innocent).

```
<cfquery datasource="#mydsn#" name="kw">
SELECT I.cID, I.picName, I.picNumber
FROM pics I INNER JOIN pic_keyword_rel R
ON i.picID = r.picID
INNER JOIN keyw K ON r.kwID = k.kwID
WHERE r.cID = #someval#
AND k.kword IN (<cfqueryparam value="auto,taxi" cfsqltype="cf_sql_varchar" list="yes">)
GROUP BY I.cID, I.picName, I.picNumber
HAVING count(i.picNumber) = <cfqueryparam cfsqltype="CF_SQL_INTEGER"
value="#listlen('auto,taxi')#"/>
</cfquery>
```

Nifty eh? This query builds a group of pics and then filters out the ones that don't match *all* of the words in the list. I can think of no easier SQL way to do this (but I'm prepared to be enlightened :).