Get Ready to Learn SQL Server 23: Using Subqueries in the HAVING Clause

This is the fifth in a series of articles about subqueries. In this article we discuss subqueries in the HAVING clause. Other articles discuss their uses in other clauses.

All the examples for this lesson are based on Microsoft SQL Server Management Studio and the AdventureWorks2012 database. You can get started using these free tools using my *Guide Getting Started Using SQL Server*.

Using Subqueries in the HAVING Clause

You can use sub queries in the HAVING clause to filter out groups of records. Just as the WHERE clause is used to filter rows of records, the HAVING clause is used to filter groups. Because of this, it becomes very useful in filtering on aggregate values such as averages, summations, and count.

The power of using a subquery in the HAVING clause is now you don't have to hard-code values within the comparisons. You can rely on the subquery's results to do so for you.

For example, it is now possible to compare the average of a group to the overall average. We've always been able to use the average of the group in the HAVING clause, but had no way to cmputer the overall average. Now, using subqueries, this is possible.

In this example we're selecting employee job titles having remaining vacation hours greater than the overall average for all employees.

Here is the query written without the subquery

```
SELECT   JobTitle,
         AVG(VacationHours) AS AverageVacationHours
FROM     HumanResources.Employee
GROUP BY JobTitle
HAVING   AVG(VacationHours) > 50
```

I've highlighted the value in red that will be replaced by a subquery.

Now here is the complete statement including the subquery:

好的示例。

避免分两次查询。也避免在 HAVING 中使用 hard-code 的总平均 VacationHour 数据。

```
SELECT   JobTitle,
         AVG(VacationHours) AS AverageVacationHours
FROM     HumanResources.Employee
GROUP BY JobTitle
HAVING   AVG(VacationHours) > (SELECT AVG(VacationHours)
                               FROM   HumanResources.Employee)
```

This query is executed as:

1. Compute the remaining average vacation hours for all employees. (subquery)
2. Group records by JobTitle and computer the average vacation hours.
3. Only keep groups whose average vacation hours are greater than the overall average.

Correlated Subqueries in HAVING Clause

As with any other subquery, subqueries in the HAVING clause can be correlated with fields from the outer query.

Suppose we further group the job titles by marital status and only want to keep those combinations of job titles and martial statuses whose vacation hours are greater than those for their corresponding overall marital status?

In other words, we want to answer a question similar to “do married accountants have, on average, more remaining vacation, than married employees in general?”

One way to find out is to us the following query:

找出某一婚姻状态下，哪些 JobTitle 的平均休假时间大于该婚姻状态下全部雇员的平均休假时间。

```
SELECT   JobTitle,
         MaritalStatus,
         AVG(VacationHours)
FROM     HumanResources.Employee AS E
GROUP BY JobTitle, MaritalStatus
HAVING   AVG(VacationHours) >
            (SELECT AVG(VacationHours)
             FROM   HumanResources.Employee
             WHERE  HumanResources.Employee. MaritalStatus =
                    E.MaritalStatus)
```

There are a couple of things to point out. First, notice that I aliased the Employee as “E” in the outer query. This allows me to reference the outer table within the inner query.

Also, with the correlated query, only fields used in the GROUP BY can be used in the inner query. For instance, for kicks and grins, I tried replacing MaritalStatus with Gender and got an error.

```
SELECT   JobTitle,
         MaritalStatus,
         AVG(VacationHours)
FROM     HumanResources.Employee AS E
GROUP BY JobTitle, MaritalStatus
HAVING   AVG(VacationHours) >
            (SELECT AVG(VacationHours)
             FROM   HumanResources.Employee
             WHERE  HumanResources.Employee. Gender =
                    E. Gender)
```

Is a broken query. If you try to run it you’ll get the following error:

Column ‘HumanResources.Employee.Gender’ is invalid in the HAVING clause because it is not contained in either an aggregate function or the GROUP BY clause.

Summary

One advantage of using a subquery in the HAVING clause is to avoid hard coding values, such as an overall average, which can can change and are easily

computed.

As with other queries, it is possible to build correlated subqueries in the HAVING clause. This can be useful when the subquery is dependent on the outer query’s column values, and may make it easier to initially understand a query; however, care should be taken! As with all SQL there are usually many ways to write a query to return the same result. If performance is a concern, then use query plans to understand performance and explore alternatives.

Share Tweet Pin