Table of Contents

We have done a bit of work with subqueries in the Where clause so some of this is a review.

The purpose of using a subquery is to let the subquery determine values from the database that can then be used by another query to get the final desired result. We want to automate this as a single query rather than running two separate queries and manually transferring the answer from one query to the second.

Subqueries are also called nested queries. There are a variety of places where you can use subqueries. Subqueries can return different shapes of result tables. And subqueries can be a self contained query or rely on the outer query for processing. So first we will talk about the various types of subqueries and then go on to the examples.

SQL is a redundant language; sometimes there is more than one way to accomplish a task. Subqueries can be used instead of joins for some queries. In the past, subqueries were considered to be more efficient than joins. But with the current database engines, joins may be just as, or more, efficient. In many queries, joins can be easier to understand than subqueries.

1. Classification of subqueries

1.1. Classification of subqueries by usage

A subquery is a Select query that is placed inside another query. We use the terms inner and outer query for this; we also use the terms parent query and child query. Queries can be nested several layers deep.

You can nest a subquery within the

- Where clause of the parent query
- Select clause of the parent query
- From clause of the parent query
- Having clause of the parent
- Where clause of Update, Insert, and Delete statements; we have seen examples of this in the section on Action queries.

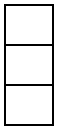
1.2. Classification of subqueries by return table shape

Subqueries can be classified by their return table:

- scalar- return a single value (one row and one column)

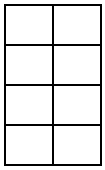
You can test this subquery with an equality test.

- multi-row- return a single column and possibly multiple rows



If you have a subquery that can return multiple rows with a single column, then you have to use the subquery with a test where multiple rows are acceptable. You often use these subqueries in the where clause of a query. You cannot test this return set with the equality (=) operator but you can use the IN list operator. It makes sense to ask if a column value is in the data returned by a multi-row subquery.

- multi-column-return a multi-column, multi-row table



This is a subquery that can return multiple columns and multiple rows- so it is essentially returning a virtual table that could be used as a table expression in the from clause. In this case the subquery is sometimes called an in-line view. The subquery does not have to be the only table expression in the From clause; you can join the subquery result to a regular table to do a join.

1.3. Classification as correlated or non correlated

Subqueries can be classified as:

- non-correlated
- correlated

A non-correlated subquery can "stand by itself". When you think about a non-correlated query, you should think of the inner query being evaluated and returning a value, possibly a table value, which is then given to the parent query.

With a correlated query, the child query cannot stand on its own. The child query refers to columns in the parent query. With a correlated query, you should think of the parent query operating on its first row, then evaluating the child query; then the parent query works on its next row and re-evaluates the child query. This means that a correlated subquery can be very inefficient; you should use other techniques if possible.