10/04/2012 • 2 minutes to read

In this article

Using Common Table Expressions

A common table expression (CTE) can be thought of as a temporary result set that is defined within the execution scope of a single SELECT, INSERT, UPDATE, DELETE, or CREATE VIEW statement. A CTE is similar to a derived table in that it is not stored as an object and lasts only for the duration of the query. Unlike a derived table, a CTE can be self-referencing and can be referenced multiple times in the same query.

A CTE can be used to:

- Create a recursive query. For more information, see Recursive Queries Using Common Table Expressions.
- Substitute for a view when the general use of a view is not required; that is, you do not have to store the definition in metadata.
- Enable grouping by a column that is derived from a scalar subselect, or a function that is either not deterministic or has external access.
- Reference the resulting table multiple times in the same statement.

Using a CTE offers the advantages of improved readability and ease in maintenance of complex queries. The query can be divided into separate, simple, logical building blocks. These simple blocks can then be used to build more complex, interim CTEs until the final result set is generated.

CTEs can be defined in user-defined routines, such as functions, stored procedures, triggers, or views.

Structure of a CTE

A CTE is made up of an expression name representing the CTE, an optional column list, and a query defining the CTE. After a CTE is defined, it can be referenced like a table or view can in a SELECT, INSERT, UPDATE, or DELETE statement. A CTE can also be used in a CREATE VIEW statement as part of its defining SELECT statement.

The basic syntax structure for a CTE is:

WITH expression_name [(column_name [,...n])]

AS

(CTE_query_definition)

The list of column names is optional only if distinct names for all resulting columns are supplied in the query definition.

The statement to run the CTE is:

SELECT <column_list>

FROM expression_name;

Example

The following example shows the components of the CTE structure: expression name, column list, and query. The CTE expression Sales_CTE has three columns (SalesPersonID, SalesOrderID, and OrderDate) and is defined as the total number of sales orders per year for each salesperson.

```
USE AdventureWorks2008R2;
GO
-- Define the CTE expression name and column list.
WITH Sales_CTE (SalesPersonID, SalesOrderID, SalesYear)
AS
-- Define the CTE query.
(
    SELECT SalesPersonID, SalesOrderID, YEAR(OrderDate) AS SalesYear
    FROM Sales.SalesOrderHeader
    WHERE SalesPersonID IS NOT NULL
)
-- Define the outer query referencing the CTE name.
SELECT SalesPersonID, COUNT(SalesOrderID) AS TotalSales, SalesYear
FROM Sales_CTE
GROUP BY SalesYear, SalesPersonID
ORDER BY SalesPersonID, SalesYear;
GO
```

Here is a partial result set:

```
SalesPersonID TotalSales  SalesYear

------------- ----------- -----------

274         4          2001

274         20          2002

274         14          2003

274         10          2004

275         56          2001

275         139          2002

275         169          2003
```

See Also

Reference

WITH common_table_expression (Transact-SQL)