

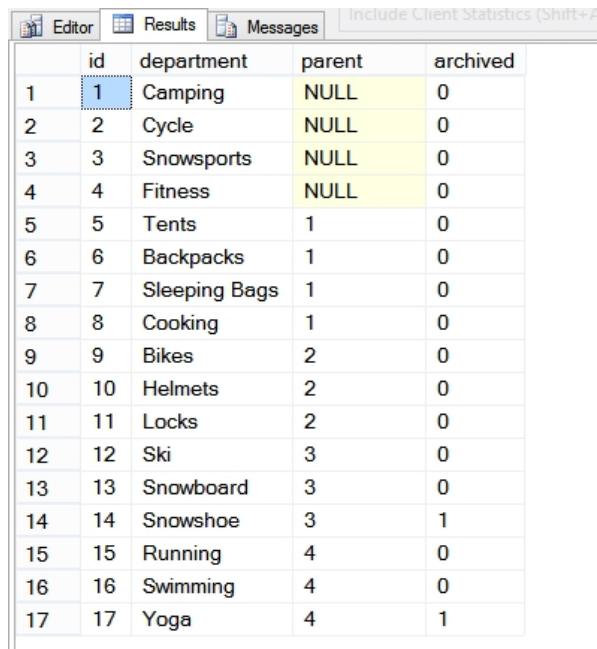
## Writing Your First Common Table Expression with SQL Server

Day 2 of Common Table Expression Month (June) at [SteveStedman.com](http://SteveStedman.com), today I will cover creating your very first CTE.

Keep in mind that this may look a bit weird or strange at first, the syntax is different from anything you may have seen with SQL Server before, and there are some parts that are very similar.

First lets take a look at a Non-CTE query using a derived table subquery. These queries will be using that database that was set up in a previous posting on the [CTE DEMO Sample Database](#), if you haven't set up the sample database, download it and set it up now.

First a simple SELECT statement to see what is in the Departments table. We will be using the Departments table in the CTE queries.



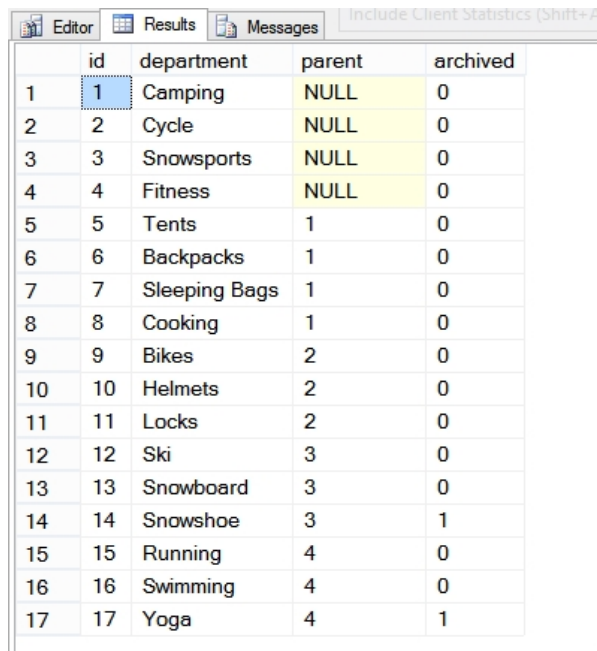
The screenshot shows the 'Results' tab in SQL Server Enterprise Manager. It displays a table with 5 columns: 'id', 'department', 'parent', and 'archived'. The 'id' column is highlighted with a blue selection box around the value '1'. The table contains 17 rows of data, showing a hierarchy of departments. The 'parent' column contains NULL values for the first four rows and numeric values for the remaining rows, indicating a parent-child relationship. The 'archived' column contains 0 or 1, indicating whether a department is archived.

	id	department	parent	archived
1	1	Camping	NULL	0
2	2	Cycle	NULL	0
3	3	Snowsports	NULL	0
4	4	Fitness	NULL	0
5	5	Tents	1	0
6	6	Backpacks	1	0
7	7	Sleeping Bags	1	0
8	8	Cooking	1	0
9	9	Bikes	2	0
10	10	Helmets	2	0
11	11	Locks	2	0
12	12	Ski	3	0
13	13	Snowboard	3	0
14	14	Snowshoe	3	1
15	15	Running	4	0
16	16	Swimming	4	0
17	17	Yoga	4	1

### Derived Table Example

A derived table, often referred to as a subquery is the placement of a query inside of another query instead of a table name. It looks like this:

It is likely that if you have used T-SQL that this may be familiar to you. When the derived table query is run, it produces the same thing as the original query because we are just selecting everything from the original query, however derived table queries are often more complex than this example.



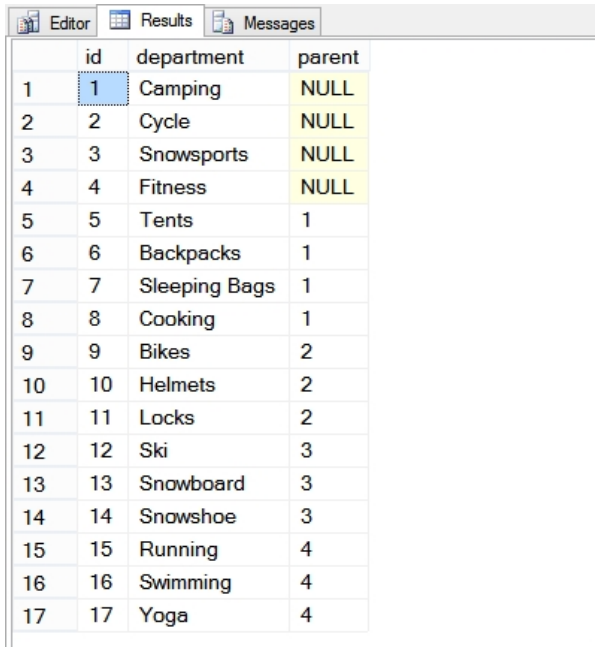
	id	department	parent	archived
1	1	Camping	NULL	0
2	2	Cycle	NULL	0
3	3	Snowsports	NULL	0
4	4	Fitness	NULL	0
5	5	Tents	1	0
6	6	Backpacks	1	0
7	7	Sleeping Bags	1	0
8	8	Cooking	1	0
9	9	Bikes	2	0
10	10	Helmets	2	0
11	11	Locks	2	0
12	12	Ski	3	0
13	13	Snowboard	3	0
14	14	Snowshoe	3	1
15	15	Running	4	0
16	16	Swimming	4	0
17	17	Yoga	4	1

In the derived table, example it doesn't do anything more than the original query, it is just a quick introduction to the derived table.

## CTE Query

Similar to the derived table example, your first CTE will be very straightforward. The CTE query starts with the WITH keyword instead of a SELECT keyword.

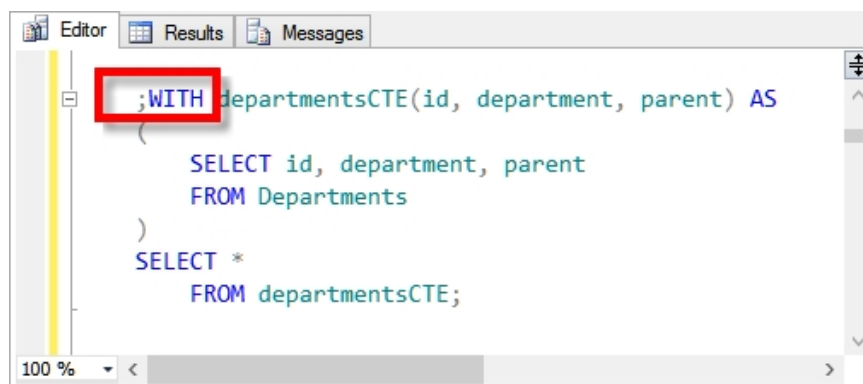
Which produces the following results:



	id	department	parent
1	1	Camping	NULL
2	2	Cycle	NULL
3	3	Snowsports	NULL
4	4	Fitness	NULL
5	5	Tents	1
6	6	Backpacks	1
7	7	Sleeping Bags	1
8	8	Cooking	1
9	9	Bikes	2
10	10	Helmets	2
11	11	Locks	2
12	12	Ski	3
13	13	Snowboard	3
14	14	Snowshoe	3
15	15	Running	4
16	16	Swimming	4
17	17	Yoga	4

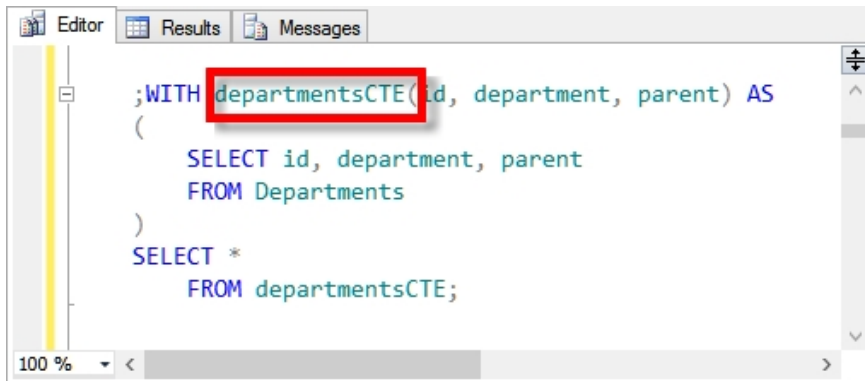
You could just copy and paste the CTE query above and run it in SQL Server Management Studio, but first lets look at the parts that make up the CTE.

First the CTE starts with a WITH statement. You can included a ; in front of the WITH statement to help ensure that the previous statement was properly terminated.

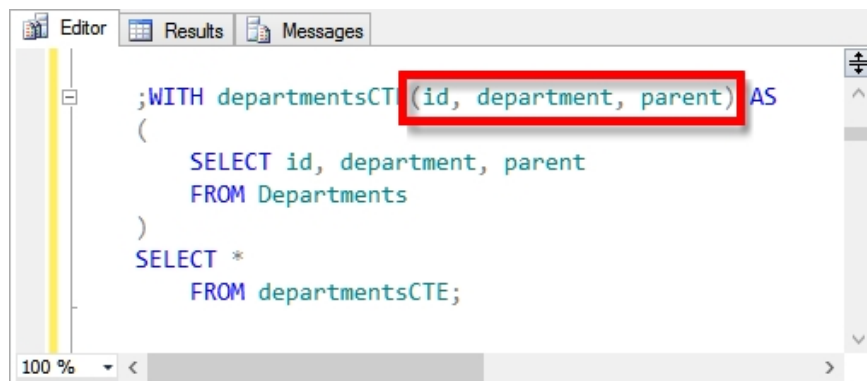


```
Editor Results Messages
;WITH departmentsCTE(id, department, parent) AS
(
    SELECT id, department, parent
    FROM Departments
)
SELECT *
FROM departmentsCTE;
```

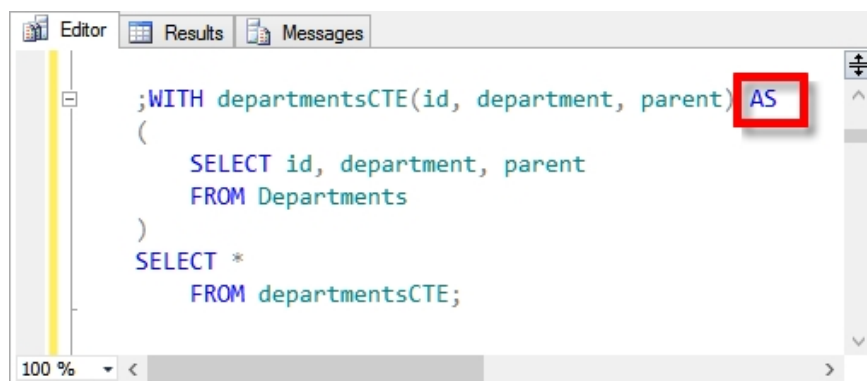
Next we name the CTE, in this case the name of the CTE is departmentsCTE. You can tack the CTE acronym on to the end of the CTE name to avoid confusion with similar table names.



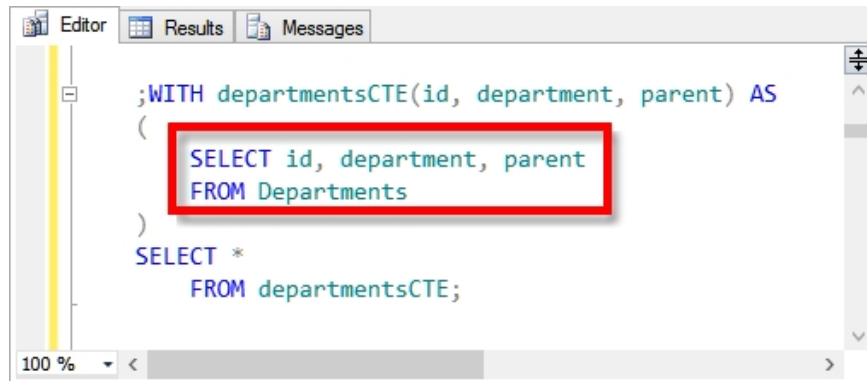
Next is the optional columns list. For a simple CTE, this column list can be skipped and is not required.



Then the AS keyword separates the CTE declaration from the CTE query.

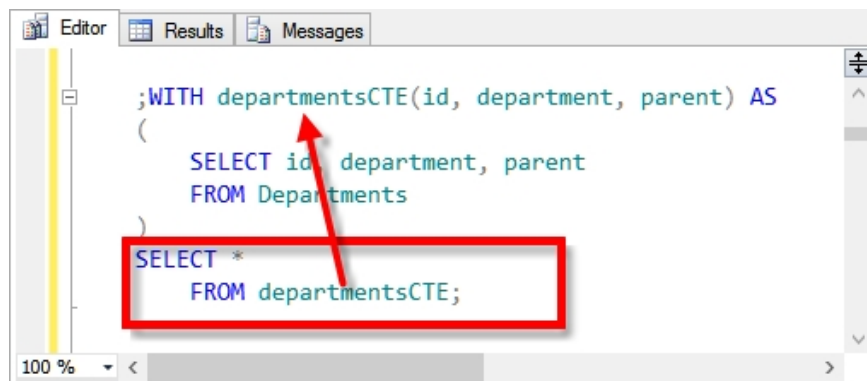


Following the AS statement is an open parenthesis ( the CTE query, and then a closing parenthesis ). Here the CTE query is just selecting 3 columns from the departments table.



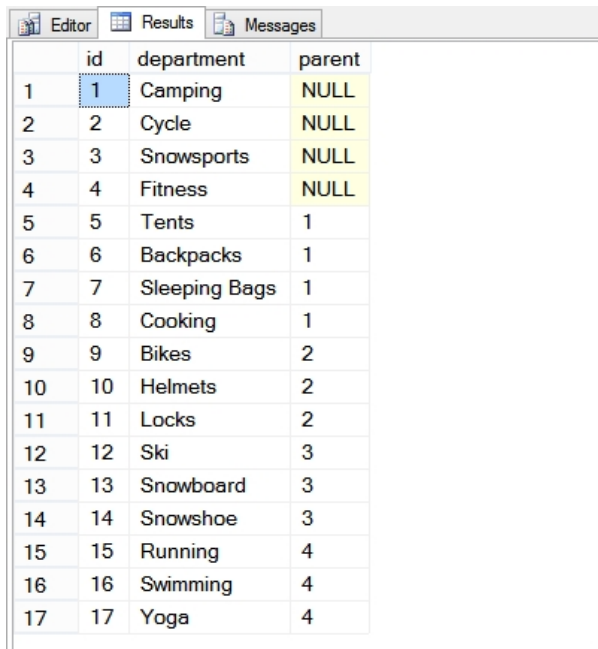
```
Editor Results Messages
;WITH departmentsCTE(id, department, parent) AS
(
    SELECT id, department, parent
    FROM Departments
)
SELECT *
FROM departmentsCTE;
```

And finally the query that calls or runs the CTE. Here we are just selecting all columns from the CTE that was defined above.



```
Editor Results Messages
;WITH departmentsCTE(id, department, parent) AS
(
    SELECT id, department, parent
    FROM Departments
)
SELECT *
FROM departmentsCTE;
```

Put it all together and you get the following output:



	id	department	parent
1	1	Camping	NULL
2	2	Cycle	NULL
3	3	Snowsports	NULL
4	4	Fitness	NULL
5	5	Tents	1
6	6	Backpacks	1
7	7	Sleeping Bags	1
8	8	Cooking	1
9	9	Bikes	2
10	10	Helmets	2
11	11	Locks	2
12	12	Ski	3
13	13	Snowboard	3
14	14	Snowshoe	3
15	15	Running	4
16	16	Swimming	4
17	17	Yoga	4

At this point it is not much more interesting than the derived table example, or even the simple select statement. This has been intended as just an overview or introduction to the anatomy of a SQL Server Common Table Expression. Tomorrows post will start showing some interesting things you can do now that you understand the simple CTE.

## Related Links:

- [My Book on Common Table Expressions](#)

- [cte demo sample database](#)

## Common Table Expressions Book

If you enjoyed this posting, and want to learn more about common table expressions, please take a look at my book on CTE's at Amazon.com. The book is titled Common Table Expressions - Joes 2 Pros® - A CTE Tutorial on Performance, Stored Procedures, Recursion, Nesting and the use of Multiple CTEs.