# Introduction to Common Table Expressions

Have you ever wanted to create a recursive query, but didn't see how to do it. With the Common Table Expressions session you will learn everything needed to start using CTE's for recursive queries, as temporary views, and to use the result set multiple times in the same query. Learn how simplify query syntax using CTE's. One of the most overlooked features of SQL Server is the CTE which not only simplifies the query, but gives you the ability to do things that would otherwise be impossible (or at least very challenging) with SQL Server. The class is designed for people who haven’t used CTE’s before, or for those who want to learn the basics of CTEs including data paging. This session pairs well with the Advanced Common Table Expressions session.

## Memory Tables

* Virtual Tables
* Views
* Temporary Named Result Sets
* Common Table Expressions

## What is a Common Table Expression?

* Overview of a CTE
* Writing Your First CTE
  + WITH Syntax
  + Optional Columns
  + Query Definition
  + Calling or Running the CTE
  + Terminate Previous Statements
  + Scope
* Why Use a CTE
  + Simplifying Queries with a CTE
  + Recursive Queries
* Summary

## CTEs Instead of Derived Tables

* Overwhelming Derived Tables
  + Cleanup with a CTE
* Performance Considerations
* Reducing Mistakes
* Summary

## Data Paging with CTEs

* Data Paging
  + Data Paging CTE
* Compared to Offset and Fetch in SQL Server 2012
* Summary

## Multiple CTEs in a Query

* Adding a Second CTE
* Nesting CTEs
  + How Many Levels of Nesting
* Additional Breakdown of Derived Tables
* Theoretical Example
* Summary

# Advanced Common Table Expressions

You have been introduced to Common Table Expression, you understand the ;WITH syntax, but you want to know more. Learn how to recursive queries work with CTEs and how to display hierarchical data. Did you know that you can INSERT, UPDATE and DELETE data from CTEs. Some of the common use cases for CTEs will be covered including finding holes in patterns, finding and removing duplicate data, string parsing, and more. See how CTEs compare to SQL Server 2012 offset and fetch paging techniques. Get an in depth understanding of the performance behind a common table expression. Understand when the CTE is the right solution, and the wrong solution to use. Finally we will take a look at some classic recursive algorithms and how they can be implemented with CTEs. This session pairs well with the Introduction to Common Table Expressions session.

## Recursive CTEs

* What is Recursion?
* Recursive CTE Syntax
  + Anchor Query
  + Recursive Query
  + Recursive Rows
  + Anchor Query
  + Recursive Query
  + MAXRECURSION
* Bring it All Together
* Advanced Recursion
  + Using Multiple An chor Queries
  + Using Multiple Recursive Queries
* Summary

## Hierarchical CTEs

* Hierarchical Tree Path
* Hierarchical Formatting
* Hierarchy in Multiple Recursive Queries
* Summary

## Manipulating Data

* Deleting Data
  + Deleting with Multiple Base Tables
* Inserting Data
* Updating Data
  + Multiple Base Tables
* Update with No Base Tables
* Summary

## CTEs in Functions, Stored Procedures and Views

* Table Valued Functions
* Scalar Functions
* Stored Procedures
* Views
* Summary

## Common Use Cases

* Finding Holes in Patterns
  + The Numbers Table Alternative
  + Dates of the Year (More Holes)
* Finding and Removing Duplicates
* String Parsing with CTEs
  + Recursive String Parsing
  + Parsing a Query String into a Table
* Summary

## CTE Performance Considerations

* Performance Overview
  + Actual Execution Plan
  + Statistics IO
  + Statistics Time
* Non-Recursive Performance
  + Multiple References to a Single CTE in a Query
  + CTEs vs. Derived Tables
  + CTEs vs. Views
  + Multiple CTEs in a Query
  + Nested CTEs
* Recursive Performance
  + Deep Recursion Performance
* CTE Data Paging Performance vs. Offset and Fetch
* Summary

## CTEs for Geeks – Classic Recursive Algorithms

* Fibonacci
  + The Anchor Query
  + Recursion
  + Data Type Overflow
  + The Output
* Factorial
  + The Anchor Query
  + Recursion
  + Overflow
* Summary