Common Table Expressions (CTEs) are used to structure SQL queries by creating temporary result sets, often simplifying complex queries. Multiple CTEs can be defined and used together in a query by separating them with commas.

Here's an example of how you can use **multiple CTEs** in a SQL query:

Example

```
WITH
-- First CTE to get total sales by ProductID
CTE_TotalSales AS (
    SELECT
        ProductID,
        SUM(LineTotal) AS TotalSales
    FROM
        Sales.SalesOrderDetail
    GROUP BY
        ProductID
),
-- Second CTE to get product information
CTE_ProductInfo AS (
    SELECT
        ProductID,
        Name AS ProductName,
        ListPrice
    FROM
        Production.Product
),
-- Third CTE to calculate sales performance
CTE_SalesPerformance AS (
    SELECT
        t.ProductID,
        p.ProductName,
        t.TotalSales,
        p.ListPrice,
        t.TotalSales - p.ListPrice AS Profit
    FROM
        CTE_TotalSales t
    INNER JOIN
        CTE_ProductInfo p ON t.ProductID = p.ProductID
)
-- Final query using the CTEs
SELECT
    ProductID,
    ProductName,
    TotalSales,
    ListPrice,
    Profit
FROM
    CTE_SalesPerformance
ORDER BY
    Profit DESC;
```

Explanation:

1. **CTE_TotalSales**: This CTE calculates the total sales for each `ProductID` from the `SalesOrderDetail` table.
2. **CTE_ProductInfo**: This CTE retrieves product information (like `ProductName` and `ListPrice`) from the `Product` table.

3. **CTE_SalesPerformance**: This CTE joins the two previous CTEs to calculate the profit (TotalSales - ListPrice) for each product.
4. The **final query** selects the result set from the `CTE_SalesPerformance` CTE and orders it by profit.

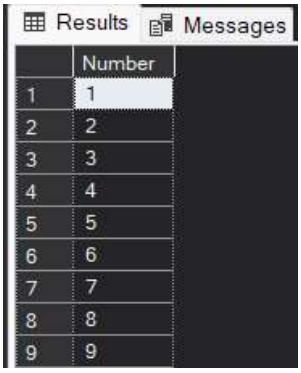
Recurssion

Example-1: Number Series

```
WITH NumberSeries AS (
    -- Anchor member: Start with number 1
    SELECT 1 AS Number

    UNION ALL

    -- Recursive member: Increment the number by 1 each time
    SELECT Number + 1
    FROM NumberSeries
    WHERE Number < 100 -- Limit the recursion to 100
)
SELECT Number
FROM NumberSeries;
```

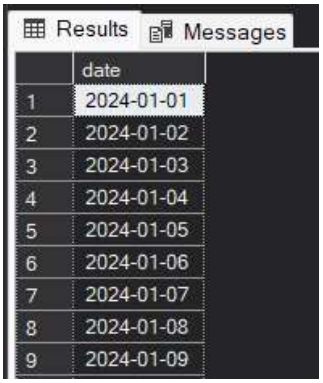


Example - 2: Calendar

```
WITH  DateSeries AS (
    -- Anchor member: Start with number 1
    SELECT CAST('2024-01-01' AS DATE) AS myDate

    UNION ALL

    -- Recursive member: Increment the Day by 1 each time
    SELECT DATEADD(Day,1,myDate)
    FROM DateSeries
    WHERE myDate < CAST('2025-01-01' AS DATE) -- Limit the recursion to 100
)
SELECT myDate
FROM dateSeries
OPTION (MAXRECURSION 366); -- Set maximum recursion for 1 year (adjust as needed)
```


Results Messages

	date
1	2024-01-01
2	2024-01-02
3	2024-01-03
4	2024-01-04
5	2024-01-05
6	2024-01-06
7	2024-01-07
8	2024-01-08
9	2024-01-09