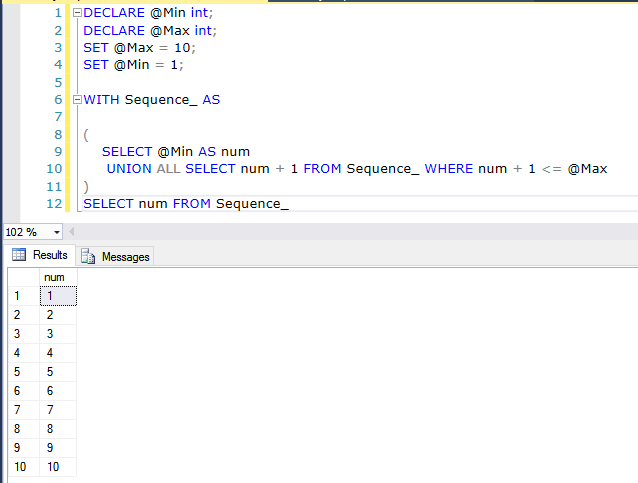
CTE is an abbreviation of “Common Table Expression.” CTE was introduced in SQL Server. It works as a temporary result set, which is defined within the execution scope of a single select, insert, update, delete statement. CTE is typically the result of complex sub queries. Similar to temporary tables, CTE doesn’t store as an object; the scope is limited to the current query. CTE improves readability and ease in maintenance of the complex queries and sub-queries. A CTE must be followed by a single SELECT, INSERT, UPDATE, or DELETE statement, which references some or all CTE columns. A CTE can also be specified in a CREATE VIEW statement as a part of defining SELECT statement of the view. Multiple CTE query definitions can be defined in a nonrecursive CTE. The definitions must be combined by one of these set operators: UNION ALL, UNION, INTERSECT or EXCEPT.

**Example**

1. **DECLARE** @**Min** **int**;
2. **DECLARE** @**Max** **int**;
3. **SET** @**Max** = 10;
4. **SET** @**Min** = 1;
5. **WITH** Sequence\_ **AS**(**SELECT** @**Min** **AS** num **UNION** ALL **SELECT** num + 1 **FROM** Sequence\_ **WHERE** num + 1 <= @**Max**)
6. **SELECT** num **FROM** Sequence\_

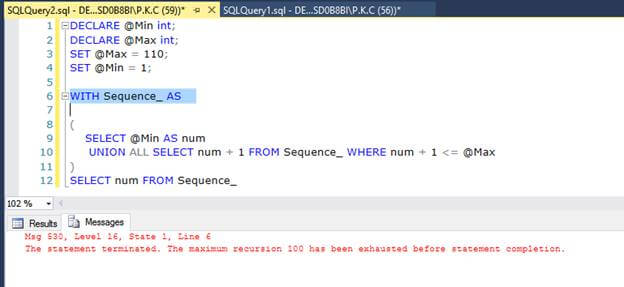
**Output  
  
**

In the example above, we write a recursive query, which returns a number series from 1 to 10 and the number of the recursion is 9. Let’s try to increase the value of @Max from 10 to 110.

**Query**

1. **DECLARE** @**Min** **int**;
2. **DECLARE** @**Max** **int**;
3. **SET** @**Max** = 110;
4. **SET** @**Min** = 1;
5. **WITH** Sequence\_ **AS**(**SELECT** @**Min** **AS** num **UNION** ALL **SELECT** num + 1 **FROM** Sequence\_ **WHERE** num + 1 <= @**Max**)
6. **SELECT** num **FROM** Sequence\_

**Output**



When you run the query given above, you will get an error of “The maximum recursion 100 has been exhausted before statement completion”. The reason behind this error is that by default, maximum number of recursion allowed for CTE is 100. If the numbers of recursion in your solution is more than 100, you will get this error.

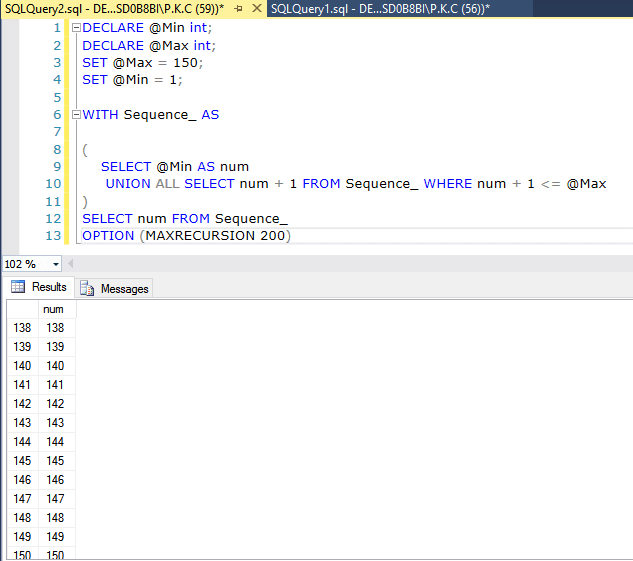
**How to fix this Error**

We can solve this query, using MAXRECURSION option. MAXRECURSION query hint specifies the maximum number of recursions allowed for a query. The number of recursions is a non-negative integer between 0 and 32,767.

**Example**

1. **DECLARE** @**Min** **int**;
2. **DECLARE** @**Max** **int**;
3. **SET** @**Max** = 150;
4. **SET** @**Min** = 1;
5. **WITH** Sequence\_ **AS**(**SELECT** @**Min** **AS** num **UNION** ALL **SELECT** num + 1 **FROM** Sequence\_ **WHERE** num + 1 <= @**Max**)
6. **SELECT** num **FROM** Sequence\_
7. **OPTION**(MAXRECURSION 200)

**Output**

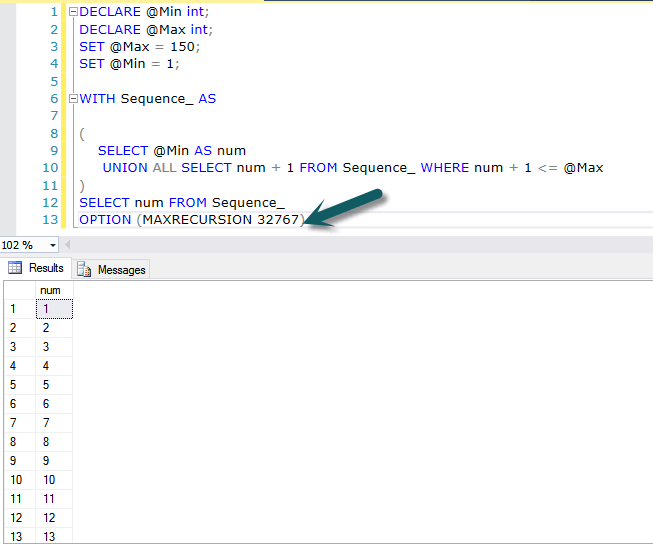


The result given above is OK, if you already know the maximum numbers of recursions for your query but what will be the possibility, if you don't have an idea about the maximum numbers of recursions? There are two solutions for this query.

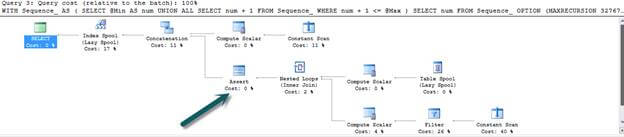
**Method 1**

Define maximum numbers of recursion to 32767

**Example**



The method given above allows you to perform the number of recursions between 1 to 32767 but this is not a good approach because when you examine the execution plan of this query, you will find a term assert.



In a SQL Server unit test, a Transact-SQL test script runs and returns a result. Sometimes, the results are returned as a result set. You can validate the results by using test conditions. For example, you can use a test condition to check how many rows were returned in a specific result set or to verify how long a particular test took to run. Thus, assert is used to examine a specified condition which exists. We can see the cost of this operation as 0 but we have another method, which is better than this, where we don’t need an assert.

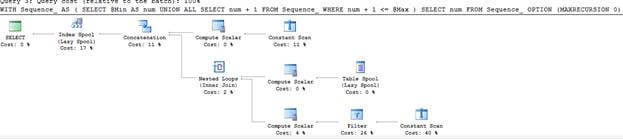
**Method 2**

Set MAXRECURSION to 0. When 0 is specified , no limit is applied and we also need not to assert any operation which is performed.

**Example**

1. **DECLARE** @**Min** **int**;
2. **DECLARE** @**Max** **int**;
3. **SET** @**Max** = 150;
4. **SET** @**Min** = 1;
5. **WITH** Sequence\_ **AS**(**SELECT** @**Min** **AS** num **UNION** ALL **SELECT** num + 1 **FROM** Sequence\_ **WHERE** num + 1 <= @**Max**)
6. **SELECT** num **FROM** Sequence\_
7. **OPTION**(MAXRECURSION 0)

**Execution Plan**

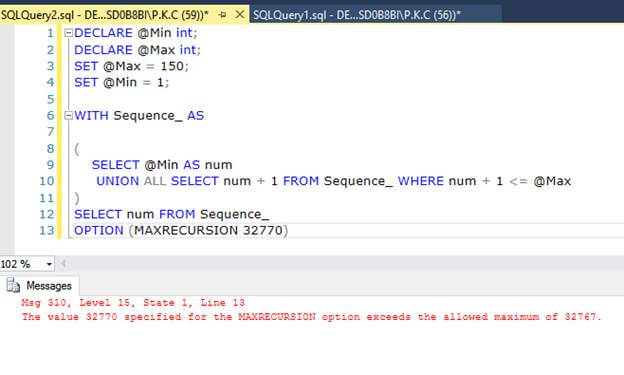


**How to perform recursion more than 32767.**

There is no way to perform a recursion more than 32767, if you increase the value of MAXRECURSION more than 32767, as you will get an error.

1. **DECLARE** @**Min** **int**;
2. **DECLARE** @**Max** **int**;
3. **SET** @**Max** = 150;
4. **SET** @**Min** = 1;
5. **WITH** Sequence\_ **AS**(**SELECT** @**Min** **AS** num **UNION** ALL **SELECT** num + 1 **FROM** Sequence\_ **WHERE** num + 1 <= @**Max**)
6. **SELECT** num **FROM** Sequence\_
7. **OPTION**(MAXRECURSION 32770)

**Output**

  
**Conclusion**

You can define the maximum number of recursions for CTE, using the MAXRECURSION option. Set the value of MAXRECURSION to 0, if you don’t know the exact numbers of recursions. I hope, you like this article. Thanks for reading this article.