

Logical Processing Order of SELECT

1. **FROM** table
2. **ON** join condition
3. **JOIN** table
4. **WHERE** clauses
5. **GROUP BY** columns
6. **WITH CUBE / WITH ROLLUP**
7. **HAVING** condition
8. **SELECT** columns
9. **DISTINCT**
10. **ORDER BY** columns
11. **TOP %** or number

The steps above show the logical processing order, or binding order, for a SELECT statement. This order determines when the objects defined in one step are made available to the clauses in subsequent steps.

CTEs

```

; WITH cteName ( columnList )
AS ( SELECT statement )
SELECT columns
FROM cteName
INNER JOIN table ON condition
    
```

Below, is a list of those statements and/or clauses that cannot be used in ANY CTE.

- > COMPUTE or COMPUTE BY
- > ORDER BY (except when a TOP clause is specified)
- > INTO
- > OPTION clause with query hints
- > FOR XML
- ✓ FOR BROWSE

Recursive CTEs

```

; WITH cteName ( columnList )
AS ( -- Anchor statement:
    SELECT columns FROM table...
    UNION ALL
    -- Recursion statement:
    SELECT columns FROM table...
    INNER JOIN cteName ON ...
)
SELECT columns
FROM cteName
    
```

Here are the statements and/or clauses that cannot be used in a recursive CTE:

- > SELECT DISTINCT
- > GROUP BY
- > HAVING
- > Scalar aggregation (meaning you can't use min or max)
- > TOP
- > LEFT, RIGHT, OUTER JOIN (INNER JOIN is allowed)


EXCEPT/INTERSECT

```

SELECT col1, col2 FROM Table1
EXCEPT
SELECT col3, col4 FROM Table2
    
```

```

SELECT col1, col2 FROM Table1
INTERSECT
SELECT col3, col4 FROM Table2
    
```

 INTERSECT, EXCEPT and UNION

MERGE

```

DECLARE @Changes
    TABLE(Change VARCHAR(20))
; MERGE INTO DestTable
USING
( SELECT from sourceTable
) AS Source ( columnList )
ON DestTable.ID = Source.ID
    
```

WHEN MATCHED THEN

Action on destination
-- E.g., UPDATE SET col1 = 1

WHEN NOT MATCHED BY TARGET|SOURCE

Action on destination
-- E.g., INSERT (col1) VALUES(1)

OUTPUT \$action INTO @Changes

```
SELECT * FROM @Changes
```

 MERGE Statement Generator

 Generate SQL MERGE statements with Table data

OVER and PARTITION BY

```
/* Aggregate functions include COUNT, MIN,  
MAX, AVG, ROW_COUNT(), etc. */
```

```
SELECT
```

```
  agg_func(col1) OVER(),  
  agg_func(col1)  
    OVER(PARTITION BY col2),
```

```
columns
```

```
FROM table...
```

OVER allows you to get aggregate information without using a GROUP BY. In other words, you can retrieve detail rows, and get aggregate data alongside it.

Using PARTITION BY the result set is broken into into partitions.

XML Trick: List of Details

```
/* Table2 holds detail rows for Table1; e.g., order details to order  
headers. */
```

```
SELECT columns,
```

```
  colname = STUFF(  
( SELECT ','  
  + Name  
  FROM Table2  
  WHERE Table1.ID = Table2.ID  
  ORDER BY Name  
  FOR XML PATH(''
```

```
), 1, 1, ''
```

```
FROM Table2
```

By **Tigersi** (renegrin)
[cheatography.com/renegrin/
tigersi.com](http://cheatography.com/renegrin/tigersi.com)

Published 29th January, 2016.
Last updated 21st March, 2017.
Page 1 of 2.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish
Yours!
<https://apollopad.com>