1. Create a Function and Call Another Function:

This process is known as nesting or function nesting.

Example:

sql

Copy code

```
CREATE FUNCTION MultiplyByTwo(@Value INT)
RETURNS INT
AS
BEGIN
   RETURN @Value * 2
END;
CREATE FUNCTION MultiplyByFour(@Value INT)
RETURNS INT
AS
BEGIN
   RETURN dbo.MultiplyByTwo(@Value) * 2
END;
```

Here, the MultiplyByFour function calls the MultiplyByTwo function.

2. Inspecting the Query's Execution Plan:

In SQL Server Management Studio (SSMS):

Open your query window.

Click on the "Include Actual Execution Plan" icon or press CTRL + M.

Execute your query.

Go to the 'Execution Plan' tab to see the graphical representation of how SQL Server executed (or will execute) your query.

3. Purpose of MAXDOP and Recompiling Keywords:

MAXDOP: Stands for Maximum Degree of Parallelism. It's used to limit the number of processors to be used for the execution of a query. For instance, OPTION (MAXDOP 2) would restrict the SQL query to use only 2 processors.

Recompiling: When you use the RECOMPILE option, SQL Server doesn't cache the execution plan for that query, meaning a new plan will be generated each time the query is executed. This can be useful for queries that are seldom executed or for queries where the data distribution changes frequently.

4. Build DDL Statements from an Existing Database Table:

Using SQL Server Management Studio (SSMS):

Connect to your database in SSMS.

Navigate to the desired table in the Object Explorer.

Right-click on the table > Script Table as > CREATE To > New Query Editor Window.

This will generate a DDL CREATE TABLE script for the selected table.

5. Update Data in a Table Using an Inner Join:

Example: Let's say we have two tables, Products and PriceUpdates, and we want to update the price of products using data from the PriceUpdates table:

sql

Copy code

```
UPDATE Products
SET Products.Price = PriceUpdates.NewPrice
FROM Products
INNER JOIN PriceUpdates ON Products.ProductID = PriceUpdates.ProductID;
```

6. Differentiate between Truncate, Delete, and Drop:

Truncate: Removes all rows from a table without logging individual row deletions. It's faster than the DELETE statement. However, it cannot delete specific rows, and triggers won't be executed.

sql

Copy code

```
TRUNCATE TABLE TableName;
```

Delete: Removes one or more rows from a table based on a condition. It logs individual row deletions, triggers will be executed, and can be slower than TRUNCATE.

sql

Copy code

DELETE FROM TableName WHERE Condition;

Drop: Removes an object from the database, such as a table. When you drop a table, all rows, indexes, triggers, constraints, and permission specifications for that table are removed.

sql

Copy code

DROP TABLE TableName;

In summary:

TRUNCATE is for quickly clearing all rows from a table.

DELETE is for selectively removing rows based on conditions and can be combined with WHERE clause.

DROP is for removing entire database objects.