1. What do you mean by Nested Stored Procedures?

SQL Stored Procedures that are called or executed from within another SQL stored procedure are called **Nested Stored procedures**. Each time a stored procedure calls another stored procedure (or executes managed code by referencing a common language runtime (CLR) routine, type, or aggregate,) the **nesting level** is incremented. When the **maximum of 32** is exceeded, the transaction is terminated.

Although the nesting limit is 32 levels, SQL Server has no limit on the number of stored procedures that can be invoked from a given stored procedure, provided that the subordinate stored procedures do not invoke other subordinate stored procedures and the maximum nesting level is never exceeded.

2. Analysis – Pros and Cons of Nested Stored Procedures

The following section provides the pros and cons of Nested SQL Stored procedures when compared with the Static SQL statements.

2.1 Benefits of Nested SQL Stored Procedures –

**Reuse:** The real benefit of Nested SQL Stored procedure is to reuse the already written functionality that than **save time** both in testing and in development.

**Simplicity:** With Nested approach the SQL code is simpler and modularized.
2.2 Drawbacks of Nested Stored Procedures –

Unable to view all Dependencies: Only one level of nested Stored Procedures in the Dependencies list can be viewed from using SQL Server Management Studio. Custom code needs to be written to view all dependencies.

Transaction Handling: With nested approach there is no easy way to turn the transaction handling on or off without removing or commenting out lines of code.

3. Best Practices while using Nested SQL

The following are the best practices for using Nested SQL.

1. Using Transactions

   Transactions group sets of related database calls in a single, logical unit-of-work. Thus managing transactions ensures that unexpected errors do not corrupt the database. For MS SQL Server applications transactions can be managed in the following ways:

   a. Application Level: For applications written against a SQL Server 2005 database, write transaction logic in C#, using the System.Data.SqlClient.SqlTransaction class that wraps calls to SQL Server statements or stored procedures.

   b. Microsoft Transaction Server: Utilizing MTS (Microsoft Transaction Server) to support distributed transactions. When these approaches are used, writing T-SQL code to manage transactions typically can be avoided.

   c. Stored Procedure level: Stored Procedural level, transaction management makes most sense when one might choose to implement a process that requires numerous, process-intensive queries and data manipulation statements as a database stored procedure or a set of procedures. [2] and [3] provides the details for stored procedure level transaction management. Note: Transactions can be nested and @@TRANCOUNT system function can be used to detect the level.

2. Error Handling

   a. Using RETURN or OUTPUT Parameter: Nesting stored procedures means you have stored procedures that call stored procedures; each stored procedure may or may not have a transaction. To trap non-fatal errors in a called stored procedure, the called procedure must have some way to communicate back to the calling procedure that an error has occurred. To do this, pass a value back
via the RETURN statement, or use an OUTPUT parameter. Either way works,
but once method is adopted, all the procedures in the system must use it.

b. **Error Exit Path Check:** In the procedure's error exit path, test whether this
procedure began a transaction. If it did, then the procedure issues a ROLLBACK, In
either case the procedure should RETURN a -1 to tell a calling procedure that it
should also exit through its error exit path. **A stored procedure transaction
should be rolled back at the same level at which it was started, so only
the calling procedure that starts a transaction should ever roll back.**

4. References

1. Nesting Stored Procedures
2. TIP: Nested Stored Procedure Calls with SQL Server Transactions
3. Handling SQL Server Errors in Nested Procedures
   [http://www.devx.com/codemag/Article/16120](http://www.devx.com/codemag/Article/16120)