**General Best Practices for Writing TSQL Code**

* Consistent naming convention should be used at all times
* Proper Error Handling Method should be used
* End the TSQL Statement with Semicolon (;)
* Wherever necessary avoid using Dynamic Queries
* Use Table variables for small datasets and Temporary tables for large datasets
* Do not use Select \* in the query, instead list only the required columns
* Ensure each table have primary key and there should be non-clustered index on all foreign keys.
* Avoid using functions in joins & where clause. Instead make a use of CTE and derived tables
* Make a use of Object\_id(), Instead of querying sysobjects table.
* While using orderby clause, avoid using order by ordinals
  1. Eg.. select empcode,name from employee order by 1,2
* Avoid using custom datatypes, instead make a use of SQL Data types.
* Avoid using Cross Datatypes for comparison in TSQL or Stored Procedures.
* Avoid using Functions in Join Condition or Where Clause

**Procedure Best Practices**

* Each Procedure Should Contain Version history as follows

**Release Procedure Date Who Comment**

**Version Version**

17.0 1.0 30-11-2015 Jitendra Mhaskar Procedure For

Zerochaos(DB) Insert

--------------------------------------------------------------------------------------------------------------------------------

* Use proper indentation for the statements in Stored Procedures. It will improve the readability.
* Write the proper comments between the logics. So the others can understand quickly.
* Write all the SQL Server keywords in the CAPS letter. For example SELECT, FROM and CREATE.
* Write the stored procedure name with full qualified names.  
    
  **CREATE PROCEDURE [dbo].EmployeeSalaryCalculation**
* Always try to declare the DECLARATION and initialization at the beginning of the stored procedure.
* It is not recommended to use more variables in the procedure. It will occupy more space in the memory.
* Do not write the stored procedure name beginning with sp\_. It is reserved for the system stored procedures in SQL Server and when the request comes to the SQL Server engine, it will be considered to be a system stored procedure and looks for it in the master database. After it understands that this is a user defined stored procedure, it requires a bit more response time. So name the procedure name with another prefix such a usp\_.
* Set the SET NOCOUNT ON option in the beginning of the stored procedure to avoid the unnecessary message like number of rows affected by the SQL Server.
* Always use SET ANSI NULLS ON, because SET ANSI NULLS OFF would not be allowed in future release.
* Try to avoid the temp table in the stored procedure. Stored procedures usually use a cached execution plan to increase the performance. When you use the temp table it will do the compilation every time.
* Do not use the select all columns (SELECT \*) option; use only specific columns to query the result.
* Try to avoid the cursor in the stored procedure. It will consume more memories. It will degrade the performance of the stored procedure. Try to use the table variable and WHILE loop statement to iterate the query result set.
* Set the default value to the parameter and always set the size of the variable to be equivalent to or more than the table field column length. For example Name (10) in the table, but if you give Name(25) in the procedure then you will get the run time error time "string truncated  error".
* Use the Try catch statement properly in the stored procedure to handle the errors in the runtime.
* Use the SELECT TOP 1 in the exists condition checking.
* Avoid Using Below Statement:  
    
  **SELECT @name=name FROM employees WHERE name like '%rob%'**  
  This will give the run time error when returns more than one result.  
    
  **SELECT TOP 1 @name=name FROM employees WHERE name like '%rob%'**  
  It is always recommended to use the TOP 1 in that case. The result may differ from what is expected.
* Avoid the nested IF statements and use the CASE statement. It will execute the matching part immediately.
* Dynamic Queries - Try to minimize the usage of dynamic queries. If you are using a dynamic query like:
* Use the ORDER BY and DISTINCT, TOP only when requires. The SQL Server engine will get the result first and it will do again the query execution for these operations.
* It is recommended to use a Table variable when the result set is small. It is always in the memory and when the limit exceeds it will be created as a table in the temp. But the temp table will be created on the temp database and that makes it slower.
* Use the proper indexing to the columns in the table. Do not create an index on the columns that are not used anywhere in the where clause. It will require an extra roundtrip to query the result.
* **Avoid GOTO stmt:** Do not use GOTO statements in your code as it is considered a bad programming practice (and in every other programming language).
* Avoid using Outer Apply wherever possible, Instead use Left Joins