DEVELOPMENT STANDARDS AND BEST PRACTICES

CREATE BY: TUSHAR CHAKRABORTY

DATE: 06.08.21

Follow the below STANDARDS AND BEST PRACTICES when developing SQL STORED PROCEDURE and FUNCTION

1) At top of SQL code

```
USE [Database Name]
GO
```

2) Next write below code

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER OFF
GO
SET NOCOUNT ON;
GO
```

3) Existence Check:

```
IF EXISTS (SELECT * FROM sysobjects WHERE TYPE = 'P' AND NAME = 'spGenerateDBDictionary')
  BEGIN
    DROP PROCEDURE spGenerateDBDictionary;
  END
GO
```

4) Signature:

```
/*      ============================================================
        Author:              TUSHAR CHAKRABORTY
        Create Date:         02/08/2021
        Description:         GENERATE DATA DICTIONARY
                             FROM SQL SERVER
        Modified By:
        Modified Date:
        Reason:
        Dependencies:        SYSOBJECTS
                             SYSCOLUMNS
                             SYSTYPES
                             SYSINDEXKEYS
                             FOREIGN_KEY_COLUMNS
                             EXTENDED_PROPERTIES
        ============================================================
*/
```

5) Create Stored Procedure: There will be always CREATE, no ALTER

```
CREATE PROCEDURE [dbo].[spGenerateDBDictionary]
```

6) Parameters

```
CREATE PROCEDURE [dbo].[spGenerateDBDictionary]
(
    @LastName NVARCHAR(50),
    @FirstName NVARCHAR(50)
)
AS
```

7) Optional Parameters

```
CREATE PROCEDURE [dbo].[spGenerateDBDictionary]
(
        @LastName NVARCHAR(50)      = NULL,
        @FirstName NVARCHAR(50)     = NULL
)
AS
```

8) Code should be within Begin End Block

```
CREATE PROCEDURE [dbo].[spGenerateDBDictionary]
(
    @LastName NVARCHAR(50),
    @FirstName NVARCHAR(50)
)
AS

BEGIN

       Code goes here…

END;
```

9) Declaration Section

```
--Declaration Section BEGIN
       DECLARE       @intA  INT,
                     @strB  VARCHAR(100);
--Declaration Section END
```

10) Executable Section

```
-- Executable Section BEGIN
              Code goes here…

-- Executable Section END
```

11) Exception Handling Section with TRY CATCH

```
BEGIN TRY
    Code goes here…
END TRY
-- Exception Handling Section BEGIN
BEGIN CATCH
    DECLARE @ErrorMessage NVARCHAR(4000);
    DECLARE @ErrorSeverity INT;
    DECLARE @ErrorState INT;

    SELECT
        @ErrorMessage = ERROR_MESSAGE(),
        @ErrorSeverity = ERROR_SEVERITY(),
        @ErrorState = ERROR_STATE();

    -- Use RAISERROR inside the CATCH block to return error
    -- information about the original error that caused
    -- execution to jump to the CATCH block.
    RAISERROR (
                    @ErrorMessage, -- Message text.
                    @ErrorSeverity, -- Severity.
                    @ErrorState -- State.
               );
END CATCH;
       -- Exception Handling Section END
```

12) Parameter Sniffing: To avoid DO NOT use parametric variable, use local variable.

```
CREATE PROCEDURE dbo.get_order_metrics_by_sales_person
(
       @sales_person_id INT
)
AS
BEGIN
       SET NOCOUNT ON;

       DECLARE @sales_person_id_local INT = @sales_person_id;

       SELECT
              SalesOrderHeader.SalesOrderID,
```

```
            SalesOrderHeader.DueDate,
            SalesOrderHeader.ShipDate
    FROM Sales.SalesOrderHeader
    WHERE SalesOrderHeader.SalesPersonID = @sales_person_id_local;
END;
```

13) All SQL Statements should be ended with ';'

14) To assign variable use SET, no SELECT

```
SET @sales_person_id_local = @sales_person_id;
```

15) Each Table should have Alias

16) Each Column should be prefixed by Table Alias

17) Fully qualified table names
Fully qualified table names in SQL Server consists of three parts. database name, schema name & the actual table name.

18) Proper indentation of code

```
SELECT
    A.NAME [TABLE],
    B.NAME [ATTRIBUTE],
    C.NAME [DATATYPE],
    B.ISNULLABLE [ALLOW NULLS?],
    CASE WHEN
        D.NAME IS NULL
        THEN 0
        ELSE 1
    END [PKEY?],
    CASE WHEN
        E.PARENT_OBJECT_ID IS NULL
        THEN 0
        ELSE 1
    END [FKEY?],
    CASE WHEN
        E.PARENT_OBJECT_ID IS NULL
        THEN '-'
        ELSE G.NAME
    END [REF TABLE],
    CASE WHEN
        H.VALUE IS NULL
        THEN '-'
        ELSE H.VALUE
    END [DESCRIPTION]
FROM SYSOBJECTS AS A
INNER JOIN SYSCOLUMNS AS B ON A.ID = B.ID
INNER JOIN SYSTYPES AS C ON B.XTYPE = C.XTYPE
```

19) DML Code should be in transaction block and transaction block should be minimum

20) DO NOT use IN and NOT IN.
Use EXISTS and NOT EXISTS
EXISTS checks occurrence only.

21) DO NOT use CURSOR, use WHIL.

22) DO NOT use DISTINCT, use GROUP BY.

23) If managing rows < 1000, use Table Variable, else use Temporal Table

24) Check Existence of Temporal Table before CREATE

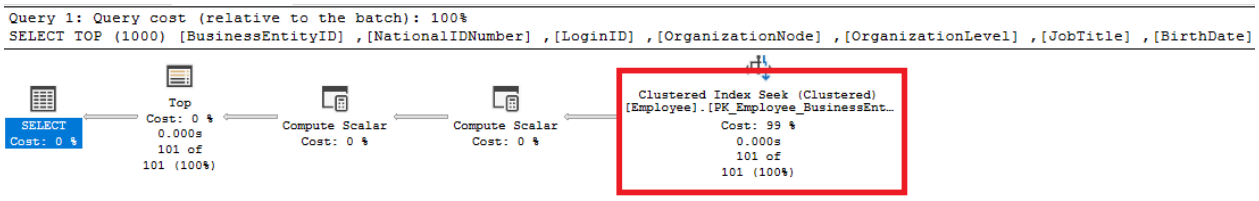
```
IF OBJECT_ID('TEMPDB..#TABLE','U') IS NOT NULL
       DROP TABLE #TABLE;

CREATE TABLE #TABLE (COL1 INT);
```

25) DROP Temporal Table after use.

26) All Table and Column names should be in CAPITAL only.

27) Check Execution Plan. SQL Optimizer should perform INDEX SEEK.



28) There should have proper comment before each execution

29) DO NOT use `SELECT * FROM`
Use `SELECT COL1, COL2 FROM`

30) Template:

```
USE [TUSHAR]
GO
IF EXISTS (SELECT * FROM sysobjects WHERE TYPE = 'P' AND NAME = 'spGenerateDBDictionary')
    BEGIN
        DROP  PROCEDURE spGenerateDBDictionary;
    END
GO

/****** Object:  StoredProcedure [dbo].[spGenerateDBDictionary]    Script Date: 06-08-2021
22:17:44 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER OFF
GO
/*     ========================================================
--     Author:       TUSHAR CHAKRABORTY
--     Create Date:  02/08/2021
--     Description:  GENERATE DATA DICTIONARY FROM SQL SERVER
--     Modified By:
--     Modified Date:
--     Reason:
--     Dependencies: SYSOBJECTS
                     SYSCOLUMNS
                     SYSTYPES
                     SYSINDEXKEYS
                     FOREIGN_KEY_COLUMNS
                     EXTENDED_PROPERTIES
   ========================================================
*/
CREATE  proc [dbo].[spGenerateDBDictionary]
(
       @sales_person_id INT
)
AS
BEGIN

       --Declaration Section BEGIN
              DECLARE       @intA  INT,
                            @strB  VARCHAR(100),
```

```
                    @sales_person_id_local INT = @sales_person_id;
--Declaration Section END

-- Executable Section BEGIN
BEGIN TRY
	BEGIN TRANSACTION;
	SELECT
		A.NAME [TABLE],
		B.NAME [ATTRIBUTE],
		C.NAME [DATATYPE],
		B.ISNULLABLE [ALLOW NULLS?],
		CASE WHEN
			D.NAME IS NULL
			THEN 0
			ELSE 1
		END [PKEY?],
		CASE WHEN
			E.PARENT_OBJECT_ID IS NULL
			THEN 0
			ELSE 1
		END [FKEY?],
		CASE WHEN
			E.PARENT_OBJECT_ID IS NULL
			THEN '-'
			ELSE G.NAME
		END [REF TABLE],
		CASE WHEN
			H.VALUE IS NULL
			THEN '-'
			ELSE H.VALUE
		END [DESCRIPTION]
	FROM SYSOBJECTS AS A
	INNER JOIN SYSCOLUMNS AS B ON A.ID = B.ID
	INNER JOIN SYSTYPES AS C ON B.XTYPE = C.XTYPE
	LEFT JOIN (SELECT  SO.ID,SC.COLID,SC.NAME
		FROM    SYSCOLUMNS SC
		INNER JOIN SYSOBJECTS SO ON SO.ID = SC.ID
		INNER JOIN SYSINDEXKEYS SI ON SO.ID = SI.ID
					AND SC.COLID = SI.COLID
		WHERE SI.INDID = 1) D ON A.ID = D.ID AND B.COLID = D.COLID
	LEFT JOIN SYS.FOREIGN_KEY_COLUMNS AS E
		ON A.ID = E.PARENT_OBJECT_ID AND B.COLID = E.PARENT_COLUMN_ID
	LEFT JOIN SYS.OBJECTS AS G ON E.REFERENCED_OBJECT_ID = G.OBJECT_ID
	LEFT JOIN SYS.EXTENDED_PROPERTIES AS H
		ON A.ID = H.MAJOR_ID AND B.COLID = H.MINOR_ID
	WHERE A.TYPE = 'U' ORDER BY A.NAME
	COMMIT TRANSACTION;
END TRY
-- Executable Section END
-- Exception Handling Section BEGIN
BEGIN CATCH
	DECLARE @ErrorMessage NVARCHAR(4000);
	DECLARE @ErrorSeverity INT;
	DECLARE @ErrorState INT;

	IF @@TRANCOUNT > 0
		ROLLBACK TRANSACTION;

	SELECT
		@ErrorMessage = ERROR_MESSAGE(),
		@ErrorSeverity = ERROR_SEVERITY(),
		@ErrorState = ERROR_STATE();

	-- Use RAISERROR inside the CATCH block to return error
	-- information about the original error that caused
	-- execution to jump to the CATCH block.
	RAISERROR (
			@ErrorMessage, -- Message text.
			@ErrorSeverity, -- Severity.
```

```
                                @ErrorState -- State.
                        );
        END CATCH;
        -- Exception Handling Section END
END;
```