**The Data Hub Programme**

**Technical Standards**

T-SQL Coding Standards

|  |  |  |
| --- | --- | --- |
| **Document Owner:** |  |  |

**Versions:**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Description | Author |
| 0.1 | 26/04/2018 | Initial Document | Suneetha Ganganakuntla |
|  |  |  |  |

**Reference Documents**

|  |  |  |
| --- | --- | --- |
| Document | Version | Date |
|  |  |  |

**Distribution & Reviewers list**

|  |  |  |
| --- | --- | --- |
| Name | Title | Date of Review |
| Paul Sinclair | Head of Data Management & Governance |  |
| Ellie Fraizer | IAAG Lead |  |
| Matthew Gascoyne | Lead Solution Architect |  |
| Damien O’Connor | Solution Architect |  |
| George Brennan | Data Architect |  |
| Andrew Roberts | Solution Architect |  |
| Brian Labrom | Data Architect |  |
| Suneetha Ganganakuntla | Data Architect |  |
|  |  |  |

**Approval List**

|  |  |  |
| --- | --- | --- |
| Name | Title | Date of Approval |

Contents

[T-SQL Coding Standards 4](#_Toc16169787)

[Background 4](#_Toc16169788)

[Conventions 4](#_Toc16169789)

[SQL Objects Naming Conventions Summary 5](#_Toc16169790)

[SQL Objects Naming Conventions Detail 6](#_Toc16169791)

[SQL Schema Naming 6](#_Toc16169792)

[SQL Table Naming 6](#_Toc16169793)

[SQL Column Naming 7](#_Toc16169794)

[SQL View Naming 7](#_Toc16169795)

[SQL Stored Procedure Naming 8](#_Toc16169796)

[SQL Function Naming 9](#_Toc16169797)

[SQL Trigger Naming 9](#_Toc16169798)

[SQL Primary Key/Unique Key/Default/Check Constraint/Index Naming 9](#_Toc16169799)

[SQL Foreign Keys Naming 10](#_Toc16169800)

[SQL User Defined Type Naming 10](#_Toc16169801)

[SQL Variables Naming 11](#_Toc16169802)

[SQL Parameter Naming 11](#_Toc16169803)

[SQL Best Practices 12](#_Toc16169804)

# T-SQL Coding Standards

# Background

This document covers the Transact Structured Query Language (T-SQL) guidelines and some industry standard best practices. The aim is to utilise these conventions, best practices, in all the development products for SQL Server. The document covers a wide range of objects used in the day to day SQL Development life cycle and provides standards that must be adhered to. The main purpose of defining these standards is consistency and utilisation of industry best practice that would lay the basic foundation for each data project within the department.

# Conventions

* Upper Case

This convention is used for SQL Keywords. Example SELECT, FROM, WHERE, ORDER BY

* Pascal Casing

This convention capitalises the very first character of each word. Example: RiskScore, UserName, FirstName

* Camel Casing

Camel case is the practice of writing compound words or phrases such that each word or abbreviation in the middle of the phrase begins with a capital letter, with no intervening spaces or punctuation. Example: riskScores, username, firstName

# SQL Objects Naming Conventions Summary

The table below enlists some basic SQL Objects along with the convention to be followed, illustrated with an example.

|  |  |  |  |
| --- | --- | --- | --- |
| Object Type | Convention | Prefix | Example |
| [Schema](#_SQL_Schema_Naming) | Pascal Case |  | Staging |
| [Table](#_SQL_Table_Naming) | Pascal Case |  | [Schema Name].Customer |
| [Column](#_SQL_Columns_Naming) | Pascal Case |  | FirstName, LastName, Salary |
| [View](#_SQL_View_Naming) | Pascal Case | vw | [Schema Name].vwCustomerDetails |
| [Stored Procedure](#_SQL_Stored_Procedure) | Pascal Case | sp | [Schema Name].spInsertCustomerDetails |
| [Function](#_SQL_Function_Naming) | Pascal Case | fn | [Schema Name].fnSplitString |
| [Trigger](#_SQL_Trigger_Naming) | Pascal Case | trg | trgRecordInsert |
| [Primary Key/Index](#_SQL_Primary_Key/Unique) | Pascal Case | PK\_ | PK\_CustomerKey |
| [Index](#_SQL_Primary_Key/Unique) | Pascal Case | IX\_ | IX\_AcademicYearCustomerType |
| [Unique Key/Index](#_SQL_Primary_Key/Unique) | Pascal Case | UK\_ | UK\_CustomerKey |
| [Foreign Key](#_SQL_Primary_Key/Unique) | Pascal Case | FK\_ | FK\_CustomerKey |
| [Default Constraint](#_SQL_Primary_Key/Unique) | Pascal Case | DF\_ | DF\_StartDate |
| [Check Constraint](#_SQL_Primary_Key/Unique) | Pascal Case | CK\_ | CK\_Age |
| [User Defined Type](#_SQL_User_Defined) | Pascal Case | udt | udtOrderRecord |
| [Variables](#_SQL_Variables_Naming) | Pascal Case | @ | @CustomerId |
| [Parameter](#_SQL_Parameter_Naming) | Pascal Case | @ | @CustomerName |
| [SQL Keywords](#_SQL_Keywords) | Upper Case |  | SELECT, FROM, HAVING, GROUP BY |

# SQL Objects Naming Conventions Detail

## 

## SQL Schema Naming

* Name it so that it describes the purpose of Subject Area of the data in the schema
* Do not use SQL keywords as Schema names e.g. YEAR, VALUE, COUNT
* Use Pascal casing
* Do not use spaces, reserved words or special characters in the schema names

## SQL Table Naming

* Do not use SQL keywords as table names e.g. Year
* Do use Pascal casing
* Do not use abbreviations e.g. dbo.Sal, dbo.Cust
* Do not use underscores instead use Pascal casing
* Do not use spaces in the name e.g. dbo.[This is Customer Table]
* Create separate Schemas to create tables instead of always using the dbo schema

Example:

USE [RATDataPublic]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dim].[Collection](

[CollectionCode] [int] IDENTITY(1,1) NOT NULL,

[CollectionDescription] [varchar](255) NULL,

[FailureThreshold] [decimal](6, 2) NULL,

[IsActive] [bit] NULL,

[CreatedOn] [date] NULL,

[CreatedBy] [varchar](255) NULL,

[ModifiedOn] [date] NULL,

[ModifiedBy] [varchar](255) NULL,

[Remarks] [varchar](255) NULL,

[OverrideGroup] [varchar](50) NULL,

CONSTRAINT [PK\_Collection] PRIMARY KEY CLUSTERED (

[CollectionCode] ASC

) WITH (

PAD\_INDEX = OFF,

STATISTICS\_NORECOMPUTE = OFF,

IGNORE\_DUP\_KEY = OFF,

ALLOW\_ROW\_LOCKS = ON,

ALLOW\_PAGE\_LOCKS = ON

) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

## 

## SQL Column Naming

* Do use Pascal casing
* Do not use spaces in the name
* Do not use abbreviations
* Do not use under scores and special characters
* Do not use SQL Keywords e.g. Year, Count

Example:

CREATE TABLE [dim].[AppliedCollection](

[ProviderGroupCode] [int] NOT NULL,

[CollectionCode] [int] NOT NULL,

[IsActive] [bit] NOT NULL,

[CreatedOn] [date] NULL,

[CreatedBy] [varchar](255) NULL,

[ModifiedOn] [date] NULL,

[ModifiedBy] [varchar](255) NULL,

[Remarks] [varchar](255) NULL,

CONSTRAINT [PK\_AppliedCollections\_1] PRIMARY KEY CLUSTERED

(

[ProviderGroupCode] ASC,

[CollectionCode] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

## SQL View Naming

* Use Pascal casing
* Do not use abbreviations e.g. dbo.vwSalDet, dbo.vwCustDtl
* Do not use underscores instead use Pascal casing
* Do not use spaces in the name
* Include comment block describing the view

Example:

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

-- Script Name: [dim].[vwSponsorTrackerRating]

-- Description: Sponsor Tracker Rating View

-- Author: Raja Kamboj

-- Creation Date: 10/02/2015

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

-- Change History

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

--S.No. Date Author Change Description

--0.1 10/02/2015 RK Initial draft

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

CREATE VIEW [dim].[vwSponsorTrackerRating]

AS

SELECT

[SponsorTrackerRatingCode]

,[SponsorTrackerRatingDescription]

,[IsActive]

,[OrderBy]

FROM

[dim].[SponsorTrackerRating]

## SQL Stored Procedure Naming

* Use “sp” as a lowercase prefix for every stored procedure name
* Use Pascal casing after the sp prefix e.g. Customer.spPostCodeUpdate
* Create a name with a structure such as <Schema>.<Object><Action> e.g. Customer.spCustomerSelect, Customer.spPostCodeDelete
* Do not use underscores instead use Pascal casing
* Do not use spaces in the name
* Do include comment block describing the Stored Procedure and include version and person who last changed the stored procedure
* Use code block from the template stored procedure.
* Use the name of the stored procedure to give us info about Update or Aggregate action at the end of stored procedure e.g spDailySalesAggregate
* Use below list of action names or abbreviations to identify each action
* Controller : A procedure to coordinate control flow and call child procedures
* Select: A procedure that presents data, typically used for presenting value lists.
* Update: A procedure to update values.
* Insert: A procedure used to insert data into a table
* Delete: A procedure used to delete data
* Transform: A procedure used to implement ETL logic
* Match/Lookup: A procedure used within the Matching processes
* Cleanse: A procedure used to cleanse a table or tables of problematic data
* Load: A procedure used to load a final table
* Aggregate: A procedure to create aggregated levels of data

Example:

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

-- Script Name: Customer.spGetOffsetLookup

-- Description: Procedure to get offset lookup data

-- Author: Raja Kamboj

-- Creation Date: 17/04/2015

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

-- Change History

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

--S.No. Date Author Change Description

--0.1 17/04/2015 RK Initial Draft

--0.2 02/06/2015 RK Changed the table name (dbo to dim)

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

CREATE Procedure [Customer].[spGetOffsetLookup]

(

@IndicatorCode INT, @CurrentYear INT

)

AS

BEGIN

SET NOCOUNT ON;

SELECT

CollectionCode,

IndicatorCode,

YearOffset,

OffsetValue,

(@CurrentYear + YearOffset) AS OffsetYear

FROM

dim.OffsetLookup oLook

WHERE

oLook.IndicatorCode = @IndicatorCode

END

## 

## SQL Function Naming

* Do use Pascal casing
* Do use abbreviations e.g. Customer.fnCustomerLookup, Customer.fnProviderSelect
* Do use underscores instead use Pascal casing
* Do use spaces in the name
* Do include comment block describing the function

Example:

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

-- Script Name: dbo.fnGetLookupValue

-- Description: Function to return lookup value for the given indicator

-- Author: Raja Kamboj

-- Creation Date: 10/11/2015

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

-- Change History

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

--S.No. Date Author Change Description

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

CREATE FUNCTION [dbo].[fnGetLookupValue]

(

@IndicatorCode INT,

@LookupCode DECIMAL(18,6)

)

RETURNS VARCHAR(255)

AS

BEGIN

DECLARE @LookupValue VARCHAR(255)

SELECT

@LookupValue = mLook.LookupValue

FROM

dim.MasterLookup mLook

WHERE

mLook.IndicatorCode = @IndicatorCode

AND mLook.LookupCode = @LookupCode

RETURN @LookupValue

END

## SQL Trigger Naming

Do not use triggers unless they are really required and have been discussed with the technical leads

## SQL Primary Key/Unique Key/Default/Check Constraint/Index Naming

* Do use Pascal casing
* Prefix primary indices with PK\_
* Prefix non-primary indices with IX\_
* Prefix unique non primary indices with UK\_
* Prefix default constraints with DF\_
* Prefix check constraints with CK\_
* Name indices after the columns that are used
* Do not use spaces in the name
* Prefix Column Store with CS\_
* Prefix Non clustered column store with NCS\_
* Prefix Clustered Index (Non Unique) with CI\_
* Prefix Foreign Key with FK\_

Example:

CONSTRAINT [PK\_Collection] PRIMARY KEY CLUSTERED

(

[CollectionCode] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

ALTER TABLE [dim].[Collection] ADD CONSTRAINT [DF\_Collection\_CreatedOn] DEFAULT (getdate()) FOR [CreatedOn]

GO

## SQL Foreign Keys Naming

* Do use Pascal casing
* Prefix foreign keys with FK\_
* Name the foreign keys including the parent table, Parent column and Child table and child column
* Do not use spaces in the name
* Do not use abbreviations

Example:

ALTER TABLE [dim].[AppliedCollection] WITH CHECK ADD CONSTRAINT [FK\_Collection\_CollectionCode\_AppliedCollection\_CollectionCode] FOREIGN KEY([CollectionCode])

REFERENCES [dim].[Collection] ([CollectionCode])

GO

## SQL User Defined Type Naming

You can access user-defined type (UDT) functionality in Microsoft SQL Server from the Transact-SQL language by using regular query syntax. UDTs can be used in the definition of database objects, as variables in Transact-SQL batches, in functions and stored procedures, and as arguments in functions and stored procedures.

* Do use Pascal casing
* Do not use spaces in the name
* Do not use abbreviations in the name

Example:

USE [RATDataPublic]

GO

CREATE TYPE [dbo].[udtRATRiskScore] AS TABLE(

[ProviderCode] [int] NULL,

[CollectionCode] [int] NULL,

[IndicatorCode] [int] NULL,

[IsPublished] [bit] NULL,

[LoadEffectiveFrom] [datetime] NULL,

[DataApplicableFrom] [datetime] NULL,

[AcademicYear] [int] NULL,

[IndicatorValue] [decimal](18, 6) NULL,

[MajorRiskScore] [decimal](18, 4) NULL,

[MinorRiskScore] [decimal](18, 4) NULL,

[OperandField1Value] [decimal](18, 6) NULL,

[OperandField2Value] [decimal](18, 6) NULL,

[OperandField3Value] [decimal](18, 6) NULL

)

GO

## SQL Variables Naming

* Do use Pascal casing
* Do not use spaces in the name
* Do not use abbreviations
* Do not use under scores

Example:

DECLARE @IndicatorCode INT

DECLARE @AcademicYear INT

DECLARE @CurrentYear INT

## SQL Parameter Naming

* Do use Pascal casing
* Do not use spaces in the name
* Do not use abbreviations
* Do not use under scores

Example:

CREATE PROCEDURE [dbo].[sprPersistRATFactIndicatorOutput]

(

@RiskScore RATRiskScore READONLY, @IndicatorCode INT, @DataLoadedForYear INT

)

**Note:**

Do not use SQL key words as object names in SQL Server like Year, Value, Count…etc

Below is the Microsoft link for list of SQL server keywords

<https://docs.microsoft.com/en-us/sql/t-sql/language-elements/reserved-keywords-transact-sql?view=sql-server-2017>

# SQL Best Practices

Below are the list best practices we should follow while coding in T-SQL:

* Optimize queries using the tools provided by SQL Server
* Do not use SELECT \* and always select the column names instead of \*
* Avoid unnecessary use of temporary tables
* Use 'Derived tables' or CTE (Common Table Expressions) wherever possible, as they

perform better

* Use SET NOCOUNT ON at the beginning of stored procedures
* Do not use cursors or application loops to do inserts. Instead, use INSERT INTO
* Fully qualify tables and column names in JOINs
* Fully qualify all stored procedure and table references in stored procedures.
* Do not use the RECOMPILE option for stored procedures.
* Place all DECLARE statements before any other code in the procedure.
* Do not use column numbers in the ORDER BY clause.
* Do not use GOTO.
* Check the global variable @@ERROR immediately after executing a data manipulation statement (like INSERT/UPDATE/DELETE), so that you can roll back the transaction if an error occurs. Or use TRY/CATCH
* Always use a column list in your INSERT statements this helps avoid problems when the table structure changes (like adding or dropping a column).
* Do keep code logic simple in each stored procedure. Split the code into multiple procedures if need be