SQL SCRIPT WRITING

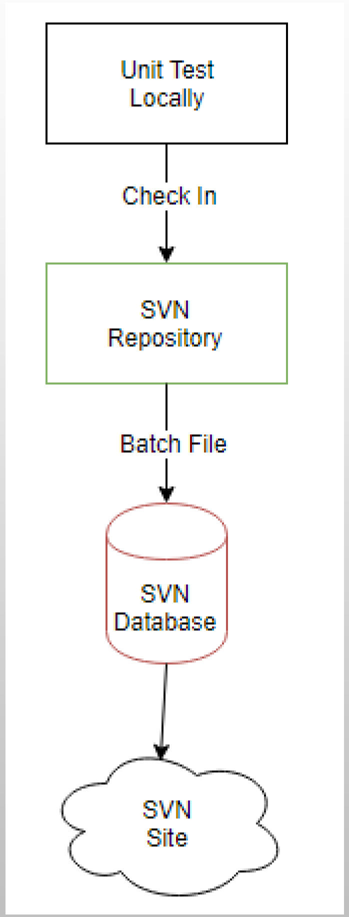
DB GURUS INTERNAL TRAINING

DB GURUS

1. PROCESS

- You check your script into the SVN repository with the next available number e.g. **1234.sql**
- An automatic process runs your script every hour, on the hour
- It is then tested by QA
- Once stable we deploy it to the UAT or live databases

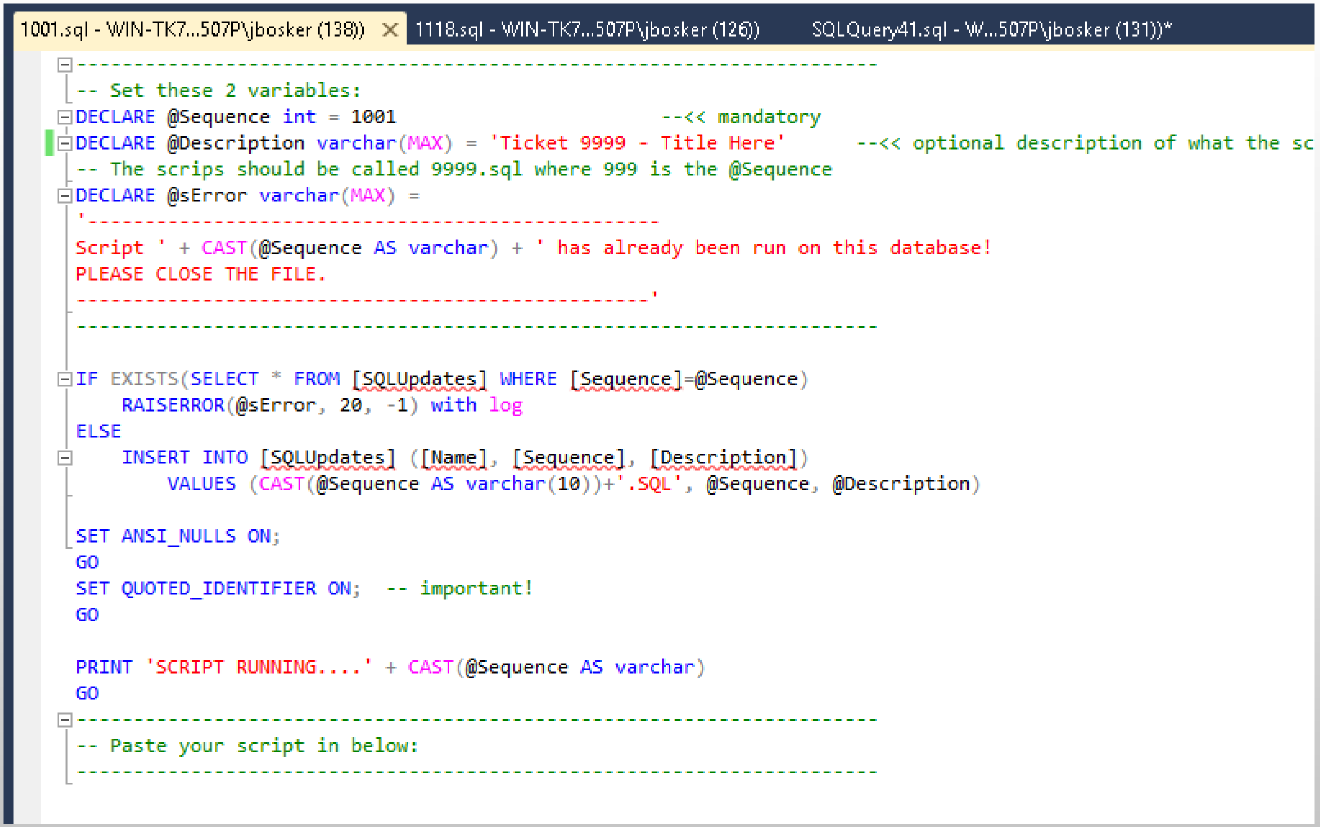
Note: You cannot update a script.. You must create a new script to fix any issues with an earlier script



DB GURUS

2. HEADER

1. Copy the latest version of **1001.sql** and use that as your template
2. Set the @Sequence to the next available number.
3. Set the @Description to the Ticket Number and Ticket Description
4. Paste your script in at the bottom
5. Make sure you leave the SET commands in there ☺


1001.sql - WIN-TK7...507P\jbosker (138))
1118.sql - WIN-TK7...507P\jbosker (126))
SQLQuery41.sql - W...507P\jbosker (131))*

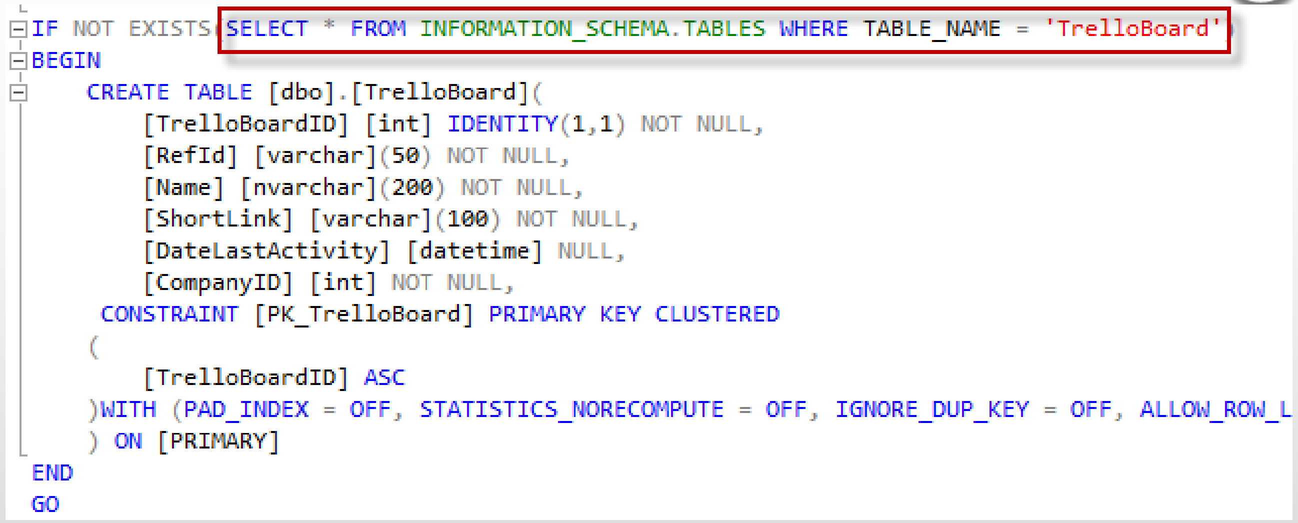
```sql
----------------------------------------------------------------
-- Set these 2 variables:
DECLARE @Sequence int = 1001                       --<< mandatory
DECLARE @Description varchar(MAX) = 'Ticket 9999 - Title Here'      --<< optional description of what the sc
-- The scrips should be called 9999.sql where 999 is the @Sequence
DECLARE @sError varchar(MAX) =
'---------------------------------------------------
Script ' + CAST(@Sequence AS varchar) + ' has already been run on this database!
PLEASE CLOSE THE FILE.
---------------------------------------------------'
----------------------------------------------------------------


IF EXISTS(SELECT * FROM [SQLUpdates] WHERE [Sequence]=@Sequence)
    RAISERROR(@sError, 20, -1) with log
ELSE
    INSERT INTO [SQLUpdates] ([Name], [Sequence], [Description])
        VALUES (CAST(@Sequence AS varchar(10))+'.SQL', @Sequence, @Description)

SET ANSI_NULLS ON;
GO
SET QUOTED_IDENTIFIER ON;  -- important!
GO

PRINT 'SCRIPT RUNNING....' + CAST(@Sequence AS varchar)
GO
----------------------------------------------------------------
-- Paste your script in below:
----------------------------------------------------------------
```

This script has been designed to ensure that scripts are run in the correct order and are not run multiple times. If run a 2nd time they end with an error.

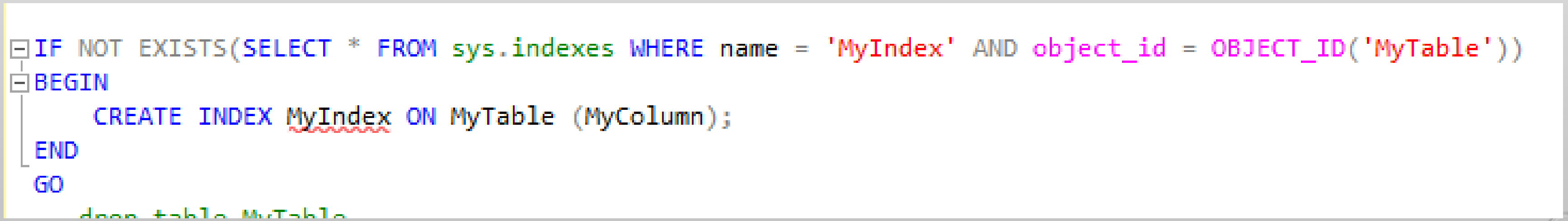
3. DB CHANGES

When we create a TABLE, COLUMN, CONSTRAINT, INDEX or anything we should check it does not exist first.

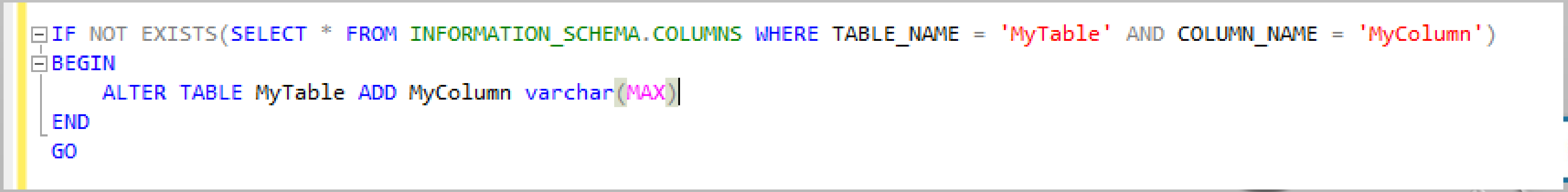
Use the INFORMATION_SCHEMA wherever possible, and the sys objects when not.



```
IF NOT EXISTS(SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'TrelloBoard')
BEGIN
    CREATE TABLE [dbo].[TrelloBoard](
        [TrelloBoardID] [int] IDENTITY(1,1) NOT NULL,
        [RefId] [varchar](50) NOT NULL,
        [Name] [nvarchar](200) NOT NULL,
        [ShortLink] [varchar](100) NOT NULL,
        [DateLastActivity] [datetime] NULL,
        [CompanyID] [int] NOT NULL,
     CONSTRAINT [PK_TrelloBoard] PRIMARY KEY CLUSTERED
    (
        [TrelloBoardID] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_L
    ) ON [PRIMARY]
END
GO
```



```
IF NOT EXISTS(SELECT * FROM sys.indexes WHERE name = 'MyIndex' AND object_id = OBJECT_ID('MyTable'))
BEGIN
    CREATE INDEX MyIndex ON MyTable (MyColumn);
END
GO
  drop table MyTable
```

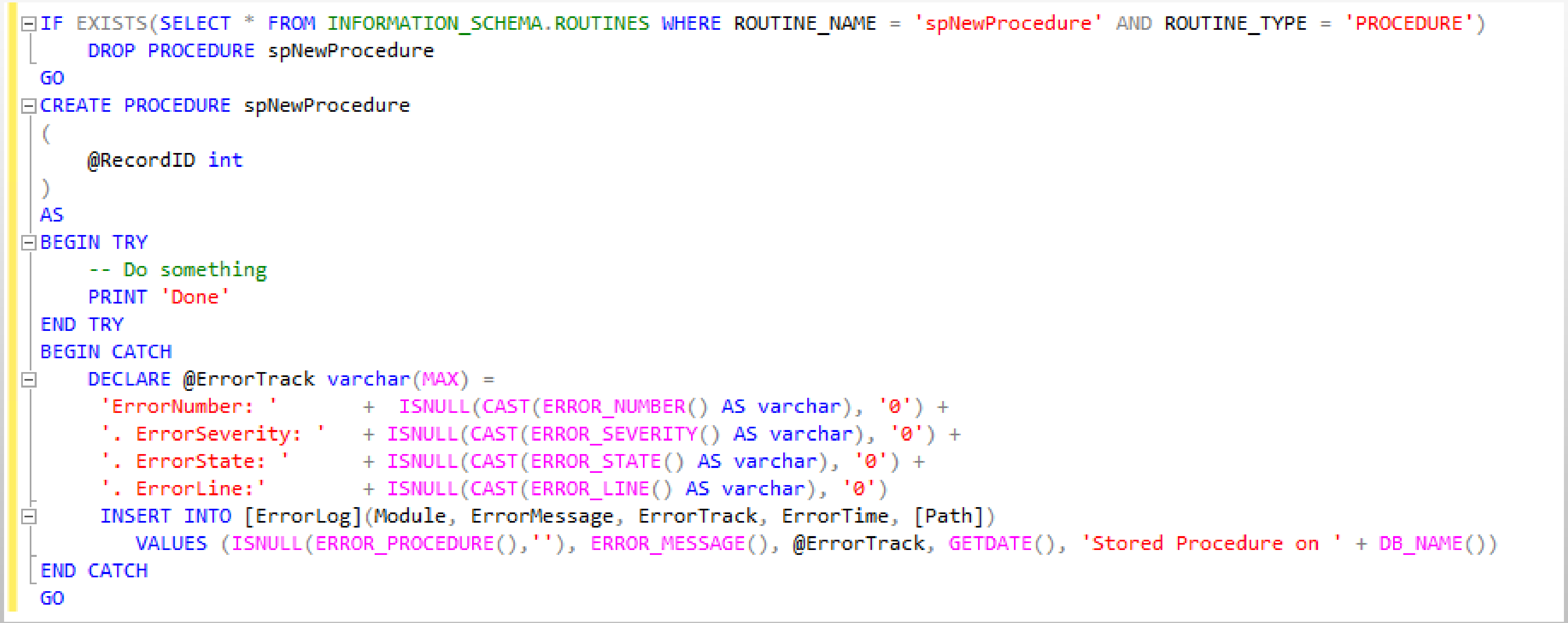


```
IF NOT EXISTS(SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'MyTable' AND COLUMN_NAME = 'MyColumn')
BEGIN
    ALTER TABLE MyTable ADD MyColumn varchar(MAX)
END
GO
```

6. ROUTINES

When creating or editing PROCEDURES of FUNCTIONS check and drop them first.

Then CREATE them. This ensures it will be the latest version:



```sql
IF EXISTS(SELECT * FROM INFORMATION_SCHEMA.ROUTINES WHERE ROUTINE_NAME = 'spNewProcedure' AND ROUTINE_TYPE = 'PROCEDURE')
    DROP PROCEDURE spNewProcedure
GO
CREATE PROCEDURE spNewProcedure
(
    @RecordID int
)
AS
BEGIN TRY
    -- Do something
    PRINT 'Done'
END TRY
BEGIN CATCH
    DECLARE @ErrorTrack varchar(MAX) =
     'ErrorNumber: '        +  ISNULL(CAST(ERROR_NUMBER() AS varchar), '0') +
     '. ErrorSeverity: '    + ISNULL(CAST(ERROR_SEVERITY() AS varchar), '0') +
     '. ErrorState: '       + ISNULL(CAST(ERROR_STATE() AS varchar), '0') +
     '. ErrorLine:'         + ISNULL(CAST(ERROR_LINE() AS varchar), '0')
     INSERT INTO [ErrorLog](Module, ErrorMessage, ErrorTrack, ErrorTime, [Path])
        VALUES (ISNULL(ERROR_PROCEDURE(),''), ERROR_MESSAGE(), @ErrorTrack, GETDATE(), 'Stored Procedure on ' + DB_NAME())
END CATCH
GO
```

Note that we now add TRY.. CATCH to any SP that we edit if it does not have one.

7. CASE

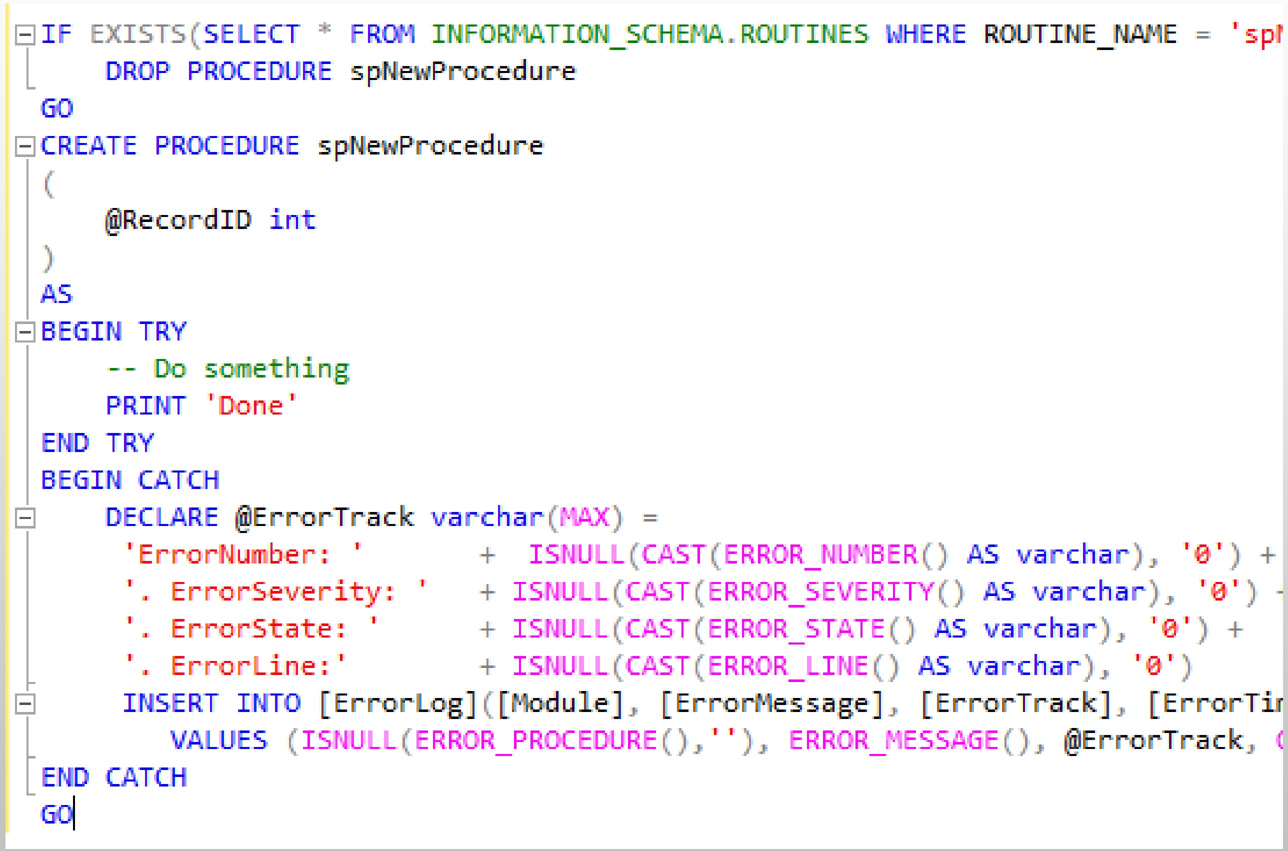
These is no right and wrong to it but this is what we have decided on:

Commands and SQL functions are in caps: e.g. SELECT, INSERT, UPDATE, DELETE, UPPER() and so on.

Variable names are in mixed case e.g. @Counter or @nInterval

Types are in lower case e.g. int, bit, varchar and decimal

Field and table names are in Mixed case with square brackets e.g. [Record].[RecordID], [SystemName]



```
IF EXISTS(SELECT * FROM INFORMATION_SCHEMA.ROUTINES WHERE ROUTINE_NAME = 'spN
    DROP PROCEDURE spNewProcedure
GO
CREATE PROCEDURE spNewProcedure
(
    @RecordID int
)
AS
BEGIN TRY
    -- Do something
    PRINT 'Done'
END TRY
BEGIN CATCH
    DECLARE @ErrorTrack varchar(MAX) =
     'ErrorNumber: '        +  ISNULL(CAST(ERROR_NUMBER() AS varchar), '0') +
     '. ErrorSeverity: '    + ISNULL(CAST(ERROR_SEVERITY() AS varchar), '0') -
     '. ErrorState: '       + ISNULL(CAST(ERROR_STATE() AS varchar), '0') +
     '. ErrorLine:'         + ISNULL(CAST(ERROR_LINE() AS varchar), '0')
     INSERT INTO [ErrorLog]([Module], [ErrorMessage], [ErrorTrack], [ErrorTi
        VALUES (ISNULL(ERROR_PROCEDURE(),''), ERROR_MESSAGE(), @ErrorTrack, (
END CATCH
GO
```

8. INDENTS

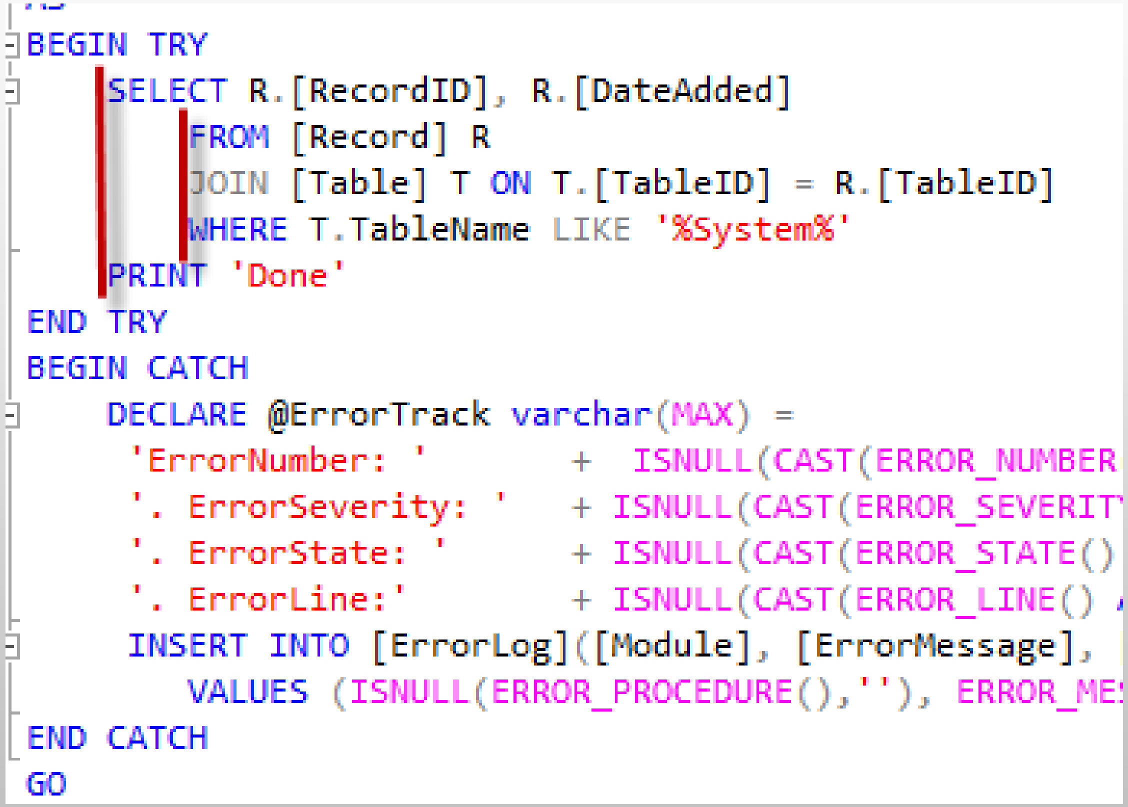
Again there is no right and wrong but consistency is good so please follow the examples:

Indent after a BEFORE

Indent follow up lines

FROM, JOIN, WHERE etc. starts a new (indented) line

Also note preferred way to use aliases and JOINs



```
BEGIN TRY
    SELECT R.[RecordID], R.[DateAdded]
        FROM [Record] R
        JOIN [Table] T ON T.[TableID] = R.[TableID]
        WHERE T.TableName LIKE '%System%'
    PRINT 'Done'
END TRY
BEGIN CATCH
    DECLARE @ErrorTrack varchar(MAX) =
     'ErrorNumber: '        +  ISNULL(CAST(ERROR_NUMBER
     '. ErrorSeverity: '    + ISNULL(CAST(ERROR_SEVERIT
     '. ErrorState: '       + ISNULL(CAST(ERROR_STATE()
     '. ErrorLine:'         + ISNULL(CAST(ERROR_LINE()
     INSERT INTO [ErrorLog]([Module], [ErrorMessage],
        VALUES (ISNULL(ERROR_PROCEDURE(),''), ERROR_ME
END CATCH
GO
```

DB GURUS