(T22)檢查如果 Object 存在(CheckIfObjectExist)
CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc
==========
(T22)檢查如果 Object 存在(CheckIfObjectExist)
==========
1. Create Sample Data
2. SYSOBJECTS_SYS.TABLES_INFORMATION_SCHEMA.TABLES
---
3. Recreate
3.1. Create or ReCreate Database
3.2. Recreate Table
3.3. Recreate Clomn
3.4. Recreate View
3.5. Recreate Stored Procedure
3.6. Recreate Table value function
3.7. Recreate scalar value function
3.8. Recreate Data Manipulation Language (DML) Trigger
3.9. Recreate Database Level Data Defination Language (DDL) Trigger
3.10. Recreate Server Level Data Defination Language (DDL) Trigger
3.11. Recreate Table Value Type
3.12. Recreate SequenceObject
3.13. Recreate Local Temp Table
3.14. Recreate Global Temp Table
3.15. Recreate default constraint
3.16. Recreate Check constraint
3.17. Recreate foreign key constraint
---
4. Clean up
==========

1. Create Sample Data

```
--=============================================================================
--T022_01_Create Sample Data
--=============================================================================

--=============================================================================
--T022_01_01
--Drop View if it exists.
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwGamer' ) )
    BEGIN
        DROP VIEW vwGamer;
    END;
GO -- Run the previous command and begins new batch
--=============================================================================
--T022_01_02
--Drop Table if it exists.
--If Table exists then DROP Tables
--IF OBJECT_ID('Gamer') IS NOT NULL
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'Gamer' ) )
    BEGIN
```

```
        DROP TABLE Gamer;
    END;
GO -- Run the previous command and begins new batch
/*
1.
You may use
--IF OBJECT_ID('Gamer') IS NOT NULL
or
--IF ( EXISTS ( SELECT    *
--              FROM      INFORMATION_SCHEMA.TABLES
--              WHERE     TABLE_NAME = 'Gamer' ) )
to see the if the Table exists or not.
Programer should always use INFORMATION_SCHEMA rather than sys objects
Reference:
https://stackoverflow.com/questions/219434/query-to-list-all-stored-procedures
https://stackoverflow.com/questions/3653637/sql-server-should-i-use-information-schema-tables-over-sys-tables
*/
--=============================================================================
--T022_02_03
--Create Table
CREATE TABLE Gamer
(
  GamerID INT PRIMARY KEY
                  IDENTITY(1, 1)
                  NOT NULL ,
  [Name] NVARCHAR(100) NULL ,
  GameScore NVARCHAR(50) NULL ,
  RegisteredDateTime DATETIME NULL
)
GO -- Run the previous command and begins new batch
--=============================================================================
--T022_01_04
--Gamer Counter
--***** Changeable data rows
DECLARE @TotolGamerRows INT = 20;
DECLARE @GamerCount INT = 1;
-- @RandomGameScore
DECLARE @RandomGameScore INT;
DECLARE @RandomGameScore_Max INT = 100000;
DECLARE @RandomGameScore_Min INT = 1;
--@RandomRegisteredDateTime
--Reference: http://crodrigues.com/sql-server-generate-random-datetime-within-a-range/
DECLARE @RandomRegisteredDateTime DATETIME;
DECLARE @DateFrom DATETIME = '2012-01-01';
DECLARE @DateTo DATETIME = '2017-06-30';
DECLARE @DaysRandom INT= 0;
DECLARE @MillisRandom INT= 0;
WHILE ( @GamerCount <= @TotolGamerRows )
    BEGIN
          --1. @RandomGameScore
        SELECT  @RandomGameScore = FLOOR(RAND() * ( @RandomGameScore_Max
                                                    - @RandomGameScore_Min )
                                    + @RandomGameScore_Min);
          --2. @RandomRegisteredDateTime
          --get random number of days
        SELECT  @DaysRandom = DATEDIFF(DAY, @DateFrom, @DateTo);
        SELECT  @DaysRandom = ROUND(( ( @DaysRandom - 1 ) * RAND() ), 0);
          --get random millis
```

```
        SELECT  @MillisRandom = ROUND(( ( 99999999 ) * RAND() ), 0);
        SELECT  @RandomRegisteredDateTime = DATEADD(DAY, @DaysRandom,
                                                    @DateFrom);
        SELECT  @RandomRegisteredDateTime = DATEADD(MILLISECOND, @MillisRandom,
                                                    @RandomRegisteredDateTime);
        INSERT  INTO Gamer
        VALUES  ( ( 'Name ' + CONVERT(NVARCHAR, @GamerCount) ),
                  CONVERT(NVARCHAR, @RandomGameScore), @RandomRegisteredDateTime );
        PRINT @GamerCount;
        SET @GamerCount += 1;
    END;
GO -- Run the previous command and begins new batch
--================================================================================
--T022_01_05
CREATE VIEW vwGamer
AS
    SELECT  *
    FROM    Gamer
GO -- Run the prvious command and begins new batch
--================================================================================
--T022_01_05
SELECT  *
FROM    vwGamer;
SELECT  *
FROM    dbo.Gamer;
GO -- Run the previous command and begins new batch
```

	GamerID	Name	GameScore	RegisteredDateTime
1	1	Name 1	5837	2013-04-19 23:32:37.790
2	2	Name 2	93058	2014-04-14 11:23:28.590
3	3	Name 3	45629	2016-11-26 12:21:43.637
4	4	Name 4	29491	2016-09-23 07:19:27.710
5	5	Name 5	73668	2016-05-23 11:16:14.947
6	6	Name 6	93296	2013-03-04 05:40:37.200
7	7	Name 7	66995	2012-08-05 00:46:03.083
8	8	Name 8	84939	2014-10-22 19:23:25.380
9	9	Name 9	66743	2017-02-25 12:33:05.883
10	10	Name 10	30268	2013-03-16 02:51:28.630
11	11	Name 11	99964	2014-07-23 22:36:51.617
12	12	Name 12	67215	2013-11-04 15:27:45.477
13	13	Name 13	82947	2015-07-23 00:30:19.743
14	14	Name 14	47133	2012-07-15 13:23:57.343
15	15	Name 15	76245	2012-07-06 13:32:00.360
16	16	Name 16	27417	2017-04-29 21:02:12.077
17	17	Name 17	56326	2013-07-02 14:47:04.263
18	18	Name 18	46986	2015-05-28 07:33:44.390
19	19	Name 19	36607	2014-11-30 05:09:01.847
20	20	Name 20	3886	2015-01-26 00:10:16.273

==========

2. SYSOBJECTS_SYS.TABLES_INFORMATION_SCHEMA .TABLES

```
--============================================================================
--T022_02_SYSOBJECTS_SYS.TABLES_INFORMATION_SCHEMA.TABLES
--============================================================================

--============================================================================
--T022_02_01
--Get Table List - 1
SELECT  *
FROM    sys.sysobjects
WHERE   xtype = 'U';
```

	name	id	xtype	uid	info	status	base_schema_ver	replinfo	parent_obj	crdate	ftcatid	schema_ver	stats_schema_ver	type	userstat	sysstat	indexdel	refdate
1	Gamer	645577338	U	1	0	0	0	0	0	2017-11-11 12:47:44.483	0	0	0	U	1	3	0	2017-11-11 12:47:44

```
--xtype in sys.sysobjects
SELECT DISTINCT
        xtype
FROM    sys.sysobjects;
```

	xtype
1	V
2	SQ
3	PK
4	U
5	IT
6	S

```
/*
--SELECT DISTINCT
--        xtype
--FROM    sys.sysobjects;
xtype in sys.sysobjects indicate the type of system objects.
Reference:
http://msdn.microsoft.com/en-us/library/ms177596.aspx
IT - Internal table
P - Stored procedure
PK - PRIMARY KEY constraint
S - System table
SQ - Service queue
U - User table
V - View
*/
--============================================================================
--T022_02_02
--Get Table List - 2
SELECT  *
FROM    sys.tables;
```

	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date	modify_date	is_ms_shipped	is_published	is_schema_published	lob_data_space_id	file
1	Gamer	645577338	NULL	1	0	U	USER_TABLE	2017-11-11 12:47:44.483	2017-11-11 12:47:44.483	0	0	0	0	N

```
--============================================================================
--T022_02_03
--Get Table List and Views
SELECT  *
FROM    INFORMATION_SCHEMA.TABLES;
/*
Everything with prefix 'sys' means system object.
For security reason, we try not to use system object.
```

```
Thus, It is always better to use INFORMATION_SCHEMA object
rather than sys.sysobjects and sys.tables.
*/
```

	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE
1	Sample	dbo	Gamer	BASE TABLE
2	Sample	dbo	vwGamer	VIEW

==========

3. Recreate

```
--=============================================================================
--T022_03_Recreate
--=============================================================================
```

3.1. Create or ReCreate Database

```
--=============================================================================
--T022_03_01
--Create or ReCreate Database.
USE master;
-- be sure that you're not on the database you want to delete
GO -- Run the prvious command and begins new batch
IF ( EXISTS ( SELECT    [name] ,
                        database_id ,
                        create_date
              FROM      sys.databases
              WHERE     name = N'Sample' ) )
    BEGIN
        --forced to delete DATABASE Sample
        ALTER DATABASE [Sample] SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
        DROP DATABASE [Sample];
    END;
GO -- Run the previous command and begins new batch
CREATE DATABASE [Sample];
GO -- Run the previous command and begins new batch
USE [Sample];
GO -- Run the prvious command and begins new batch
/*
1.
--IF ( EXISTS ( SELECT    [name] ,
--                        database_id ,
--                        create_date
--              FROM      sys.databases
--              WHERE     name = N'Sample' ) )
If the Sample exist.
2.
Reference:
https://stackoverflow.com/questions/17095472/cannot-drop-database-because-it-is-currently-in-use-mvc
Error Message:
Cannot drop database "NewDatabaseName" because it is currently in use.
Solutons:
--ALTER DATABASE [Sample] SET SINGLE_USER WITH ROLLBACK IMMEDIATE
--DROP DATABASE [Sample];
put the database in single user mode which
will rollback all incomplete transactions and closes the connection to the database.
```

```
then drop the database.
*/
```

3.2. Recreate Table

```
--=============================================================================
--T022_03_02
--Recreate Table
--Drop Table if it exists.
--IF OBJECT_ID('Gamer') IS NOT NULL
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'Gamer' ) )
    BEGIN
        TRUNCATE TABLE Gamer;
        DROP TABLE Gamer;
    END;
GO -- Run the previous command and begins new batch
CREATE TABLE Gamer
(
  GamerID INT PRIMARY KEY
                  IDENTITY(1, 1)
                  NOT NULL ,
  [Name] NVARCHAR(100) NULL
)
GO -- Run the previous command and begins new batch
SELECT  *
FROM    Gamer;
GO -- Run the previous command and begins new batch
```

3.3. Recreate Clomn

```
--=============================================================================
--T022_03_03
--Recreate Clomn
--IF the Column not exist, Add the Column, otherwise Alter the column
--IF COL_LENGTH('Gamer', 'Email') IS NULL
IF NOT EXISTS ( SELECT  *
                FROM    INFORMATION_SCHEMA.COLUMNS
                WHERE   COLUMN_NAME = 'Email'
                        AND TABLE_NAME = 'Gamer'
                        AND TABLE_SCHEMA = 'dbo' )
    BEGIN
        ALTER TABLE dbo.Gamer
        ADD Email NVARCHAR(100);
        PRINT 'Email Column has been added.';
    END;
ELSE
    BEGIN
            --IF COL_LENGTH('Gamer', 'Email') IS NOT NULL
        ALTER TABLE dbo.Gamer
```

```
        ALTER COLUMN Email NVARCHAR(100);
        PRINT 'Email Column has been altered.';
    END;
GO -- Run the previous command and begins new batch
-------------------------------------------
SELECT  *
FROM    Gamer;
GO -- Run the previous command and begins new batch
-------------------------------------------
--IF the Column not exist, Add the Column, otherwise Alter the column
--IF COL_LENGTH('Gamer', 'GameScore') IS NULL
IF NOT EXISTS ( SELECT  *
                FROM    INFORMATION_SCHEMA.COLUMNS
                WHERE   COLUMN_NAME = 'GameScore'
                        AND TABLE_NAME = 'Gamer'
                        AND TABLE_SCHEMA = 'dbo' )
    BEGIN
        ALTER TABLE dbo.Gamer
        ADD GameScore INT;
        PRINT 'GameScore Column has been added.';
    END;
ELSE
    BEGIN
            --IF COL_LENGTH('Gamer', 'Email') IS NOT NULL
        ALTER TABLE dbo.Gamer
        ALTER COLUMN GameScore INT;
        PRINT 'GameScore Column has been altered.';
    END;
GO -- Run the previous command and begins new batch
-------------------------------------------
SELECT  *
FROM    Gamer;
GO -- Run the previous command and begins new batch
-------------------------------------------
/*
1.
IF the Column not exist, Add the Column, otherwise Alter the column
2.
When I create Gamer Table, I set GameScore datatype is NVarchar
If you want to alter the column type in SSMS.
Make sure you have following setting in your SSMS.
-->
Tools --> Options --> Designers --> Table and Database Designers -->
Un-slected
Prevent saving changes that require table re-creation
*/
-------------------------------------------
IF EXISTS ( SELECT  *
                FROM    INFORMATION_SCHEMA.COLUMNS
                WHERE   COLUMN_NAME = 'GameScore'
                        AND TABLE_NAME = 'Gamer'
```

```
                        AND TABLE_SCHEMA = 'dbo' )
    BEGIN
        ALTER TABLE dbo.Gamer
             DROP COLUMN GameScore;
    END;
GO -- Run the previous command and begins new batch
-----------------------------------------
SELECT  *
FROM    Gamer;
GO -- Run the previous command and begins new batch
```

3.4. Recreate View

```
--=============================================================================
--T022_03_04
--Recreate View
--Drop View if it exists.
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwGamer' ) )
    BEGIN
        DROP VIEW vwGamer;
    END;
GO -- Run the previous command and begins new batch
CREATE VIEW vwGamer
AS
    SELECT  *
    FROM    Gamer
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    vwGamer
GO -- Run the previous command and begins new batch
```

3.5. Recreate Stored Procedure

```
--=============================================================================
--T022_03_05
--Recreate Stored Procedure
--Drop Stored Procedure if it exists.
--IF OBJECT_ID('spSearchGamer') IS NOT NULL
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.ROUTINES
              WHERE     ROUTINE_TYPE = 'PROCEDURE'
                        AND LEFT(ROUTINE_NAME, 3) NOT IN ( 'sp_', 'xp_', 'ms_' )
                        AND SPECIFIC_NAME = 'spGetGamers' ) )
    BEGIN
        DROP PROCEDURE spGetGamers;
    END;
GO -- Run the previous command and begins new batch
```

```
CREATE PROCEDURE spGetGamers
AS
    BEGIN
        SELECT  *
             FROM    Gamer
    END;
GO -- Run the prvious command and begins new batch
EXEC spGetGamers
GO -- Run the prvious command and begins new batch
```

3.6. Recreate Table value function

```
--=============================================================================
--T022_03_06
--Recreate Table value function
--Drop Table value function if it exists.
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.ROUTINES
              WHERE     ROUTINE_TYPE = 'FUNCTION'
                        AND LEFT(ROUTINE_NAME, 2) NOT IN ( '@@' )
                        AND SPECIFIC_NAME = 'fnGamers' ) )
    BEGIN
        DROP FUNCTION fnGamers;
    END;
GO -- Run the previous command and begins new batch
CREATE FUNCTION fnGamers ( )
RETURNS TABLE
AS
RETURN
    ( SELECT    *
      FROM      Gamer
    );
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    fnGamers();
```

3.7. Recreate scalar value function

```
--=============================================================================
--T022_03_07
--Recreate scalar value function
/*
/// <summary>
/// Input a date, then return the string value of duration between that date to today.
/// E.g. 33 Years 5 Months 14 Days
/// </summary>
/// <param name="Date">The input date</param>
/// <returns>The string value of duration between that date to today </returns>
*/
--Drop scalar value function if it exists.
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.ROUTINES
```

```
                WHERE     ROUTINE_TYPE = 'FUNCTION'
                          AND LEFT(ROUTINE_NAME, 2) NOT IN ( '@@' )
                          AND SPECIFIC_NAME = 'fnDurationByDate' ) )
    BEGIN
        DROP FUNCTION fnDurationByDate;
    END;
GO -- Run the previous command and begins new batch
CREATE FUNCTION fnDurationByDate ( @Date DATETIME )
RETURNS NVARCHAR(50)
AS
    BEGIN
        DECLARE @tempdate DATETIME ,
            @years INT ,
            @months INT ,
            @days INT;
        SELECT  @tempdate = @Date;
             -- Caculate Years
        SELECT  @years = DATEDIFF(YEAR, @tempdate, GETDATE())
                - CASE WHEN ( MONTH(@Date) > MONTH(GETDATE()) )
                            OR ( MONTH(@Date) = MONTH(GETDATE())
                                 AND DAY(@Date) > DAY(GETDATE())
                               ) THEN 1
                       ELSE 0
                  END;
        SELECT  @tempdate = DATEADD(YEAR, @years, @tempdate);
             -- Caculate Months
        SELECT  @months = DATEDIFF(MONTH, @tempdate, GETDATE())
                - CASE WHEN DAY(@Date) > DAY(GETDATE()) THEN 1
                       ELSE 0
                  END;
        SELECT  @tempdate = DATEADD(MONTH, @months, @tempdate);
             -- Caculate Days
        SELECT  @days = DATEDIFF(DAY, @tempdate, GETDATE());
        DECLARE @Duration NVARCHAR(50);
        SET @Duration = CAST(@years AS NVARCHAR(4)) + ' Years '
            + CAST(@months AS NVARCHAR(2)) + ' Months '
            + CAST(@days AS NVARCHAR(2)) + ' Days';
        RETURN @Duration;
    END;
GO -- Run the prvious command and begins new batch
PRINT dbo.fnDurationByDate('1984/09/10');
```

3.8. Recreate Data Manipulation Language (DML) Trigger

```
--=============================================================================
```

```
--T022_03_08
--Recreate Data Manipulation Language (DML) Trigger
--Drop DML Trigger if it exists.
IF EXISTS ( SELECT  *
            FROM    sys.objects
            WHERE   [name] = N'trgGamerForInsert'
                    AND [type] = 'TR' )
    BEGIN
        DROP TRIGGER trgGamerForInsert;
    END;
GO -- Run the previous command and begins new batch
-------------------------------------------------
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgGamerForInsert' )
    BEGIN
        --DROP TRIGGER trgNoNewTables ON DATABASE;
             DROP TRIGGER trgGamerForInsert;
    END;
GO -- Run the previous command and begins new batch
-------------------------------------------------
CREATE TRIGGER trgGamerForInsert ON Gamer
    --AFTER INSERT
    FOR INSERT
AS
    BEGIN
        PRINT 'AFTER INSERT event fired';
    END;
GO -- Run the prvious command and begins new batch
/*
2.
There are 2 Types of Data Manipulation Language (DML) triggers
2.1.
After/For Trigger:
After/For Triggers fires after the INSERT/UPDATE/DELETE event happened.
After/For Trigger Syntax:
--CREATE TRIGGER {TriggerName} ON {TableName}
--{ After/For Insert | AFTER/For DELETE | AFTER/For UPDATE }
--AS
--    BEGIN
--        ...
--    END
2.2.
INSTEAD OF Trigger:
Syntax:
--CREATE TRIGGER {TriggerName} ON {TableName}
--{ INSTEAD OF Insert | INSTEAD OF DELETE | INSTEAD OF UPDATE }
--AS
--    BEGIN
--        ...
--    END
INSTEAD OF Triggers fires when the Table/View run INSERT/UPDATE/DELETE event,
instead of running the default behaviour, it will run the query in the trigger body.
INSTEAD OF triggers normally correct updating views that are based on multiple tables.
*/
```

3.9. Recreate Database Level Data Defination Language (DDL) Trigger

```
--=============================================================================
--T022_03_09
--Recreate Database Level Data Defination Language (DDL) Trigger
--Drop DATABASE level DDL Trigger if it exists.
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgCreateTable' )
    BEGIN
        DROP TRIGGER trgCreateTable ON DATABASE;
    END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trgCreateTable ON DATABASE
    FOR CREATE_TABLE
AS
    BEGIN
        PRINT 'CREATE_TABLE event fired';
    END;
GO -- Run the previous command and begins new batch
/*
Create DDL Database scope Triggers in SSMS
Database Name --> Programmability --> Database Triggers
Create DDL All Server scope Triggers in SSMS
Server Objects --> Triggers --> ...
*/
```

3.10. Recreate Server Level Data Defination Language (DDL) Trigger

```
--=============================================================================
--T022_03_10
--Recreate Server Level Data Defination Language (DDL) Trigger
--Drop Server level DDL Trigger if it exists.
IF EXISTS ( SELECT  *
            FROM    sys.server_triggers
            WHERE   name = 'trgCreateAlterDropTable' )
    BEGIN
        DROP TRIGGER trgCreateAlterDropTable ON ALL SERVER;
    END;
GO -- Run the previous command and begins new batch
CREATE TRIGGER trgCreateAlterDropTable ON ALL SERVER
    FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
AS
    BEGIN
        PRINT 'CREATE_TABLE, ALTER_TABLE, DROP_TABLE DDL server level Trigger';
    END;
GO -- Run the previous command and begins new batch
```

```
/*
Create DDL Database scope Triggers in SSMS
Database Name --> Programmability --> Database Triggers
Create DDL All Server scope Triggers in SSMS
Server Objects --> Triggers --> ...
*/
```

3.11. Recreate Table Value Type

```
--=============================================================================
--T022_03_11
--Recreate Table Value Type
--Drop Table Value Type if it exists.
IF EXISTS ( SELECT  *
            FROM    sys.types
            WHERE   is_table_type = 1
                    AND name = 'PersonType' )
    BEGIN
        DROP TYPE PersonType;
    END;
GO -- Run the previous command and begins new batch
CREATE TYPE PersonType AS TABLE
(
Id INT PRIMARY KEY,
[Name] NVARCHAR(100),
Gender NVARCHAR(10)
);
GO -- Run the previous command and begins new batch
/*
In SSMS
Database --> Programmability --> Types --> User-Defined Types
*/
```

3.12. Recreate SequenceObject

```
--=============================================================================
--T022_03_12
--Recreate SequenceObject
--Drop Table if it exists.
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'Gamer' ) )
    BEGIN
        TRUNCATE TABLE Gamer;
        DROP TABLE Gamer;
    END;
GO -- Run the previous command and begins new batch
CREATE TABLE Gamer
(
  GamerID INT PRIMARY KEY NOT NULL,
  [Name] NVARCHAR(100) NULL
)
```

```
GO -- Run the previous command and begins new batch
------------------------------------------------------------
--Drop SequenceObject if it exists.
IF ( EXISTS ( SELECT    *
              FROM      sys.sequences
              WHERE     name = 'SequenceObject' ) )
    BEGIN
        DROP SEQUENCE SequenceObject
    END;
GO -- Run the previous command and begins new batch
CREATE SEQUENCE [dbo].[SequenceObject]
AS INT
START WITH 1
INCREMENT BY 1
INSERT INTO Gamer VALUES
   (NEXT VALUE for [dbo].[SequenceObject], N'Name01')
INSERT INTO Gamer VALUES
   (NEXT VALUE for [dbo].[SequenceObject], N'Name02')
GO -- Run the previous command and begins new batch
SELECT  *
FROM    Gamer;
GO -- Run the previous command and begins new batch
```

3.13. Recreate Local Temp Table

```
--============================================================================
--T022_03_13
--Recreate Local Temp Table
--Drop Local Temp Table if it exists.
IF OBJECT_ID('tempdb..#Gamer') IS NOT NULL
    BEGIN
        TRUNCATE TABLE #Gamer;
        DROP TABLE #Gamer;
    END;
GO -- Run the previous command and begins new batch
CREATE TABLE #Gamer
(
  GamerID INT PRIMARY KEY NOT NULL,
  [Name] NVARCHAR(100) NULL
)
GO -- Run the previous command and begins new batch
SELECT  *
FROM    #Gamer;
GO -- Run the previous command and begins new batch
```

3.14. Recreate Global Temp Table

```
--============================================================================
--T022_03_14
```

```
--Recreate Global Temp Table
--Drop Global Temp Table if it exists.
IF OBJECT_ID('tempdb..##Gamer') IS NOT NULL
    BEGIN
        TRUNCATE TABLE ##Gamer;
        DROP TABLE ##Gamer;
    END;
GO -- Run the previous command and begins new batch
CREATE TABLE ##Gamer
(
  GamerID INT PRIMARY KEY NOT NULL,
  [Name] NVARCHAR(100) NULL
)
GO -- Run the previous command and begins new batch
SELECT  *
FROM    ##Gamer;
GO -- Run the previous command and begins new batch
```

3.15. Recreate default constraint

```
--=============================================================================
--T022_03_15
--Recreate default constraint
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'Gamer' ) )
    BEGIN
        TRUNCATE TABLE Gamer;
        DROP TABLE Gamer;
    END;
GO -- Run the previous command and begins new batch
CREATE TABLE Gamer
(
  GamerID INT PRIMARY KEY NOT NULL,
  [Name] NVARCHAR(100) NULL,
  --Age INT DEFAULT (1) NULL
  Age INT NULL
)
GO -- Run the previous command and begins new batch
-----------------------------------------
--Drop default constraint if it exists.
IF OBJECT_ID('DF_Gamer_Age', 'D') IS NOT NULL
    BEGIN
        ALTER TABLE Gamer
        DROP CONSTRAINT DF_Gamer_Age;
    END;
GO -- Run the prvious command and begins new batch
ALTER TABLE Gamer
```

```
ADD  CONSTRAINT DF_Gamer_Age
DEFAULT ((1)) FOR [Age];
GO -- Run the prvious command and begins new batch
-----------------------------------------
--Check if the default constraint exists
SELECT  *
FROM    sys.objects
WHERE   type_desc LIKE '%CONSTRAINT'
        AND OBJECT_NAME(object_id) = 'DF_Gamer_Age';
GO -- Run the prvious command and begins new batch
-----------------------------------------
--Drop default constraint if it exists.
IF OBJECT_ID('DF_Gamer_Age', 'D') IS NOT NULL
    BEGIN
        ALTER TABLE Gamer
        DROP CONSTRAINT DF_Gamer_Age;
    END;
GO -- Run the prvious command and begins new batch
ALTER TABLE Gamer
ADD  CONSTRAINT [DF_Gamer_Age]
DEFAULT (2) FOR [Age];
GO -- Run the prvious command and begins new batch
/*
Constraint Object Types:
C = CHECK constraint
D = DEFAULT (constraint or stand-alone)
F = FOREIGN KEY constraint
PK = PRIMARY KEY constraint
R = Rule (old-style, stand-alone)
UQ = UNIQUE constraint
*/
-----------------------------------------
--Check if the default constraint exists
SELECT  *
FROM    sys.objects
WHERE   type_desc LIKE '%CONSTRAINT'
        AND OBJECT_NAME(object_id) = 'DF_Gamer_Age';
GO -- Run the prvious command and begins new batch
```

3.16. Recreate Check constraint

```
--=============================================================================
--T022_03_16
--Recreate Check constraint
--https://stackoverflow.com/questions/2499332/how-to-check-if-a-constraint-exists-in-sql-server
--Drop Check constraint if it exists.
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLE_CONSTRAINTS
              WHERE     CONSTRAINT_NAME = 'CK_Gamer_Age' ) )
    BEGIN
        ALTER TABLE Gamer
```

```
        DROP CONSTRAINT CK_Gamer_Age;
    END;
GO -- Run the previous command and begins new batch
ALTER TABLE Gamer
ADD CONSTRAINT CK_Gamer_Age CHECK (Age > 0 AND Age < 150);
GO -- Run the prvious command and begins new batch
```

3.17. Recreate foreign key constraint

```
--=============================================================================
--T022_03_17
--Recreate foreign key constraint
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'Gamer' ) )
    BEGIN
        TRUNCATE TABLE Gamer;
        DROP TABLE Gamer;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'Gender' ) )
    BEGIN
        TRUNCATE TABLE Gender;
        DROP TABLE Gender;
    END;
GO -- Run the previous command and begins new batch
---------------------------------
CREATE TABLE Gender
(
  GenderID INT PRIMARY KEY NOT NULL,
  GenderName NVARCHAR(10) NULL
)
GO -- Run the previous command and begins new batch
---------------------------------
CREATE TABLE Gamer
(
  GamerID INT PRIMARY KEY
              NOT NULL ,
  [Name] NVARCHAR(100) NULL ,
  --GenderId INT FOREIGN KEY REFERENCES Gender ( GenderID ) NULL
  GenderID INT NULL
);
GO -- Run the previous command and begins new batch
---------------------------------
--Drop the foreign key constraint if it exists.
IF ( EXISTS ( SELECT    *
```

```
              FROM      INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS
              WHERE     CONSTRAINT_NAME = 'FK_Gender_Gamer' ) )
    BEGIN
        ALTER TABLE Gamer
        DROP CONSTRAINT FK_Gender_Gamer;
    END;
GO -- Run the previous command and begins new batch
--Create the foreign key constraint
ALTER TABLE Gamer ADD CONSTRAINT FK_Gender_Gamer
FOREIGN KEY (GenderID) REFERENCES Gender(GenderID)
ON DELETE NO ACTION;
GO -- Run the prvious command and begins new batch

==========
```

4. Clean up

```
--=============================================================================
--T022_04_Clean up
--=============================================================================
--Drop SequenceObject if it exists.
IF ( EXISTS ( SELECT    *
              FROM      sys.sequences
              WHERE     name = 'SequenceObject' ) )
    BEGIN
        DROP SEQUENCE SequenceObject
    END;
GO -- Run the previous command and begins new batch
-----------------------------------------------
--Drop Table Value Type if it exists.
IF EXISTS ( SELECT  *
            FROM    sys.types
            WHERE   is_table_type = 1
                    AND name = 'PersonType' )
    BEGIN
        DROP TYPE PersonType;
    END;
GO -- Run the previous command and begins new batch
-----------------------------------------------
--Drop Server level DDL Trigger if it exists.
IF EXISTS ( SELECT  *
            FROM    sys.server_triggers
            WHERE   name = 'trgCreateAlterDropTable' )
    BEGIN
        DROP TRIGGER trgCreateAlterDropTable ON ALL SERVER;
    END;
GO -- Run the previous command and begins new batch
-----------------------------------------------
--Drop DATABASE level DDL Trigger if it exists.
```

```
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgCreateTable' )
    BEGIN
        DROP TRIGGER trgCreateTable ON DATABASE;
    END;
GO -- Run the previous command and begins new batch
-------------------------------------------------
--Drop DML Trigger if it exists.
IF EXISTS ( SELECT  *
            FROM    sys.objects
            WHERE   [name] = N'trgGamerForInsert'
                    AND [type] = 'TR' )
    BEGIN
        DROP TRIGGER trgGamerForInsert;
    END;
GO -- Run the previous command and begins new batch
-------------------------------------------------
IF EXISTS ( SELECT  *
            FROM    sys.triggers
            WHERE   name = 'trgGamerForInsert' )
    BEGIN
        --DROP TRIGGER trgNoNewTables ON DATABASE;
             DROP TRIGGER trgGamerForInsert;
    END;
GO -- Run the previous command and begins new batch
-------------------------------------------------
--Drop scalar value function if it exists.
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.ROUTINES
              WHERE     ROUTINE_TYPE = 'FUNCTION'
                        AND LEFT(ROUTINE_NAME, 2) NOT IN ( '@@' )
                        AND SPECIFIC_NAME = 'fnDurationByDate' ) )
    BEGIN
        DROP FUNCTION fnDurationByDate;
    END;
GO -- Run the previous command and begins new batch
-------------------------------------------------
--Drop Table value function if it exists.
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.ROUTINES
              WHERE     ROUTINE_TYPE = 'FUNCTION'
                        AND LEFT(ROUTINE_NAME, 2) NOT IN ( '@@' )
                        AND SPECIFIC_NAME = 'fnGamers' ) )
    BEGIN
        DROP FUNCTION fnGamers;
```

```
    END;
GO -- Run the previous command and begins new batch
------------------------------------------------
--Drop Stored Procedure if it exists.
--IF OBJECT_ID('spSearchGamer') IS NOT NULL
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.ROUTINES
              WHERE     ROUTINE_TYPE = 'PROCEDURE'
                        AND LEFT(ROUTINE_NAME, 3) NOT IN ( 'sp_', 'xp_', 'ms_' )
                        AND SPECIFIC_NAME = 'spGetGamers' ) )
    BEGIN
        DROP PROCEDURE spGetGamers;
    END;
GO -- Run the previous command and begins new batch
------------------------------------------------
--Drop View if it exists.
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwGamer' ) )
    BEGIN
        DROP VIEW vwGamer;
    END;
GO -- Run the previous command and begins new batch
------------------------------------------------
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'Gamer' ) )
    BEGIN
        TRUNCATE TABLE Gamer;
        DROP TABLE Gamer;
    END;
GO -- Run the previous command and begins new batch
------------------------------------------------
--Drop Local Temp Table if it exists.
IF OBJECT_ID('tempdb..#Gamer') IS NOT NULL
    BEGIN
        TRUNCATE TABLE #Gamer;
        DROP TABLE #Gamer;
    END;
GO -- Run the previous command and begins new batch
------------------------------------------------
--Drop Global Temp Table if it exists.
IF OBJECT_ID('tempdb..##Gamer') IS NOT NULL
    BEGIN
        TRUNCATE TABLE ##Gamer;
        DROP TABLE ##Gamer;
    END;
```

```
GO -- Run the previous command and begins new batch
-------------------------------------------------
USE master;
-- be sure that you're not on the database you want to delete
GO -- Run the prvious command and begins new batch
IF ( EXISTS ( SELECT    [name] ,
                        database_id ,
                        create_date
              FROM      sys.databases
              WHERE     name = N'Sample' ) )
    BEGIN
        --forced to delete DATABASE Sample
        ALTER DATABASE [Sample] SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
        DROP DATABASE [Sample];
    END;
GO -- Run the previous command and begins new batch
```