(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

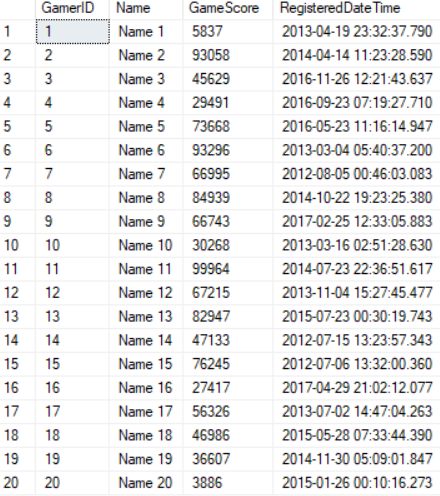
SELECT  \*

FROM    vwGamer;

SELECT  \*

FROM    dbo.Gamer;

GO -- Run the previous command and begins new batch



==================================================

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';



--xtype in sys.sysobjects

SELECT DISTINCT

        xtype

FROM    sys.sysobjects;

Table

Description automatically generated

/\*

--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;



--===================================================================================

--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.

\*/

Graphical user interface, text

Description automatically generated with medium confidence

==================================================

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