**The final document to be delivered will include the physical data model (the ‘code’) to create the database as well as data INSERT scripts of at least five (5) rows for each look-up table. Specific objects to be reflected in the code include the database, tables, data types, primary and foreign keys..**

**Common Code for creating tables, adding rows, etc:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IMT 543: Group 2: Movie Rating Database

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--CREATE DATABASE IMT543\_Proj\_A2;

USE IMT543\_Proj\_A2;

--Drop all tables if they exist

-- Drop Associative tables

DROP TABLE IF EXISTS tblREVIEW;

DROP TABLE IF EXISTS tblDESCRIPTION;

DROP TABLE IF EXISTS tblCREDIT;

DROP TABLE IF EXISTS tblNOMINATION;

-- Drop lookup tables

DROP TABLE IF EXISTS tblMOVIE;

DROP TABLE IF EXISTS tblLANGUAGE;

DROP TABLE IF EXISTS tblRATING;

DROP TABLE IF EXISTS tblVIEWER;

DROP TABLE IF EXISTS tblPROFESSIONAL;

DROP TABLE IF EXISTS tblAWARD;

DROP TABLE IF EXISTS tblGENRE;

DROP TABLE IF EXISTS tblROLE;

DROP TABLE IF EXISTS tblGENDER;

-- Create lookup tables

CREATE TABLE tblGENDER

( GenderID INT IDENTITY(1,1) PRIMARY KEY CLUSTERED,

Gender VARCHAR(20) NOT NULL UNIQUE,

)

CREATE TABLE tblRATING

( RatingID INT IDENTITY(1,1) PRIMARY KEY CLUSTERED,

RatingValue INT NOT NULL UNIQUE,

RatingName VARCHAR(30) NOT NULL UNIQUE,

CONSTRAINT Chk\_Invalid\_Rating CHECK (1 <= RatingValue AND RatingValue <= 5)

)

CREATE TABLE tblVIEWER

( ViewerID INT IDENTITY(1,1) PRIMARY KEY CLUSTERED,

ViewerFName VARCHAR(30) NOT NULL,

ViewerLName VARCHAR(30),

DateOfBirth DATE,

GenderID INT FOREIGN KEY REFERENCES tblGENDER(GenderID)

)

CREATE TABLE tblLANGUAGE

(

LanguageID INT IDENTITY(1,1) PRIMARY KEY CLUSTERED,

Language VARCHAR(30) NOT NULL UNIQUE,

)

CREATE TABLE tblMOVIE

(

MovieID INT IDENTITY(1,1) PRIMARY KEY CLUSTERED,

MovieName VARCHAR(100) NOT NULL,

YearOfRelease INT NOT NULL,

LanguageID INT FOREIGN KEY REFERENCES tblLANGUAGE(LanguageID)

)

CREATE TABLE tblAWARD

(

AwardID INT IDENTITY(1,1) PRIMARY KEY CLUSTERED,

AwardName VARCHAR(30) NOT NULL

)

CREATE TABLE tblGENRE

(

GenreID INT IDENTITY(1,1) PRIMARY KEY CLUSTERED,

GenreName VARCHAR(20) NOT NULL,

GenreDescription VARCHAR(300) NOT NULL

)

CREATE TABLE tblROLE

(

RoleID INT IDENTITY(1,1) PRIMARY KEY CLUSTERED,

RoleName VARCHAR(20) NOT NULL,

)

CREATE TABLE tblPROFESSIONAL

( ProfessionalID INT IDENTITY(1,1) PRIMARY KEY CLUSTERED,

ProfessionalFName VARCHAR(30) NOT NULL,

ProfessionalLName VARCHAR(30),

DateOfBirth DATE,

GenderID INT FOREIGN KEY REFERENCES tblGENDER(GenderID)

)

-- Create Associative Tables

CREATE TABLE tblREVIEW

(

ViewerID INT FOREIGN KEY REFERENCES tblVIEWER(ViewerID),

RatingID INT FOREIGN KEY REFERENCES tblRATING(RatingID),

MovieID INT FOREIGN KEY REFERENCES tblMOVIE(MovieID),

Body VARCHAR(500),

PRIMARY KEY(ViewerID, MovieID)

)

CREATE TABLE tblDESCRIPTION

(

GenreID INT FOREIGN KEY REFERENCES tblGENRE(GenreID),

MovieID INT FOREIGN KEY REFERENCES tblMOVIE(MovieID),

PRIMARY KEY(GenreID, MovieID)

)

CREATE TABLE tblCREDIT

(

MovieID INT FOREIGN KEY REFERENCES tblMOVIE(MovieID),

ProfessionalID INT FOREIGN KEY REFERENCES tblPROFESSIONAL(ProfessionalID),

RoleID INT FOREIGN KEY REFERENCES tblROLE(RoleID),

PRIMARY KEY(MovieID, ProfessionalID, RoleID)

)

CREATE TABLE tblNOMINATION

(

MovieID INT FOREIGN KEY REFERENCES tblMOVIE(MovieID),

ProfessionalID INT FOREIGN KEY REFERENCES tblPROFESSIONAL(ProfessionalID),

RoleID INT FOREIGN KEY REFERENCES tblROLE(RoleID),

AwardID INT FOREIGN KEY REFERENCES tblAWARD(AwardID),

YearOfNomination INT NOT NULL,

PRIMARY KEY(MovieID, ProfessionalID, RoleID, AwardID)

)

-- Insert rows into lookup tables

INSERT INTO tblGENDER(Gender) VALUES

('Male'),

('Female')

INSERT INTO tblRATING(RatingValue,RatingName) VALUES

(1, 'Very Bad'),

(2, 'Bad'),

(3, 'Average'),

(4, 'Good'),

(5, 'Very Good')

INSERT INTO tblLANGUAGE VALUES

('English'),

('French'),

('Italian'),

('Spanish'),

('Hindi'),

('Korean')

INSERT INTO tblROLE(RoleName) VALUES

('Actor'),

('Director'),

('Producer'),

('Writer'),

('Actress'),

('Cinematographer'),

('Music Director')

INSERT INTO tblAWARD(AwardName) VALUES

('Oscar'),

('Bafta'),

('Golden Globe'),

('Independent Spirit'),

('Seattle Film Critics'),

('Adi Critics'),

('Filmfare Awards')

INSERT INTO tblGENRE(GenreName, GenreDescription) VALUES

('Comedy','A comedy film is a genre of film in which the main emphasis is on humor'),

('Horror','A horror film is a film that seeks to elicit fear for entertainment purposes'),

('Romantic','Romance film can be defined as a genre wherein the plot revolves around the love between two or more protagonists.'),

('Action','Action film is a film genre in which the protagonist or protagonists are thrust into a series of events that typically include violence, extended fighting, physical feats and frantic chases'),

('Adventure','Adventure films are a genre of film that typically use their action scenes to display and explore exotic locations in an energetic way'),

('Drama','Drama Films are serious presentations or stories with settings or life situations that portray realistic characters in conflict with either themselves, others, or forces of nature'),

('Epic',' Epic films are a style of filmmaking with large scale, sweeping scope, and spectacle'),

('Mystery',' A mystery film is a genre of film that revolves around the solution of a problem or a crime.'),

('Thriller',' A thriller is a genre of literature, film, and television whose primary feature is that it induces strong feelings of excitement, anxiety, tension, suspense, fear, and other similar emotions in its readers or viewers'),

('Neo-noir',' A neo-noir film is a genre of film that revives the noir style, uses lowkey lighting, creates chiaroscuro dealing with existential dread, world weariness, paranoia,flashback plot style with anti-hero as protagonist')

-- Stored Procedures to populate tables with foreign keys (Nadia)

GO

CREATE OR ALTER PROCEDURE PopulateMovie

@MovName varchar(100),

@Yr INT,

@Lang varchar(50)

AS

DECLARE @L\_ID INT

SET @L\_ID = (SELECT LanguageID FROM tblLANGUAGE WHERE Language = @Lang)

-- error-handling --> we know the entire procedure will fail at the transaction if this is NULL

IF @L\_ID IS NULL

BEGIN

PRINT 'Error: Language does not exist!!'

RAISERROR ('@L\_ID cannot be NULL; statement is being terminated', 11,1)

RETURN

END

BEGIN TRAN G1

INSERT INTO tblMOVIE(MovieName,YearOfRelease,LanguageID)

VALUES (@MovName,@Yr,@L\_ID)

-- error-handling

IF @@ERROR <> 0

BEGIN

PRINT '@@ERROR is showing a number <> 0; transaction is being terminated'

ROLLBACK TRAN G1

END

ELSE

COMMIT TRAN G1

GO

EXEC PopulateMovie

@MovName='Pretty Woman',

@Yr='1990',

@Lang='English'

EXEC PopulateMovie

@MovName='Sweet Home Alabama',

@Yr='2002',

@Lang='English'

EXEC PopulateMovie

@MovName='Joker',

@Yr=2019,

@Lang='English'

EXEC PopulateMovie

@MovName='Titanic',

@Yr=1997,

@Lang='English'

EXEC PopulateMovie

@MovName='Todo Sobre Mi Madre',

@Yr=1999,

@Lang='Spanish'

EXEC PopulateMovie

@MovName='No Se Aceptan Devoluciones',

@Yr=2013,

@Lang='Spanish'

EXEC PopulateMovie

@MovName='Love Aaj Kal',

@Yr=2009,

@Lang='Hindi'

EXEC PopulateMovie

@MovName='Dil se',

@Yr=1999,

@Lang='Hindi'

EXEC PopulateMovie

@MovName='Divorce Italian Style',

@Yr=1961,

@Lang='Italian'

EXEC PopulateMovie

@MovName='La Dolce Vita',

@Yr=1961,

@Lang=Italian

EXEC PopulateMovie

@MovName='Freaky Friday',

@Yr=2003,

@Lang='English'

EXEC PopulateMovie

@MovName='Roma',

@Yr=2018,

@Lang='Spanish'

GO

-- Stored Procedure to populate tblVIEWER

CREATE OR ALTER PROCEDURE PopulateViewer

@FName VARCHAR(30),

@LName VARCHAR(30),

@DOB DATE,

@Gender VARCHAR(20)

AS

DECLARE @GenderID INT

SET @GenderID = (SELECT GenderID FROM tblGENDER WHERE Gender = @Gender)

IF @GenderID IS NULL

BEGIN

PRINT 'Error: Gender does not exist!!'

RAISERROR ('GenderID cannot be NULL; statement is being terminated', 11,1)

RETURN

END

INSERT INTO tblVIEWER(ViewerFName,ViewerLName,DateOfBirth,GenderID)

VALUES (@FName,@LName,@DOB,@GenderID)

GO

-- Exec procedure

EXEC PopulateViewer

@FName='Laura',

@LName  = 'Gonzalez',

@DOB='1990-12-01',

@Gender = 'Female'

EXEC PopulateViewer

@FName='Seema',

@LName  = 'Nair',

@DOB='1980-01-13',

@Gender = 'Female'

EXEC PopulateViewer

@FName='Tom',

@LName  = 'Sherbatsky',

@DOB='1994-09-23',

@Gender = 'Male'

EXEC PopulateViewer

@FName='Eddie',

@LName  = 'Ed',

@DOB='1980-07-30',

@Gender = 'Male'

EXEC PopulateViewer

@FName='Minty',

@LName  = 'Zelner',

@DOB='2000-01-03',

@Gender = 'Female'

EXEC PopulateViewer

@FName='Bonnie',

@LName  = 'Mandson',

@DOB='1994-01-04',

@Gender = 'Female'

EXEC PopulateViewer

@FName='Lebby',

@LName  = 'Parrish',

@DOB='1992-11-19',

@Gender = 'Male'

EXEC PopulateViewer

@FName='Son',

@LName  = 'Lu',

@DOB='1993-01-29',

@Gender = 'Male'

GO

-- Stored Procedure to populate tblPROFESSIONAL

CREATE OR ALTER PROCEDURE PopulateProfessional

@ProfessionalFName varchar(30),

@ProfessionalLName varchar(30),

@DateOfBirth date,

@GenderID varchar(20)

AS

DECLARE @GID INT

SET @GID = (SELECT GenderID FROM tblGENDER WHERE Gender = @GenderID)

-- error-handling --> we know the entire procedure will fail at the transaction if this is NULL

IF @GID IS NULL

BEGIN

PRINT 'Error: Gender does not exist!!'

RAISERROR ('@Gender cannot be NULL; statement is being terminated', 12,2)

RETURN

END

BEGIN TRAN G2

INSERT INTO [dbo].[tblPROFESSIONAL]

           ([ProfessionalFName]

           ,[ProfessionalLName]

           ,[DateOfBirth]

           ,[GenderID])

VALUES (@ProfessionalFName,@ProfessionalLName, @DateOfBirth,@GID)

-- error-handling

IF @@ERROR <> 0

BEGIN

PRINT '@@ERROR is showing a number <> 0; transaction is being terminated'

ROLLBACK TRAN G2

END

ELSE

COMMIT TRAN G2

GO

EXEC PopulateProfessional

@ProfessionalFName ='Manisha' ,

@ProfessionalLName ='Koirala',

@DateOfBirth ='08-16-1970' ,

@GenderID ='Female'

EXEC PopulateProfessional

@ProfessionalFName ='Pedro' ,

@ProfessionalLName ='Almodovar',

@DateOfBirth ='1999-11-24' ,

@GenderID ='Male'

EXEC PopulateProfessional

@ProfessionalFName ='Kartik' ,

@ProfessionalLName ='Aryan',

@DateOfBirth ='1990-11-22' ,

@GenderID ='Male'

EXEC PopulateProfessional

@ProfessionalFName ='Cecila' ,

@ProfessionalLName ='Roth',

@DateOfBirth ='1956-08-08' ,

@GenderID ='Female'

EXEC PopulateProfessional

@ProfessionalFName ='Eugenio' ,

@ProfessionalLName ='Dorbez',

@DateOfBirth ='1961-09-02' ,

@GenderID ='Male'

EXEC PopulateProfessional

@ProfessionalFName ='Preity' ,

@ProfessionalLName ='Zinta',

@DateOfBirth ='1975-01-31' ,

@GenderID ='Female'

EXEC PopulateProfessional

@ProfessionalFName ='Shahrukh' ,

@ProfessionalLName ='Khan',

@DateOfBirth ='1965-11-02' ,

@GenderID ='Male'

EXEC PopulateProfessional

@ProfessionalFName ='Pietro' ,

@ProfessionalLName ='Germi',

@DateOfBirth ='1914-09-14' ,

@GenderID ='Male'

EXEC PopulateProfessional

@ProfessionalFName ='Todd' ,

@ProfessionalLName ='Phillips',

@DateOfBirth ='2019-08-04' ,

@GenderID ='Male'

EXEC PopulateProfessional

@ProfessionalFName ='Bradley' ,

@ProfessionalLName ='Cooper',

@DateOfBirth ='1975-01-05' ,

@GenderID ='Male'

EXEC PopulateProfessional

@ProfessionalFName ='Leonardo' ,

@ProfessionalLName ='DiCaprio',

@DateOfBirth ='1974-11-11' ,

@GenderID ='Male'

EXEC PopulateProfessional

@ProfessionalFName ='Kate' ,

@ProfessionalLName ='Winslet',

@DateOfBirth ='1975-10-05' ,

@GenderID ='Female'

EXEC PopulateProfessional

@ProfessionalFName ='Julia' ,

@ProfessionalLName ='Roberts',

@DateOfBirth ='1978-10-27' ,

@GenderID ='Female'

-- Stored Procedure to populate tblReview

Create Procedure PopulateReviewTable

@MovName varchar(100),

@RatingName  varchar(30),

@ViewerFname varchar(50),

@Body varchar(500)

As

DECLARE @MovId INT

SET @MovId = (SELECT MovieID FROM tblMOVIE WHERE MovieName = @MovName)

-- error-handling --> we know the entire procedure will fail at the transaction if this is NULL

IF @MovId IS NULL

BEGIN

PRINT 'Error: Movie does not exist!!'

RAISERROR ('@@MovId cannot be NULL; statement is being terminated', 11,1)

RETURN

END

DECLARE @Rating INT

SET @Rating = (SELECT RatingID FROM tblRATING WHERE RatingName = @RatingName)

-- error-handling --> we know the entire procedure will fail at the transaction if this is NULL

IF @Rating IS NULL

BEGIN

PRINT 'Error: Rating does not exist!!'

RAISERROR ('@@Rating cannot be NULL; statement is being terminated', 11,1)

RETURN

END

DECLARE @ViewID INT

SET @ViewID = (SELECT ViewerID FROM tblVIEWER WHERE ViewerFName = @ViewerFname)

-- error-handling --> we know the entire procedure will fail at the transaction if this is NULL

IF @ViewID IS NULL

BEGIN

PRINT 'Error: Viewer does not exist!!'

RAISERROR ('@@ViewID cannot be NULL; statement is being terminated', 11,1)

RETURN

END

BEGIN TRAN G2

INSERT INTO tblREVIEW(ViewerID,RatingID,MovieID,Body)

VALUES (@ViewID,@Rating,@MovId,@Body)

-- error-handling

IF @@ERROR <> 0

BEGIN

PRINT '@@ERROR is showing a number <> 0; transaction is being terminated'

ROLLBACK TRAN G2

END

ELSE

COMMIT TRAN G2

GO

EXEC PopulateReviewTable

@MovName ='Love Aaj Kal',

@RatingName  ='Average',

@ViewerFname ='Seema',

@Body = 'Love Aajkal reminded me my childhood love story. I felt nostalgic while watch this movie. I really enjoyed.'

**1) Two (2) stored procedures (again, each student) to populate transaction-tables that include the following:**

**a) at least two (2) input parameters**

**b) at least two (2) variables**

Procedures created by Me (Nadia Asiz) : PopulateMovie, PopulateDescription

#First Stored Procedure

GO

CREATE OR ALTER PROCEDURE PopulateMovie

@MovName varchar(100),

@Yr INT,

@Lang varchar(50)

AS

DECLARE @L\_ID INT

SET @L\_ID = (SELECT LanguageID FROM tblLANGUAGE WHERE Language = @Lang)

-- error-handling --> we know the entire procedure will fail at the transaction if this is NULL

IF @L\_ID IS NULL

BEGIN

PRINT 'Error: Language does not exist!!'

RAISERROR ('@L\_ID cannot be NULL; statement is being terminated', 11,1)

RETURN

END

BEGIN TRAN G1

INSERT INTO tblMOVIE(MovieName,YearOfRelease,LanguageID)

VALUES (@MovName,@Yr,@L\_ID)

-- error-handling

IF @@ERROR <> 0

BEGIN

PRINT '@@ERROR is showing a number <> 0; transaction is being terminated'

ROLLBACK TRAN G1

END

ELSE

COMMIT TRAN G1

GO

EXEC PopulateMovie

@MovName='Pretty Woman',

@Yr='1990',

@Lang='English'

EXEC PopulateMovie

@MovName='Sweet Home Alabama',

@Yr='2002',

@Lang='English'

EXEC PopulateMovie

@MovName='Joker',

@Yr=2019,

@Lang='English'

EXEC PopulateMovie

@MovName='Titanic',

@Yr=1997,

@Lang='English'

EXEC PopulateMovie

@MovName='Todo Sobre Mi Madre',

@Yr=1999,

@Lang='Spanish'

EXEC PopulateMovie

@MovName='No Se Aceptan Devoluciones',

@Yr=2013,

@Lang='Spanish'

EXEC PopulateMovie

@MovName='Love Aaj Kal',

@Yr=2009,

@Lang='Hindi'

EXEC PopulateMovie

@MovName='Dil se',

@Yr=1999,

@Lang='Hindi'

EXEC PopulateMovie

@MovName='Divorce Italian Style',

@Yr=1961,

@Lang='Italian'

EXEC PopulateMovie

@MovName='La Dolce Vita',

@Yr=1961,

@Lang=Italian

EXEC PopulateMovie

@MovName='Freaky Friday',

@Yr=2003,

@Lang='English'

EXEC PopulateMovie

@MovName='Roma',

@Yr=2018,

@Lang='Spanish'

--Stored Procedure #2

--Stored Procedure to populate Description table

GO

CREATE OR ALTER PROCEDURE PopulateDescription

@MovName varchar(100),

@Gnr varchar(50)

AS

DECLARE @M\_ID INT, @G\_ID INT

SET @M\_ID = (SELECT MovieID FROM tblMOVIE WHERE MovieName=@MovName)

SET @G\_ID = (SELECT GenreID FROM tblGENRE WHERE GenreName=@Gnr)

-- error-handling --> we know the entire procedure will fail at the transaction if this is NULL

IF @M\_ID IS NULL

BEGIN

PRINT 'Error: Movie does not exist!!'

RAISERROR ('@M\_ID cannot be NULL; statement is being terminated', 11,1)

RETURN

END

IF @G\_ID IS NULL

BEGIN

PRINT 'Error: Genre does not exist!!'

RAISERROR ('@G\_ID cannot be NULL; statement is being terminated', 11,1)

RETURN

END

BEGIN TRAN G1

INSERT INTO tblDESCRIPTION(GenreID, MovieID)

VALUES (@G\_ID,@M\_ID)

-- error-handling

IF @@ERROR <> 0

BEGIN

PRINT '@@ERROR is showing a number <> 0; transaction is being terminated'

ROLLBACK TRAN G1

END

ELSE

COMMIT TRAN G1

GO

EXEC PopulateDescription

@MovName='Pretty Woman',

@Gnr='Comedy'

EXEC PopulateDescription

@MovName='Pretty Woman',

@Gnr='Romantic'

EXEC PopulateDescription

@MovName='Sweet Home Alabama',

@Gnr='Comedy'

EXEC PopulateDescription

@MovName='Sweet Home Alabama',

@Gnr='Romantic'

EXEC PopulateDescription

@MovName='Joker',

@Gnr='Drama'

EXEC PopulateDescription

@MovName='Joker',

@Gnr='Thriller'

EXEC PopulateDescription

@MovName='Love Aaj Kal',

@Gnr='Comedy'

EXEC PopulateDescription

@MovName='Love Aaj Kal',

@Gnr='Romantic'

EXEC PopulateDescription

@MovName='Todo Sobre Mi Madre',

@Gnr='Drama'

EXEC PopulateDescription

@MovName='No Se Aceptan Devoluciones',

@Gnr='Comedy'

EXEC PopulateDescription

@MovName='Freaky Friday',

@Gnr='Comedy'

EXEC PopulateDescription

@MovName='Freaky Friday',

@Gnr='Romantic'

EXEC PopulateDescription

@MovName='Roma',

@Gnr='Drama'

**2. A single-sentence description of a business rule on the project teams' database.**

The business rule that I’m adding is “ Movie that is not released cannot be rated”. Here, for example, in the case of movies that are being released in 2021 – eg: The Matrix 4, no user should be allowed to rate/review movies releasing post 2020. The constraint will be added to the tblREVIEW table that ensures that there are no reviews added for movies that haven’t already been released.

**3.  The SQL code to create the function to enforce the business rule defined in #2.**

--Enforcing business rule: Movies not released cannot be rated

GO

CREATE or ALTER FUNCTION fn\_MovieratingConstraint()

RETURNS INT

AS

BEGIN

DECLARE @RET INT = 0

IF EXISTS(SELECT \* FROM tblREVIEW R

JOIN tblMOVIE M ON R.MovieID=M.MovieID

WHERE M.YearOfRelease > YEAR(GETDATE()))

SET @RET = 1

RETURN @RET

END

GO

ALTER TABLE tblREVIEW

ADD CONSTRAINT CheckMovieYear

CHECK (dbo.fn\_MovieratingConstraint() = 0)

--Tried adding a review for The Matrix 4 and the db didn’t allow that because of the constraint. Hence, the Business rule works!

**4) The SQL code to create a function to enforce a computed column**

-- computed column to find out total number of nominations for a movie

GO

CREATE FUNCTION fn\_TotalMovieNominations(@M\_ID INT)

RETURNS INT

AS

BEGIN

DECLARE @RET INT = (SELECT COUNT(\*) FROM tblNOMINATION N

JOIN tblMOVIE M ON N.MovieID=M.MovieID

WHERE M.MovieID = @M\_ID)

RETURN @RET

END

GO

ALTER TABLE tblMOVIE

ADD TotalMovieNominations

AS dbo.fn\_TotalMovieNominations(MovieID)

SELECT \* from tblNOMINATION