-- INFO 340 ERD CODING PART

-- GROUP 2
-- Bruce Ng
-- Dajia Bao
-- Hannah Yoo

```
-- Create DATABASE MOVIE
CREATE DATABASE MOVIE;

USE MOVIE;
```

Create tables

```
-- Create table MOVIE
CREATE Table MOVIE(
MovieID INT IDENTITY(1,1) Primary Key NOT NULL,
MovieName varchar(50) NOT NULL,
MovieYear INT NOT NULL,
ReviewID INT NULL,
ProductionCompanyID INT NOT NULL
)

-- Create table MOVIE_GENRE
CREATE Table MOVIE_GENRE(
Movie_GenreID INT IDENTITY(1,1) Primary Key NOT NULL,
GenreID INT NOT NULL,
MovieID INT NOT NULL
)

-- Create table GENRE
CREATE Table GENRE(
GenreID INT IDENTITY(1,1) Primary Key NOT NULL,
GenreName varchar(50) NOT NULL,
GenreDescr varchar(100) NULL
)

-- Create table PRODUCTION_COMPANY
CREATE Table PRODUCTION_COMPANY(
ProductionCompanyID INT IDENTITY(1,1) Primary Key NOT NULL,
ProductionCompanyName varchar(50) NOT NULL,
ProductionCompanyDescr varchar(100) NULL,
ProductionCompanyCity varchar(50) NOT NULL,
ProductionCompanyState varchar(50) NOT NULL,
EmployeeID INT NOT NULL
)

-- Create table EMPLOYEE
CREATE Table EMPLOYEE(
EmployeeID INT IDENTITY(1,1) Primary Key NOT NULL,
EmployeeFName varchar(50) NOT NULL,
EmployeeLName varchar(50) NOT NULL,
```

```
EmployeeCity varchar(50) NOT NULL,
EmployeeState varchar(50) NOT NULL,
EmployeeAddress varchar(120) NOT NULL,
EmployeeZip varchar(25) NOT NULL
)

-- Create table POSITION
CREATE Table POSITION(
PositionID INT IDENTITY(1,1) Primary Key NOT NULL,
PositionName varchar(50) NOT NULL,
PositionDescr varchar(100) NULL
)

-- Create table EMPLOYEE_POSITION
CREATE Table EMPLOYEE_POSITION(
Employee_PositionID INT IDENTITY(1,1) Primary Key NOT NULL,
EmployeeID INT NOT NULL,
PositionID INT NOT NULL
)

-- Create table RATING
CREATE Table RATING(
RatingID INT IDENTITY(1,1) Primary Key NOT NULL,
RatingStar varchar(50) NULL,
RatingDescr varchar(500) NULL
)

-- Create table REVIEW
CREATE Table REVIEW(
ReviewID INT IDENTITY(1,1) Primary Key NOT NULL,
ReviewDate datetime NULL,
ReviewDescr varchar(500) NULL,
RatingID INT NOT NULL,
ReviewAuthor varchar(50) NOT NULL
)
```

Add foreign keys to link tables

```
-- ADD Foreign Key for table REVIEW
ALTER TABLE [dbo].[REVIEW]
ADD CONSTRAINT FK_REVIEW_RATING
FOREIGN KEY ([RatingID])
REFERENCES [dbo].[RATING]([RatingID])
-- ADD Foreign Key for table REVIEW***

-- ADD Foreign Key for table EMPLOYEE
ALTER TABLE [dbo].[PRODUCTION_COMPANY]
ADD CONSTRAINT FK_PC_EMPLOYEE
FOREIGN KEY ([EmployeeID])
REFERENCES [dbo].[EMPLOYEE]([EmployeeID])

ALTER TABLE [dbo].[EMPLOYEE_POSITION]
ADD CONSTRAINT FK_EP_EMPLOYEE
FOREIGN KEY ([EmployeeID])
REFERENCES [dbo].[EMPLOYEE]([EmployeeID])
```

```
-- ADD Foreign Key for table EMPLOYEE***
```

Check Constraints

```
-- This Constraint limits the date for reviews
-- only updated reviews later than 2001-01-01 is allowed
ALTER TABLE REVIEW
ADD CONSTRAINT updatedReviews
CHECK(ReviewDate > 2001-01-01)
-- This Constraint limits the date for reviews
-- only updated reviews later than 2001-01-01 is allowed***

-- This Constraint limits the date for movies
-- only updated movies later than 2001-01-01 is allowed
ALTER TABLE MOVIE
ADD CONSTRAINT updatedMovie
CHECK (MovieYear > 1800)
-- This Constraint limits the date for movies
-- only updated movies later than 2001-01-01 is allowed***

-- This Constraint limits the total amount of positions each
-- employee holds to be less than 25
ALTER TABLE Employee
ADD CONSTRAINT employeeMaxPositions
CHECK (Employee.PositionAmount < 25)
-- This Constraint limits the total amount of positions each
-- employee holds to be less than 25***
```

Populate tables

```
-- Populate table RATING
INSERT RATING(RatingStar, RatingDescr)
VALUES('*****', 'Awesome Movie'), ('****', 'Great Movie'), ('***', 'Good Movie'), ('**', 'Could of
Been Better'), ('*', 'Try Again')

DECLARE @RatingID INT
Set @RatingID = (Select RatingID From Rating Where RatingID=@RatingID)

INSERT INTO REVIEW(ReviewAuthor, ReviewDate, RatingID)
VALUES('Christy Lemires', '10-01-2014', 5), ('Christy Lemire', '10-02-2014', @RatingID),
('Christopher Orr', '10-03-2014', @RatingID), ('Wesley Morris', '10-04-2014', @RatingID), ('Andrew
Hehir', '10-05-2014', @RatingID)
-- Populate table RATING***

INSERT INTO GENRE(GenreName)
VALUES('Action'), ('Thriller'), ('Horror'), ('Comedy'), ('Mystery'), ('Adventure'), ('Romance'), ('Sci-
Fi'), ('Animation'), ('Family')

-- Populate table EMPLOYEE by copying data from Guitar_Shop
SELECT *
INTO [MOVIE].[dbo].[EMPLOYEE1]
FROM [Guitar_Shop].[dbo].[tblEMPLOYEE];

INSERT INTO [dbo].[EMPLOYEE] ([EmployeeFName], [EmployeeLName], [EmployeeState],
[EmployeeCity], [EmployeeAddress], [EmployeeZip])
```

```
SELECT [EmpFName], [EmpLName], [EmpState], [EmpCity], [EmpAddress], [EmpZip] FROM
[dbo].[EMPLOYEE1];

DROP TABLE [dbo].[EMPLOYEE1] -- Drop irrelevant table
-- Populate table EMPLOYEE by copying data from Guitar_Shop***
```

Stored Procedures

```
-- Populate table REVIEW by creating a stored procedure
CREATE PROCEDURE [dbo].[generateReview]
@ReviewDate Date,
@ReviewDescr varchar(100),
@RatingStar varchar(5),
@ReviewAuthor varchar(30)
AS
INSERT INTO REVIEW(ReviewDate, ReviewDescr, RatingID, ReviewAuthor)
VALUES(@ReviewDate, @ReviewDescr, (SELECT RatingID FROM Rating WHERE
RatingStar=@RatingStar), @ReviewAuthor)
-- Populate table REVIEW by creating a stored procedure***

-- REVIEW TABLE
exec generateReview '2014-11-21', "Well directed movie that was worth the time", '*****', 'Ben
Sachs'
exec generateReview '2014-11-16', "It's a film I didn't exactly enjoy and can't say I would
recommend", '**', 'Christy Lemire'
exec generateReview '2014-11-22', "Despite some awkward pacing, easy sentimentality, and
pandering it's still hard to resist.", '***', 'Eric Melin'
exec generateReview '2014-11-20', "The sentimentality and Hollywoodized convenience of
the storytelling prevent this mostly admirable film from achieving the escape velocities of
Kubrick or Tarkovsky.", '****', 'John Beifuss'
exec generateReview '2014-11-17', "A bombast-and-crisis megamovie, rocket-fuelled by
gobbledygeek", '**', 'Brian Gibson'
exec generateReview '2014-11-14', "The movie is complex and maddening, filled with
concepts over the heads of most mere mortals, yet grounded in barely enough humanity to
stimulate our senses. ", '***', 'Bob Bloom'
exec generateReview '2014-11-13', "The pace is gripping, the central performances
compelling and the action sequences thrilling.", '*****', 'Rich Phippen'
exec generateReview '2014-11-12', "Good special effects and acting but I didn't understand
one bit of the plot.", '***', 'Jackie K. Cooper'
exec generateReview '2014-11-12', "The spectacle is far more satisfying than the story ...", '***',
'CJ Johnson'
exec generateReview '2014-11-12', " While the ride is a bumpy one and the destination is a bit
of a letdown, getting there is a trip.", '***', 'Josh Hylton'
exec generateReview '2014-11-09', "It's an experience.", '*****', 'Sam Fragoso'
exec generateReview '2014-11-09', "This is probably the most audacious science fiction film
anyone has ever made or even tried to make.", '****', 'Mark R. Leeper'
exec generateReview '2014-11-09', "I'm not too proud to admit I couldn't follow the
convoluted storyline...", '**', 'Kam Williams'
exec generateReview '2014-11-30', "So much of this one feels disposable in the grand scheme
of things.", '**', 'Robert Levin'
exec generateReview '2014-11-27', "The film's commitment to market concerns over
storytelling ones is vividly obvious.", '*', 'Tim Brayton'
exec generateReview '2014-11-26', "In the excitable world of dystopian fantasy, the series
remains the best of the bunch.", '*****', 'Sandra Hall'
```

```
exec generateReview '2014-11-24', "It doesn't pull the trick off quite as slickly as it should,", '***', 'Rob Vaux'
exec generateReview '2014-11-22', "Dark and too often tedious.", '*', 'Boo Allen'
exec generateReview '2014-11-22', "Enjoy counting the cash", '*', 'Ed Whitfield'
exec generateReview '2014-11-22', "There's no worry about spoilers, because there's nothing to spoil.", '**', 'Al Alexander'
exec generateReview '2014-11-21', "Hollow, emotionally constipated, and unexpectedly sluggish. ", '**', 'Brian Orndorf'
exec generateReview '2014-11-21', "I was bored.", '*', 'David Kaplan'
exec generateReview '2014-11-24', "A lively and creative animated film about a robot with heart, empathy, compassion, and gentleness", '*****', 'Mary Ann Brussat'
exec generateReview '2014-11-12', "Bouncy and beguiling, it's family fare that values brains over brawn.", '****', 'Susan Granger'
-- REVIEW TABLE CREATED***

-- Populate TABLE POSITION by creating a stored procedure
CREATE Procedure generatePosition
@PositionName varchar(50),
@PositionDescr varchar(100)
AS
Insert into Position(PositionName, PositionDescr)
Values (@PositionName, @PositionDescr)
-- Populate TABLE POSITION by creating a stored procedure***

-- POSITION TABLE
exec generatePosition 'Actor', 'Portrays a character in the film'
exec generatePosition 'Actress', 'Portrays a character in the film'
exec generatePosition 'Director', 'Directs the film'
exec generatePosition 'Producer', 'Creates the conditions for making the film'
exec generatePosition 'Casting Director', 'Works closely with the director to cast the film'
exec generatePosition 'Director of Photography', 'In charge of the visual look and design of the entire movie'
exec generatePosition 'Camera Operator', 'The camera operatorassists the DP in camera operation'
exec generatePosition 'Sound Designer', 'Responsible for the ideation and creation of the overall soundtrack of the film'
exec generatePosition 'Composer', 'Writes original music to be heard i the film'
exec generatePosition 'Editor', 'Assemble the film'
exec generatePosition 'Continuty Stills Photography', 'Establish continuity referents for each shot covered in a day of shooting'
exec generatePosition 'Unit Production Manager', 'Coordinates, facilitates, and oversees the preparation of the production unit'
exec generatePosition 'Script Supervisor', 'Maintains a daily log of the shots covered and their relation to the script during the course of a production'
exec generatePosition 'Location Manager', 'Manages the discovery and securing of locations'
exec generatePosition 'Publicist', 'Promotes the film'
exec generatePosition 'Art Director', 'Oversees the design of the sets'
exec generatePosition 'Costume Designer', 'Responsible for the costumes of a production'
exec generatePosition 'Documentary Videographer', 'Captures 捷ehind the scenes?footage'
exec generatePosition 'Electrician', 'Operates electrical systems'
exec generatePosition 'Gaffer', 'Supervises set lighting'
exec generatePosition 'Visual Effect Supervisor', 'Designs and implements compositing effects'
exec generatePosition 'Set Decorator', 'Dress and decorates the set'
exec generatePosition 'Property Manager', 'Gathers, maintains, and manages all the props for a production'
```

```
exec generatePosition 'Key Hairdresser', 'Dresses and maintains the cast担 hair'
exec generatePosition 'Key Makeup Person', 'Applies and maintains the cast担 makeup'
-- POSITION TABLE CREATED***

-- Populate TABLE GENERATE by creating a stored procedure
CREATE Procedure [dbo].[GenerateGenre]
@GenreName varchar(20),
@GenreDescr varchar(100)
AS
INSERT into Genre(GenreName, GenreDescr)
Values(@GenreName, @GenreDescr)
-- Populate TABLE GENERATE by creating a stored procedure***

-- GENRE TABLE
exec GenerateGenre 'Action', 'Films in this genre often involve car chases, gun fights and
hand-to-hand combats. Violence is the key characteristic of this film genre.'
exec GenerateGenre 'Adventure', 'The main aspects of the movies in this category are exotic
locales, historical settings, epic expeditions and anything that brings thrills to the audience.'
exec GenerateGenre 'Animation', 'Consists of 2D animation, clay animation, paper animation,
stop motion animation and computer generated animation.'
exec GenerateGenre 'Biography', 'A film that dramatizes the life of an actual person or
people'
exec GenerateGenre 'Comedy', 'Humor is the main driving force of comedy films. Anything
with slapstick moments, witty dialogue and satirical elements are included in this genre. '
exec GenerateGenre 'Crime', 'Stories whose central struggle is about catching a criminal.'
exec GenerateGenre 'Documentary', 'Documentary movies involve putting together real life
events and people to tell a particular story. Movie in this genre often involve a narrator,
interviews and real footages of real events'
exec GenerateGenre 'Drama', 'Dramatic movies with serious themes and intense character
development make up films in this genre. They often portray realistic situations with realistic
people, but they sometimes involve more fantastical elements.'
exec GenerateGenre 'Family', 'These are films that are fun for the entire family to watch
together, from children to grandparents. Family movies often include teaching moments
when characters deal with issues or situations familiar to kids. '
exec GenerateGenre 'Fantasy', 'Stories which are animated, or whose central struggle plays
out in two worlds - the "real" world and an imaginary world.'
exec GenerateGenre 'Horror', 'Movies in the horror genre involve blood, gore, the
supernatural and things that go bump in the night. It includes ghost stories, alien invasions,
zombie flicks, slasher movies and everything that makes one afraid of going to sleep at night.'
exec GenerateGenre 'Musical', 'Musicals are often lighthearted stories with comedy and
drama going hand in hand. These movies involve a lot of singing and dancing to
complement the storytelling.'
exec GenerateGenre 'Mystery', 'It focuses on the efforts of the detective, private investigator
or amateur sleuth to solve the mysterious circumstances of a crime by means of clues,
investigation, and clever deduction.'
exec GenerateGenre 'Romance', 'Stories whose central struggle is between two people who
each want to win or keep the love of the other.'
exec GenerateGenre 'Sci-Fi', 'This genre involves movies that feature futuristic technology,
interstellar travel, strange monsters and anything that is very imaginative that does not fit in
the real world. Films in this genre may often intersect in action, adventure and horror
categorizations.'
exec GenerateGenre 'Sport', 'A Sport Film revolves around a sport setting, event, or an athlete.
Often, these films will center on a single sporting event that carries significant importance. '
exec GenerateGenre 'Thriller', 'Stories whose central struggle pits an innocent hero against a
lethal enemy who is out to kill him or her.'
```

```
exec GenerateGenre 'War', 'Whether it is an ancient battle or World War II, war movies always involve combat and tales of life in the battlefield. Action and drama are two key components in war movies.'
exec GenerateGenre 'Western', ' It started out as a film genre that only relied on horses, guns, dusty towns, bar fights and cowboys for categorization. However, the themes of showdowns, revenge and being outlaws are also being applied to stories not set in the Old West.'
-- GENRE TABLE CREATED***

-- TABLE EmployeePosition is an associative entity.
-- We create this table by randomly generating EmployeeID and PositionID, then put them in the table
CREATE PROCEDURE generateEmployeePosition @run INT
AS

DECLARE @Employee INT
DECLARE @Rand varchar(12)
DECLARE @Position INT

WHILE @run > 0
BEGIN
SET @RAND = (SELECT RAND())

SET @Employee = (SELECT CASE
        WHEN substring(@rand, 5, 3) = 000
        THEN 754
        ELSE substring(@rand, 5, 3)
        END)

SET @Position = (SELECT CASE
        WHEN substring (@rand, 3, 2) BETWEEN 25 AND 35
        Then 3
        WHEN substring (@rand, 3, 2) BETWEEN 35 AND 65
        Then 8
        WHEN substring (@rand, 3, 2) = 00
        Then 4
        When substring (@rand, 3, 2) Between 65 AND 99
        Then 11
        ELSE substring(@rand, 3, 2)
        END)

        INSERT Into EMPLOYEE_POSITION (EmployeeID, PositionID)
        VALUES (@Employee, @Position)

        Set @run = @run - 1
        END
-- Populate TABLE EmployeePosition by creating a stored procedure***
-- EXECTUED***

-- We populate TABLE ProductionCompany by using wikipedia resources and transfer the data to
-- our database. We first download the online info to an excel file, then
-- import the info to TABLE PRODUCTION_COMPANIES. At the end, we move the movie production company data
-- to TABLE ProductionCompany as well as randomly generating a MovieID with each of entries
```

```
CREATE PROCEDURE generateProductionCompany @runNum INT
AS

DECLARE @Rand varchar(12)
DECLARE @EmpID AS INT;
DECLARE @pc_id AS INT;

WHILE @runNum > 0
BEGIN

SET @Rand = (SELECT RAND())
SET @EmpID = (SELECT CASE
                        WHEN SUBSTRING(@Rand, 3, 4) > 6668
                        THEN SUBSTRING(@Rand, 3, 3)
                        ELSE
                        SUBSTRING(@Rand, 3, 4)
                        END)

SET @pc_id = (SELECT CASE
                        WHEN SUBSTRING(@Rand, 7, 2) = 00
                        THEN 25
                        ELSE SUBSTRING(@Rand, 7, 2)
                        END)

INSERT INTO [dbo].[PRODUCTION_COMPANY]([ProductionCompanyName],
[ProductionCompanyDescr], [ProductionCompanyCity], [EmployeeID])
VALUES ((SELECT [ProductionCompanyName] FROM [dbo].[PRODUCTION_COMPANIES]
WHERE [PC_ID] = @pc_id),
(SELECT [ProductionCompanyDescr] FROM [dbo].[PRODUCTION_COMPANIES] WHERE [PC_ID]
= @pc_id),
(SELECT [ProductionCompanyHeadquarterLocations] FROM
[dbo].[PRODUCTION_COMPANIES] WHERE [PC_ID] = @pc_id),
@EmpID)

 SET @runNum = @runNum - 1
 END
-- Populate TABLE ProductionCompany by creating a stored procedure***
-- EXECTUED***

-- We populate TABLE Movie by using IMDB resources and transfer the data to
-- our database. We first download the online info to an excel file, then
-- import the info to TABLE CopyMovie. At the end, we move the movie  data
-- to TABLE Movie as well as randomly generating a ReviewID and a ProductionCompanyID
with each of entries
CREATE PROCEDURE generateMovie @runNum INT
AS

DECLARE @Rand varchar(12)
DECLARE @reviewID AS INT;
DECLARE @companyID AS INT;
DECLARE @movieID AS INT;

WHILE @runNum > 0
BEGIN
```

```
SET @Rand = (SELECT RAND())

SET @reviewID = (SELECT CASE
                        WHEN SUBSTRING(@Rand, 3, 3) < 18
                        THEN 27
                        ELSE
                        SUBSTRING(@Rand, 3, 3)
                        END)

SET @companyID = (SELECT CASE
                        WHEN SUBSTRING(@Rand, 7, 4) < 1358
                        THEN SUBSTRING(@Rand, 7, 4) + 5000
                        ELSE SUBSTRING(@Rand, 7, 4)
                        END)

SET @movieID = (SELECT CASE
                        WHEN SUBSTRING(@Rand, 6, 2) = 00
                        THEN 34
                        ELSE SUBSTRING(@Rand, 6, 2)
                        END)

INSERT INTO [dbo].[MOVIE]([MovieName], [MovieYear], [ReviewID], [ProductionCompanyID])
VALUES ((SELECT [F1] FROM [dbo].[CopyMovie] WHERE [M_ID] = @movieID),
(SELECT [F2] FROM [dbo].[CopyMovie] WHERE [M_ID] = @movieID), @reviewID,
@companyID)

SET @runNum = @runNum - 1
END
-- Populate TABLE Movie by creating a stored procedure***
-- EXECTUED***

-- TABLE MovieGenre is an associative entity.
-- We create this table by randomly generating MovieID and GenreID, then put them in the
table
-- In the real world, we understand a movie cannot have random genre types but for the
purpose of
-- this exercise, we use this stored procedure to quickly populate the table
CREATE PROCEDURE generateMovieGenre @run INT
AS

DECLARE @GenreID INT
DECLARE @Rand varchar(12)
DECLARE @MovieID INT

WHILE @run > 0
BEGIN

SET @RAND = (SELECT RAND())

SET @GenreID = (SELECT CASE
        WHEN substring(@rand, 3, 3) < 24
        THEN substring(@rand, 3, 3) + 100
        ELSE substring(@rand, 3, 3)
        END)
```

```
SET @MovieID = (SELECT CASE
        WHEN substring (@rand, 6, 4) < 1900
        Then substring (@rand, 6, 4) + 5000
        ELSE substring(@rand, 6, 4)
        END)

INSERT Into [dbo].[MOVIE_GENRE] ([GenreID], [MovieID])
VALUES (@GenreID, @MovieID)

SET @run = @run - 1
END
-- Populate TABLE MovieGenre by creating a stored procedure***
-- EXECTUED***
```

Views

```
-- This is a VIEW to find top 10 movies by the amount of reviews
CREATE VIEW TopMovies
As

SELECT TOP 10 M.MovieName, Count(*) As ReviewsCount
FROM Review R
JOIN Movie M On R.ReviewID = M.ReviewID

GROUP BY M.MovieName
ORDER BY ReviewsCount DESC
-- This is a VIEW to find top 10 movies by the amount of reviews***

-- This is a VIEW returning top 5
-- most popular movie genre types by the amount of reviews
CREATE VIEW TopGenre
AS

SELECT TOP 5 g.GenreName, COUNT(*) AS ReviewCounts
FROM GENRE AS g
JOIN MOVIE_GENRE AS mg ON mg.GenreID = g.GenreID
JOIN MOVIE AS m ON m.MovieID = mg.MovieID
JOIN REVIEW AS r ON r.ReviewID = m.ReviewID
JOIN RATING AS ra ON ra.RatingID = r.RatingID
WHERE ra.RatingStar LIKE '****'
GROUP BY g.GenreName
ORDER BY ReviewCounts DESC
-- This is a VIEW returning top 5
-- most popular movie genre types by the amount of reviews***
-- EXample code: "SELECT * FROM TopGenre"

-- This is a VIEW returning top 10 most popular actor/actress(s)
-- by the amount of movies they participate
CREATE VIEW MostPopularCelebrity
AS

SELECT TOP 10 e.EmployeeID, COUNT(*) AS TotalMovieCount
FROM EMPLOYEE AS e
JOIN PRODUCTION_COMPANY AS pc ON pc.EmployeeID = e.EmployeeID
JOIN MOVIE AS m ON m.ProductionCompanyID = pc.ProductionCompanyID
```

```
GROUP BY e.EmployeeID
ORDER BY TotalMovieCount DESC
-- This is a VIEW returning top 10 most popular actor/actress(s)
-- by the amount of movies they participate***
-- Example code: "SELECT * FROM MostPopularCelebrity"
```

Functions

```
-- This is a user-defined FUNCTION
-- counting number of reviews for different kinds of ratings
CREATE FUNCTION CountReview(@reviewRating int)
RETURNS INT
AS
BEGIN
	DECLARE @countTotal int

	SELECT @countTotal = (SELECT COUNT(*)
	FROM REVIEW AS r
	JOIN RATING AS ra ON ra.RatingID = r.RatingID
	JOIN MOVIE AS m ON m.ReviewID = r.ReviewID
	WHERE ra.RatingID  = @reviewRating
	GROUP BY ra.RatingID)

	RETURN (@countTotal)
END
-- This is a user-defined FUNCTION
-- counting number of reviews for different kinds of ratings***
-- Example code: "SELECT [dbo].[CountReview](3)"

-- Add a new computed column for Employee's position amount
ALTER TABLE Employee
ADD PositionAmounts AS (
[dbo].amountOfPositions(EmployeeID)
)

-- This is a user-defined FUNCTION
-- counting number of amount of positions each employee hold
CREATE FUNCTION amountOfPositions(@EmployeeID INT)
RETURNS INT
AS
BEGIN
	DECLARE @Total INT

	SET @Total = (SELECT COUNT(P.PositionID)
	FROM POSITION P
	JOIN EMPLOYEE_POSITION EP ON P.PositionID = EP.PositionID
	JOIN EMPLOYEE E ON EP.EmployeeID = E.EmployeeID
	WHERE E.EmployeeID = @EmployeeID
	GROUP BY E.EmployeeID)

	RETURN @total
END
-- This is a user-defined FUNCTION
-- counting number of amount of positions each employee hold***
```

```sql
-- This is a user-defined FUNCTION
-- counting the total amount of movies for one type of genre
CREATE FUNCTION countGenre(@genreName varchar(50))
RETURNS INT
AS

BEGIN
	DECLARE @countGenreAmount INT

	SET @countGenreAmount =  (SELECT COUNT(*)
	FROM GENRE AS g
	JOIN MOVIE_GENRE AS mg ON mg.GenreID = g.GenreID
	JOIN MOVIE AS m ON m.MovieID = mg.MovieID
	WHERE g.GenreName LIKE @genreName)

	RETURN @countGenreAmount
END
-- This is a user-defined FUNCTION
-- counting the total amount of movies for one type of genre
-- Example code: "SELECT [dbo].[countGenre]('Action')"
```