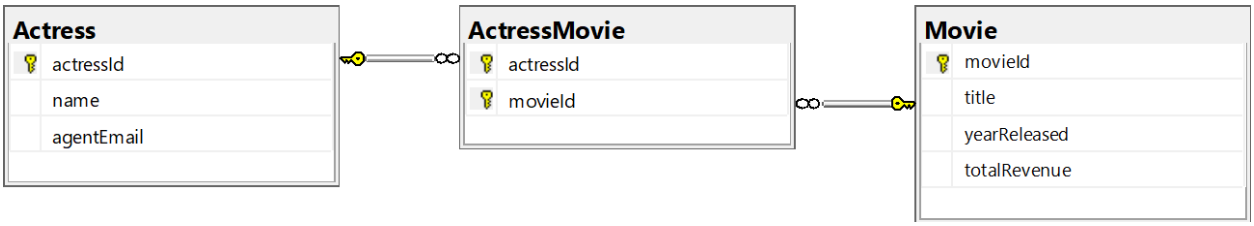
Overview

In the United States, the movie industry has proven to be one of the most resilient to periods of economic recession. Movie studio executives, for example, reported that total U.S. box office revenues exceeded $10 billion for the first time ever during the recession in 2009, with revenues in North America exceeding $11.3 billion in 2019. In an effort to better understand how different actresses impact a movie's profitability, one of these executives wants to explore the relationship between a movie's box office revenues and the actresses who appear in the movie. If certain actresses are found to be associated with higher box office revenues, the studio executive might consider hiring analytics personnel to more carefully study the relationships among actresses, revenues, salaries, directors, production costs, etc.

Details

To begin the studio executive's project, a database of actresses and movies will need to be created. The design that has been chosen for this database is shown in the figure below:



The ***Actress*** table holds information for all of the actresses that are being considered. Attributes for the actress table include:

- ***actressId*** – Primary key. A unique integer that uniquely identifies this actress.
- ***name*** – The name of this actress. Maximum length = 50 characters.
- ***agentEmail*** – The email address of the agent who represents this actress. Maximum length = 50 characters.

The ***Movie*** table holds information for all of the movies that are being considered. Attributes for the movie table include:

- ***movieId*** – Primary key. A unique integer that uniquely identifies this movie.
- ***title*** – The title of this movie. Maximum length = 100 characters.
- ***yearReleased*** – The calendar year in which this movie was released.
- ***totalRevenue*** – The total worldwide gross revenue (in U.S. dollars) earned by this movie.

The ***ActressMovie*** table serves as an associative entity (intersection table) that enables a many-to-many relationship between the actress and movie tables. This allows a movie to have more than one actress, and allows each actress to appear in more than one movie. The only attributes in this table are the primary keys from the movie and actress tables, which together comprise a composite primary key for the ActressMovie table, and individually serve as foreign key links back to their respective parent tables.

Tasks

Your tasks for this assignment are to:

1. Write a series of SQL statements that will fully implement the database depicted above in Microsoft SQL Server, including tables, attributes, data types, keys, and relationships.
2. Write a series of SQL statements that will insert the following data into your database:

Actress

actressId	*name*	*agentEmail*
1	Zoe Saldana	amara@bigbigtalent.com
2	Scarlett Johansson	tallulah@genericagency.com
3	Emma Watson	cho@leadingtalentagency.com
4	Letitia Wright	adira@hollywoodtalent.com
5	Emma Thompson	fatima@uktalentstars.co.uk

Movie

movieId	*title*	*yearReleased*	*totalRevenue*
101	Avatar	2009	2,787,965,087
102	Avengers: Endgame	2019	2,797,800,564
103	Ready Player One	2018	579,278,642
104	Harry Potter and the Deathly Hallows: Part II	2011	1,341,656,673
105	Beauty and the Beast	2017	1,263,204,500

ActressMovie

actressId	*movieId*
2	102
4	103
3	105
1	101
4	102
5	105
1	102
3	104
5	104

3. Write a single SQL statement that will select the name of each actress and the average box office revenue for all of the movies in which that actress has appeared. These results should be sorted by average revenue in descending order (largest to smallest).
4. Create all of your SQL statements for tasks 1-3 above in a single SQL Server query window, and save the results as a SQL Server query file (i.e., a ***.sql*** file). Your statements should be properly ordered so that all of the above tasks will be completed in succession when the statements are executed.

TIP #1: You may want to consider including the following three statements at the very top of your SQL Server query window:

```
IF OBJECT_ID('ActressMovie') IS NOT NULL DROP TABLE ActressMovie;
IF OBJECT_ID('Movie') IS NOT NULL DROP TABLE Movie;
IF OBJECT_ID('Actress') IS NOT NULL DROP TABLE Actress;
```

These statements will drop (delete) the ActressMovie, Movie, and Actress tables if they already exist in your database, thus allowing you to start with a "clean slate" each time you execute your queries.

TIP #2: Use a floating-point data type (e.g., *float*) for the *totalRevenue* attribute. This will allow you to avoid problems with integer division when computing each actress's average revenue per movie.

Deliverables

To ensure that you receive credit for this assignment, please submit the ***.sql*** file that contains your SQL statements for tasks 1-3 above using the appropriate link on the course website. Remember: your SQL statements should be properly ordered so that tasks 1-3 will be completed in succession when the statements are executed.