CS669

SiCheng Yi

LAB3

**Section One – Aggregating Data**

1. Create the tables in the schema, including all of their columns, datatypes, and constraints, and populate the tables with data. Most but not all of the data is given to you in the table below; *you should also insert information for one additional movie series of your choosing.* Although the data is in flattened representation below, you will of course insert the data relationally into the schema with foreign keys referencing the appropriate primary keys.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Genre** | **Creator** | **Series** | **Suggested Price** | **Movie** | **Length** |
| Fantasy | George Lucas | Star Wars | $129.99 | Episode I: The Phantom Menace | 136 |
| Fantasy | George Lucas | Star Wars | $129.99 | Episode II: Attack of the Clones | 142 |
| Fantasy | George Lucas | Star Wars | $129.99 | Episode III: Revenge of the Sith | 140 |
| Fantasy | George Lucas | Star Wars | $129.99 | Episode IV: A New Hope | 121 |
| Family Film | John Lasseter | Toy Story | $22.13 | Toy Story | 121 |
| Family Film | John Lasseter | Toy Story | $22.13 | Toy Story 2 | 135 |
| Family Film | John Lasseter | Toy Story | $22.13 | Toy Story 3 | 148 |
| Fantasy | John Tolkien | Lord of the Rings |  | The Lord of the Rings: The Fellowship of the Ring | 228 |
| Fantasy | John Tolkien | Lord of the Rings |  | The Lord of the Rings: The Two Towers | 235 |
| Fantasy | John Tolkien | Lord of the Rings |  | The Lord of the Rings: The Return of the King | 200 |

Note that the suggested price for the Lord of the Rings series is null (has no value).

CREATE TABLE Genre(

genre\_id DECIMAL(12) NOT NULL PRIMARY KEY,

genre\_name VARCHAR(64) NOT NULL);

CREATE TABLE Creator(

creator\_id DECIMAL(12) NOT NULL PRIMARY KEY,

first\_name VARCHAR(64) NOT NULL,

last\_name VARCHAR(64) NOT NULL);

CREATE TABLE Movie\_series(

movie\_series\_id DECIMAL(12) NOT NULL PRIMARY KEY,

genre\_id DECIMAL(12) NOT NULL,

creator\_id DECIMAL(12) NOT NULL,

series\_name VARCHAR(255) NOT NULL,

suggested\_price DECIMAL(8,2) NULL,

FOREIGN KEY (genre\_id) references Genre(genre\_id),

FOREIGN KEY (creator\_id) references CREATOR(creator\_id));

CREATE TABLE Movie(

movie\_id DECIMAL(12) NOT NULL PRIMARY KEY,

movie\_series\_id DECIMAL(12) NOT NULL,

movie\_name VARCHAR(64) NOT NULL,

length\_in\_minutes DECIMAL(4),

FOREIGN KEY (movie\_series\_id) references Movie\_series(movie\_series\_id));

图形用户界面, 应用程序

描述已自动生成

insert into Genre(genre\_id, genre\_name)

values(1,'Fantasy');

insert into Genre(genre\_id, genre\_name)

values(2,'Family Film');

insert into Genre(genre\_id, genre\_name)

values(3,'Action');

insert into Creator(creator\_id, first\_name, last\_name)

values(11,'George', 'Lucas');

insert into Creator(creator\_id, first\_name, last\_name)

values(12,'John', 'Lasseter');

insert into Creator(creator\_id, first\_name, last\_name)

values(13,'John', 'Tolkien');

insert into Creator(creator\_id, first\_name, last\_name)

values(14,'Jun', 'Favreau');

insert into Movie\_series(movie\_series\_id, genre\_id, creator\_id, series\_name, suggested\_price)

values(101,1,11,'Star wars',129.99);

insert into Movie\_series(movie\_series\_id, genre\_id, creator\_id, series\_name, suggested\_price)

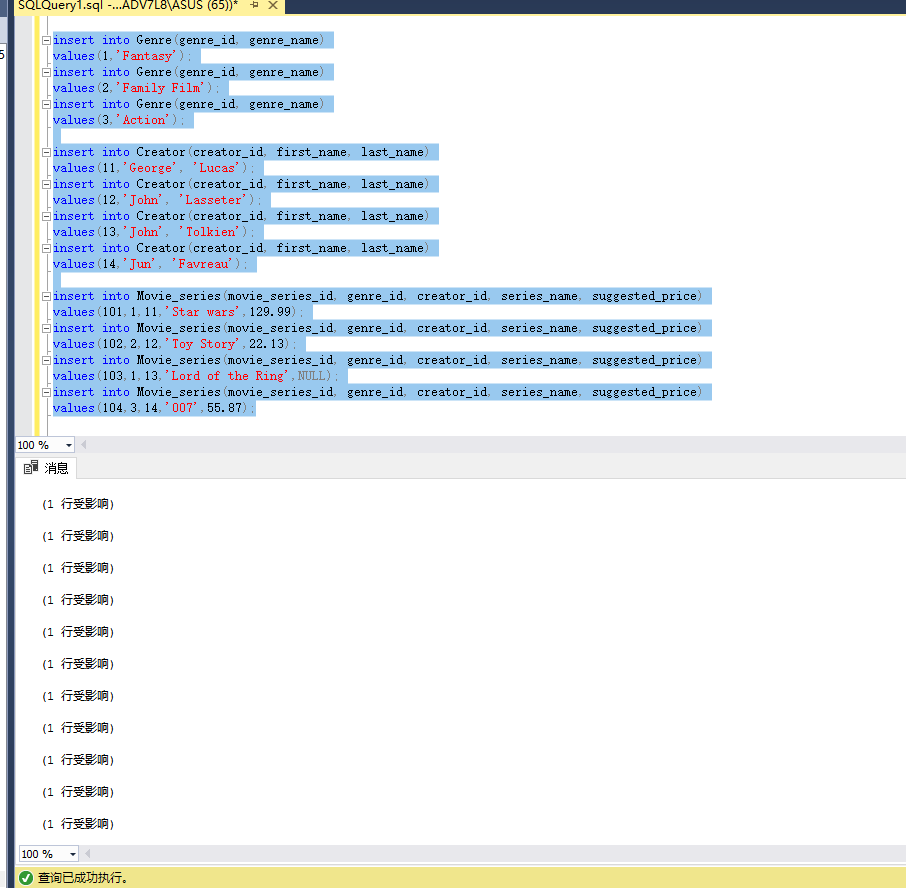
values(102,2,12,'Toy Story',22.13);

insert into Movie\_series(movie\_series\_id, genre\_id, creator\_id, series\_name, suggested\_price)

values(103,1,13,'Lord of the Ring',NULL);

insert into Movie\_series(movie\_series\_id, genre\_id, creator\_id, series\_name, suggested\_price)

values(104,3,14,'007',55.87);



insert into Movie(movie\_id, movie\_series\_id, movie\_name, length\_in\_minutes)

values(1001,101,'Episode I: The Phantom Menace',136);

insert into Movie(movie\_id, movie\_series\_id, movie\_name, length\_in\_minutes)

values(1002,101,'Episode II: Attack of the Clones',142);

insert into Movie(movie\_id, movie\_series\_id, movie\_name, length\_in\_minutes)

values(1003,101,'Episode III: Revenge of the Sith',140);

insert into Movie(movie\_id, movie\_series\_id, movie\_name, length\_in\_minutes)

values(1004,101,'Episode IV: A New Hope',121);

insert into Movie(movie\_id, movie\_series\_id, movie\_name, length\_in\_minutes)

values(1005,102,'Toy Story',121);

insert into Movie(movie\_id, movie\_series\_id, movie\_name, length\_in\_minutes)

values(1006,102,'Toy Story 2',135);

insert into Movie(movie\_id, movie\_series\_id, movie\_name, length\_in\_minutes)

values(1007,102,'Toy Story 3',148);

insert into Movie(movie\_id, movie\_series\_id, movie\_name, length\_in\_minutes)

values(1008,103,'The Lord of the Rings: The Fellowship of the Ring',228);

insert into Movie(movie\_id, movie\_series\_id, movie\_name, length\_in\_minutes)

values(1009,103,'The Lord of the Rings: The Two Towers',235);

insert into Movie(movie\_id, movie\_series\_id, movie\_name, length\_in\_minutes)

values(1010,103,'The Lord of the Rings: The Return of the King',200);

insert into Movie(movie\_id, movie\_series\_id, movie\_name, length\_in\_minutes)

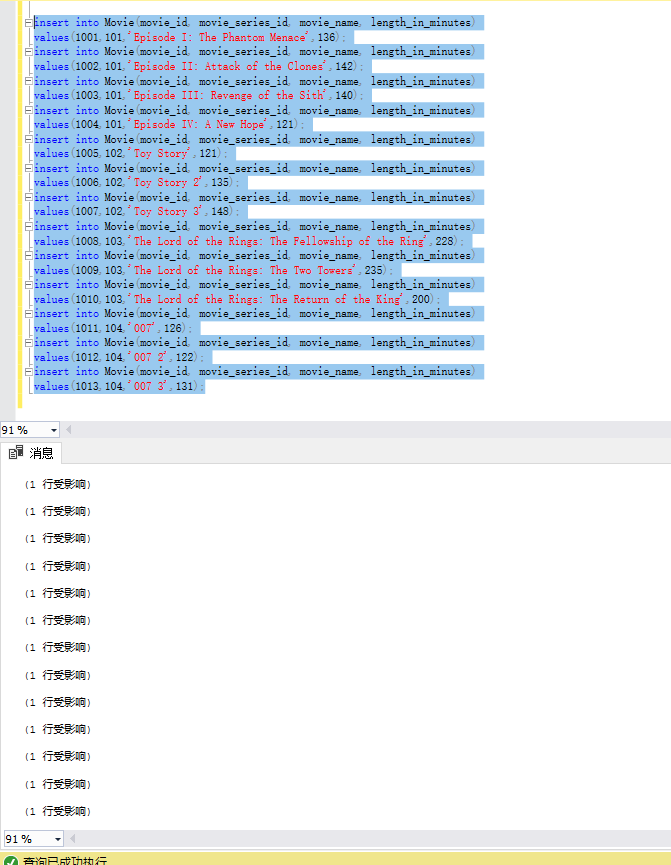
values(1011,104,'007',126);

insert into Movie(movie\_id, movie\_series\_id, movie\_name, length\_in\_minutes)

values(1012,104,'007 2',122);

insert into Movie(movie\_id, movie\_series\_id, movie\_name, length\_in\_minutes)

values(1013,104,'007 3',131);

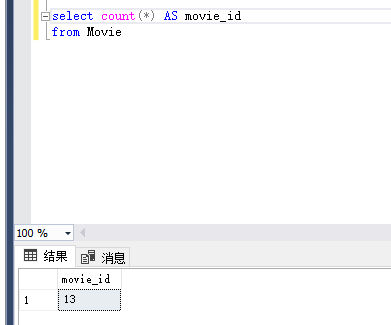


1. A video reseller needs to know how many movies are available. Write a single query

to fulfill this request.

select count(\*) AS movie\_id

from Movie



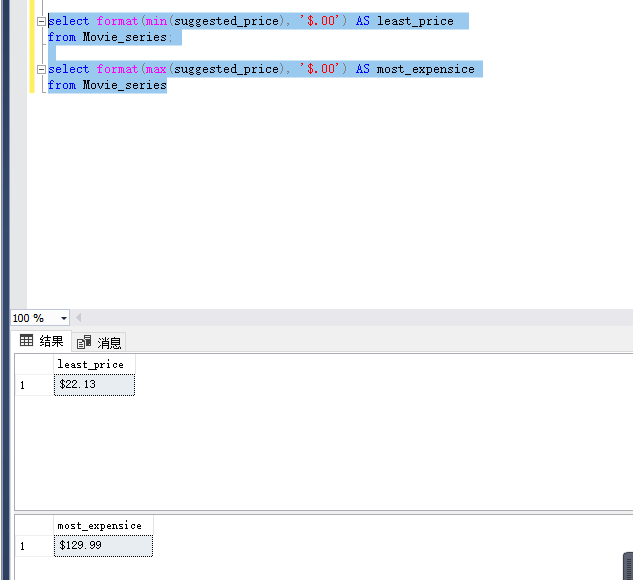
1. The same video reseller needs to know the price of the most expensive and least expensive series. Write two queries that fulfill this request, and also explain how and why the SQL processor treated the suggested price for the Lord of the Rings series differently than the other suggested price values.

select format(min(suggested\_price), '$.00') AS least\_price

from Movie\_series;

select format(max(suggested\_price), '$.00') AS most\_expensice

from Movie\_series



The SQL processor will just make the movie stand still, Because the aggregate functions will ignores null values.

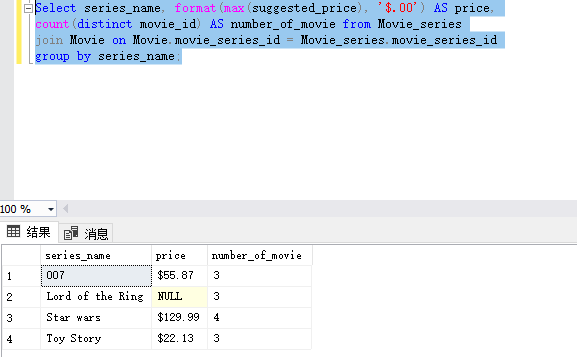
1. A film production company is considering purchasing the rights to extend a series, and needs to know the names and prices of all movie series, along with the number of movies in each series. Write a single query to fulfill this request.

Select series\_name, format(max(suggested\_price), '$.00') AS price,

count(distinct movie\_id) AS number\_of\_movie from Movie\_series

join Movie on Movie.movie\_series\_id = Movie\_series.movie\_series\_id

group by series\_name;



1. The same film production company wants to create movies in a genre that has at least 7 associated movies. Write a single query to fulfill this request, making sure to list only genres that have at least 7 associated movies, along with the number of movies for the genre.

Select genre\_name, count(distinct movie\_id) AS number\_of\_movie

from Movie\_series

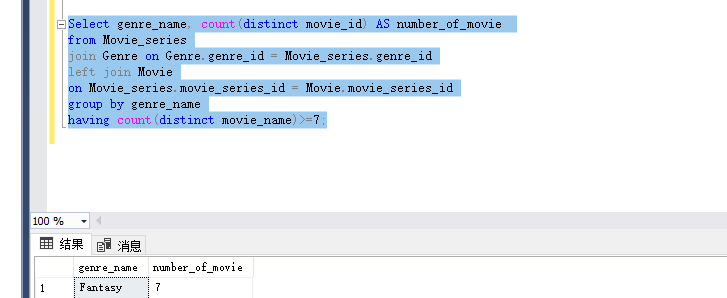
join Genre on Genre.genre\_id = Movie\_series.genre\_id

left join Movie

on Movie\_series.movie\_series\_id = Movie.movie\_series\_id

group by genre\_name

having count(distinct movie\_name)>=7;



1. Boston University wants to offer its students a movie-binge weekend by playing every movie in a series. To make sure the series is as bingeable as possible, BU wants to be sure the series will run for at least 10 hours. Write a single query that gives this information, with useful columns.

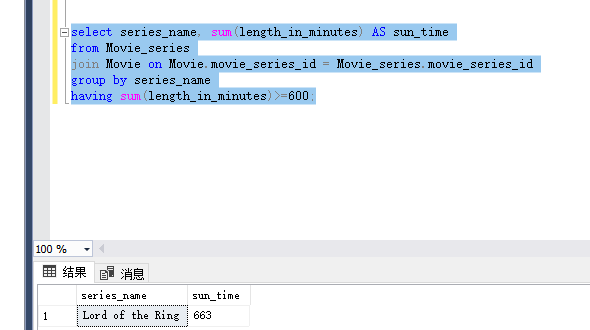
select series\_name, sum(length\_in\_minutes) AS sun\_time

from Movie\_series

join Movie on Movie.movie\_series\_id = Movie\_series.movie\_series\_id

group by series\_name

having sum(length\_in\_minutes)>=600;



1. A research institution requests the names of all movie series’ creators, as well as the number of “Family Film” movies they have created (even if they created none). The institution wants the list to be ordered from most to least; the creator who created the most family films will be at the top of the list, and the one with the least will be at the bottom. Write a single query that gives this information, with useful columns.

select concat(first\_name,' ',last\_name) AS Creators,

count(Movie.movie\_name) AS number\_Fantasy\_Movies

from Movie\_series

join Movie on Movie\_series.movie\_series\_id = Movie.movie\_series\_id

join Genre on Movie\_series.genre\_id = Genre.genre\_id

and genre.genre\_name = 'Family Film'

right join Creator on Creator.creator\_id = Movie\_series.creator\_id

group by concat(first\_name,' ',last\_name)

order by number\_Fantasy\_Movies DESC;

