|  |
| --- |
| **Overview of the Lab** |
| Sometimes we are interested in the result of aggregating multiple data items rather than in individual data items. For example, a store may be interested in the monetary amount of a single sale, but may be equally or more interested in the sum the monetary amount of all sales that occurred on a specific day. SQL provides many useful ways to aggregate data. The objective of this lab is for you to learn to aggregate data using SQL.  From a technical perspective, together, we will learn:   * how to use aggregate functions generally. * how to count items in a table. * how to determine minimum and maximum values. * how to filter rows based upon aggregate values. * how to use aggregation with joins together to answer more complex use cases with related data. |

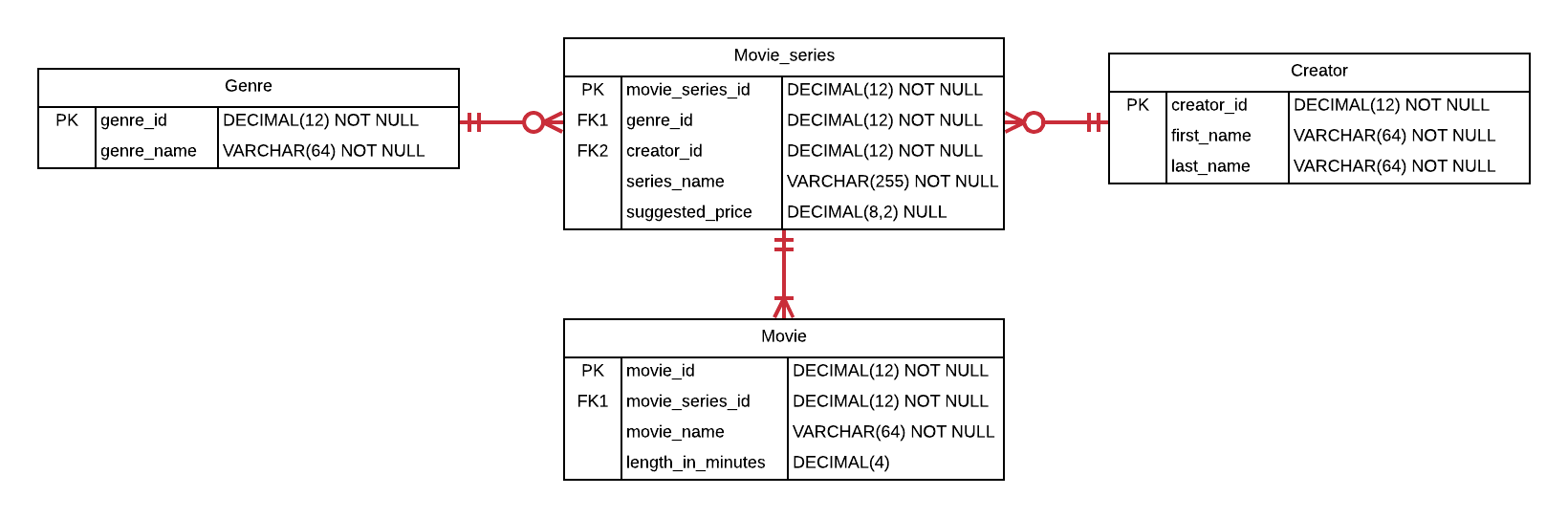
|  |
| --- |
| **Lab 3 Explanations Reminder** |
| As a reminder, it is important to read through the Lab 3 Explanation document to successfully complete this lab, available in the assignment inbox alongside this lab. The explanation document illustrates how to correctly execute each SQL construct step-by-step, and explains important theoretical and practical details. |

|  |
| --- |
| **Other Reminders** |
| * The examples in this lab will execute in modern versions of Oracle, Microsoft SQL Server, and PostgreSQL as is. * The screenshots in this lab display execution of SQL in the default SQL clients supported in the course – Oracle SQL Developer, SQL Server Management Studio, and pgAdmin – but your screenshots may vary somewhat as different version of these clients are released. * Don’t forget to commit your changes if you work on the lab in different sittings, using the “COMMIT” command, so that you do not lose your work. |

**Section One – Aggregating Data**

**Section Background**

To practice aggregating data, you will be working with the following simplified Movie Series schema.



This schema contains basic information about various movie series and the movies that comprise them, such as the Star Wars series with its movies.

In this schema, the Movie\_series table represents the overall movie series, and contains a primary key, the name of the series, foreign keys to its genre and creator, and a suggested price for the entire series. The Genre table represents the genre of a movie such as “Fantasy”, “Family Film”, and the like. It contains a primary key and the name of the genre. The Creator table represents who created the series, and contains a primary key and the name of each creator. The Movie table represents movies that comprise each movie series, and contains a primary key, a foreign key to the movie’s series, the name of the movie, and the length of the movie, in minutes.

The schema is intentionally simplified compared to what you might see in a real-world production schema. Many attributes and entities that would exist in a production database are not present. Nevertheless, there is sufficient complexity in the existing relationships and attributes to challenge you to learn various aggregation scenarios you encounter in real-world schemas.

As a reminder, for each step that requires SQL, make sure to capture a screenshot of the command and the results of its execution. *Further, make sure to eliminate unneeded columns from the result set, to name your columns something user-friendly and human readable, and to format any prices as currencies.*

**Section Steps**

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).

1. A video reseller needs to know how many movies are available. Write a single query to fulfill this request.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.

# Evaluation

Your lab will be reviewed by your facilitator or instructor with the following criteria and grade breakdown.



Use the **Ask the Facilitators Discussion Forum** if you have any questions regarding how to approach this lab. Make sure to include your name in the filename and submit it in the *Assignments* section of the course.