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.

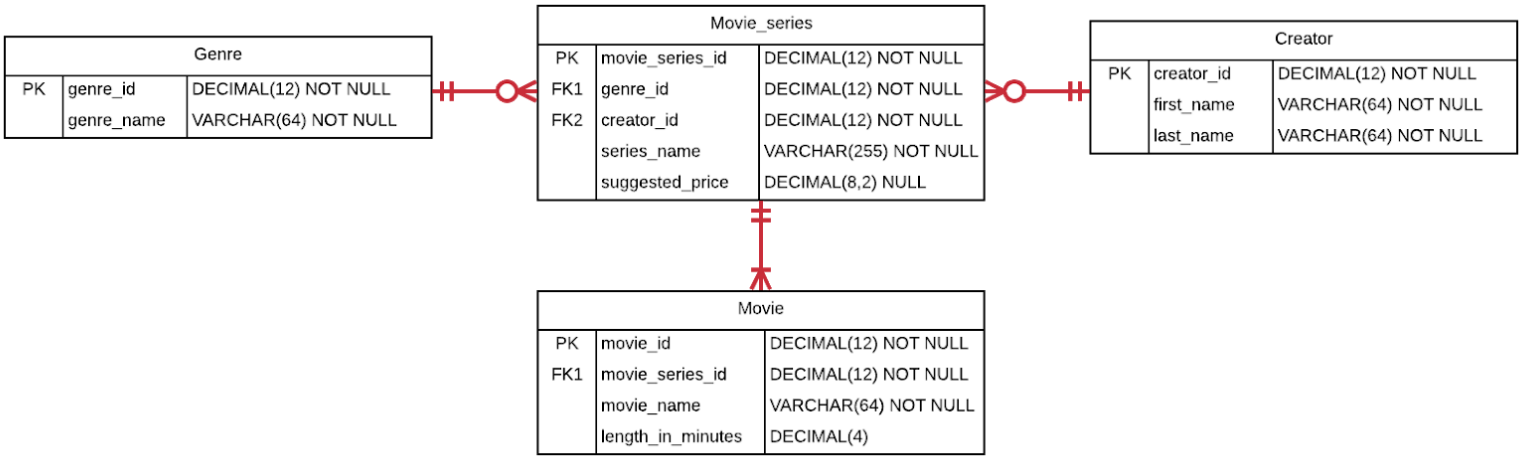
Other Reminders

- 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).

```sql
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_i
);
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2020-05-31T13:22:43.4876383-04:00
```

```sql
-- Genre
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 Genre(genre_id, genre_name)
values(4, 'Drama');


-- Creator
insert into Creator(creator_id,last_name,first_name)
values(1,'Lucas','George');

insert into Creator(creator_id,last_name,first_name)
```

Messages

```
(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

Completion time: 2020-05-31T20:08:18.5581978-04:00
```

```sql
-- Creator
insert into Creator(creator_id,last_name,first_name)
values(1,'Lucas','George');

insert into Creator(creator_id,last_name,first_name)
values(2,'Lasseter','John');

insert into Creator(creator_id,last_name,first_name)
values(3,'Tolkien','John');

insert into Creator(creator_id,last_name,first_name)
values(4,'Escandon','Phillip');


-- movie series
insert into Movie_series(movie_series_id, genre_id,creator_id, series_name, suggested_price)
values(1,1,1,'Start Wars',129.99);

insert into Movie_series(movie_series_id, genre_id,creator_id, series_name, suggested_price)
values(2,2,2,'Toy Story',22.13);

insert into Movie_series(movie_series_id, genre_id,creator_id, series_name, suggested_price)
```

essages

```
 row affected)

 row affected)

 row affected)

 row affected)

mpletion time: 2020-05-31T20:13:55.2871007-04:00
```

```sql
-- movie
insert into Movie(movie_id,movie_series_id,movie_name,length_in_minutes)
values(1,1,'Episode 1: The Phantom Menance',136);

insert into Movie(movie_id,movie_series_id,movie_name,length_in_minutes)
values(2,1,'Episode II: Attack of the Clones',142);

insert into Movie(movie_id,movie_series_id,movie_name,length_in_minutes)
values(3,1,'Episode III: Revenge of the Sith',140);

insert into Movie(movie_id,movie_series_id,movie_name,length_in_minutes)
values(4,1,'Episode IV: A New Hope',121);

insert into Movie(movie_id,movie_series_id,movie_name,length_in_minutes)
values(5,2,'Toy Story',121);

insert into Movie(movie_id,movie_series_id,movie_name,length_in_minutes)
values(6,2,'Toy Story 2',135);

insert into Movie(movie_id,movie_series_id,movie_name,length_in_minutes)
values(7,2,'Toy Story 3',148);

insert into Movie(movie_id,movie_series_id,movie_name,length_in_minutes)
values(8,3,'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(9,3,'The Lord of the Rings: The Two Towers',235);

insert into Movie(movie_id,movie_series_id,movie_name,length_in_minutes)
values(10,3,'The Lord of the Rings: The Return of the King',200);

insert into Movie(movie_id,movie_series_id,movie_name,length_in_minutes)
```

Messages

```
(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

Completion time: 2020-05-31T20:36:24.3733581-04:00
```

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

```sql
select count(*) as Movie_Count from Movie
```

Results | Messages

	Movie_Count
1	13

Verified

```
select *from Movie
```

table ErrorLogs.dbo.Movie

Results | Messages

	movie_id	movie_series_id	movie_name	length_in_minutes
1	1	1	Episode 1: The Phantom Menance	136
2	2	1	Episode II: Attack of the Clones	142
3	3	1	Episode III: Revenge of the Sith	140
4	4	1	Episode IV: A New Hope	121
5	5	2	Toy Story	121
6	6	2	Toy Story 2	135
7	7	2	Toy Story 3	148
8	8	3	The Lord of the Rings: The Fellowship of the Ring	228
9	9	3	The Lord of the Rings: The Two Towers	235
10	10	3	The Lord of the Rings: The Return of the King	200
11	11	3	Sporanos: The Jersey Shore	228
12	12	3	Sporanos: Canoli Heaven	205
13	13	3	Sporanos: Bada Bing Bada Boom	210

3. 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 most expensive and least expensive series
select min(suggested_price) as Minimum_Price from Movie_series
```

Results | Messages

	Minimum_Price
1	12.99

```
select max(suggested_price) as Maximum_Price from Movie_series
```

Results | Messages

	Maximum_Price
1	129.99

The Lord of the Rings series was treated differently because there are Null values associated with the Suggested_Price value.

4. 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.

```sql
SELECT
    series_name ,
    suggested_price,
    count(Movie.movie_series_id) as 'Movie Count'
FROM Movie_series
INNER JOIN Movie on Movie.movie_series_id = Movie_series.movie_series_id
GROUP BY
    Movie_series.series_name, suggested_price;
```

Results | Messages

series_name	suggested_price	Movie Count
Lord of the Rings	NULL	3
Sporanos	12.99	3
Toy Story	22.13	3
Start Wars	129.99	4

5. 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.

```sql
--- Found it
SELECT
    genre_name,
    count(genre_name) as 'Movie Count'
FROM
    Genre
JOIN Movie_series on Movie_series.genre_id = Genre.genre_id
JOIN Movie on Movie.movie_series_id = Movie_series.movie_series_id
GROUP BY
    genre_name
HAVING count(genre_name) >=7;
```

Results | Messages

	genre_name	Movie Count
1	Fantasy	7

6. 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.

```sql
--STEP 6
SELECT
    series_name,
    sum(Movie.length_in_minutes)/60 as 'Bingable Hours'
FROM
    Movie_series
JOIN
    Movie on Movie.movie_series_id = Movie_series.movie_series_id
GROUP BY
    series_name;
```

00 %

Results | Messages

	series_name	Bingable Hours
1	Lord of the Rings	11.050000
2	Sporanos	10.716666
3	Start Wars	8.983333
4	Toy Story	6.733333

7. 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.

```sql
-- STEP 7
SELECT
    last_name,
    first_name,
    count(Movie_series.genre_id) as 'Family Films Produced'
FROM
    Movie_series
JOIN
    Creator on Creator.creator_id = Movie_series.creator_id
JOIN
    Genre on Genre.genre_id = Movie_series.genre_id
GROUP BY
    Movie_series.genre_id, last_name,first_name
HAVING
    Movie_series.genre_id = 2
```

Results | Messages

	last_name	first_name	Family Films Produced
1	Lasseter	John	1

I can't quite understand this last SQL Call.

My template for this call was
SELECT
FROM
JOIN
GROUP BY
ORDER BY

but couldn't piece it together.

Evaluation

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

Criterion	A	B	C	D	F	Letter Grade
Correctness and Completeness of Results (70%)	All steps' results are entirely complete and correct	About ¾ of the steps' results are correct and complete	About half of the steps' results are correct and complete	About ¼ of the steps' results are correct and complete	Virtually none of the step's results are correct and complete	
Constitution of SQL and Explanations (30%)	Excellent use and integration of appropriate SQL constructs and supporting explanations	Good use and integration of appropriate SQL constructs and supporting explanations	Mediocre use and integration of appropriate SQL constructs and supporting explanations	Substandard use and integration of appropriate SQL constructs and supporting explanations	Virtually all SQL constructs and supporting explanations are unsuitable or improperly integrated	
					Assignment Grade:	#N/A
The resulting grade is calculated as a weighted average as listed using A+=100, A=96, A-=92, B+=88, B=85, B-=82 etc.						
To obtain an A grade for the course, your weighted average should be >=95, A- >=90, B+ >=87, B >= 82, B- >= 80 etc.						

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.