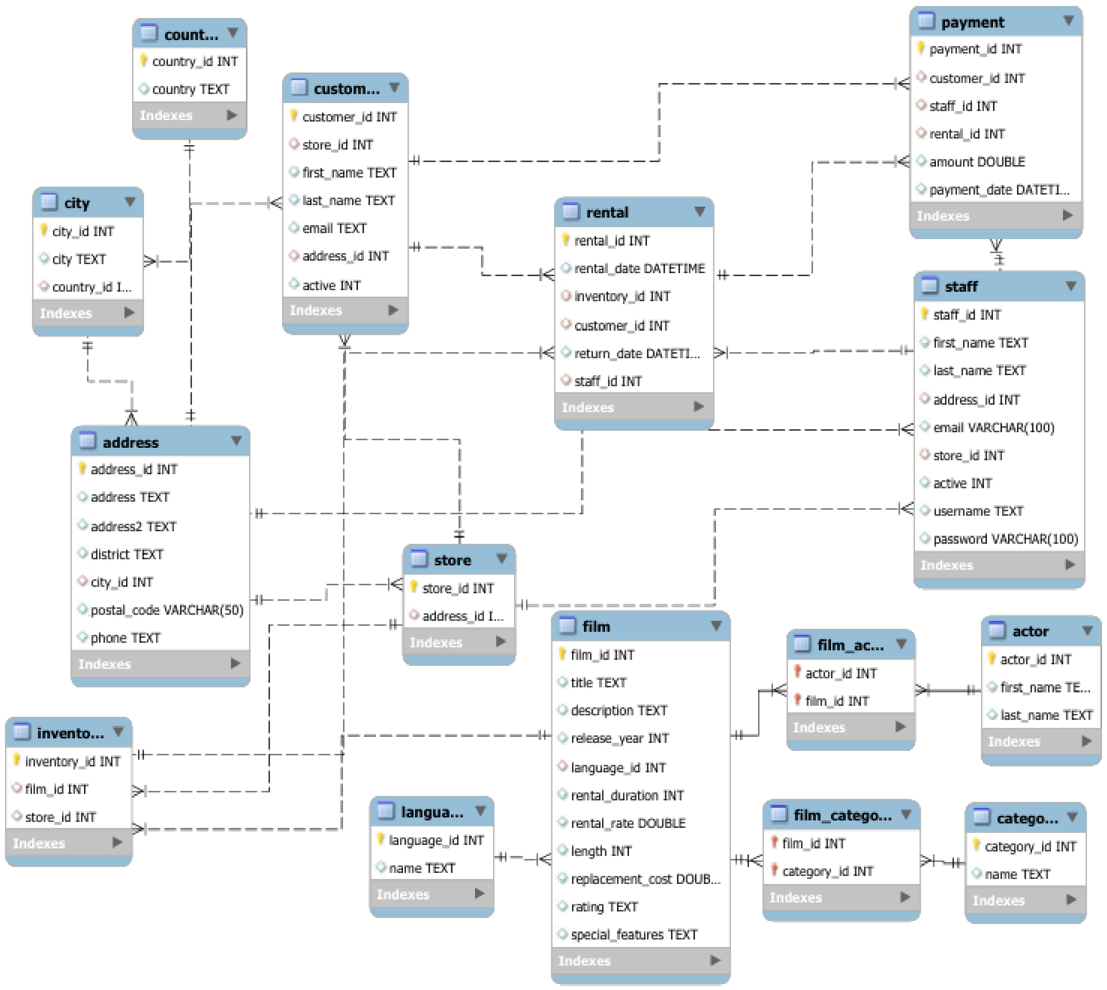
DB Assignment 4
Katrina Cwiertniewicz
11/4/2024

NO_ZERO_DATE to confirm date validity so the value is not zero.
Source [MySQL :: MySQL 8.4 Reference Manual :: B.3.4.2 Problems Using DATE Columns]

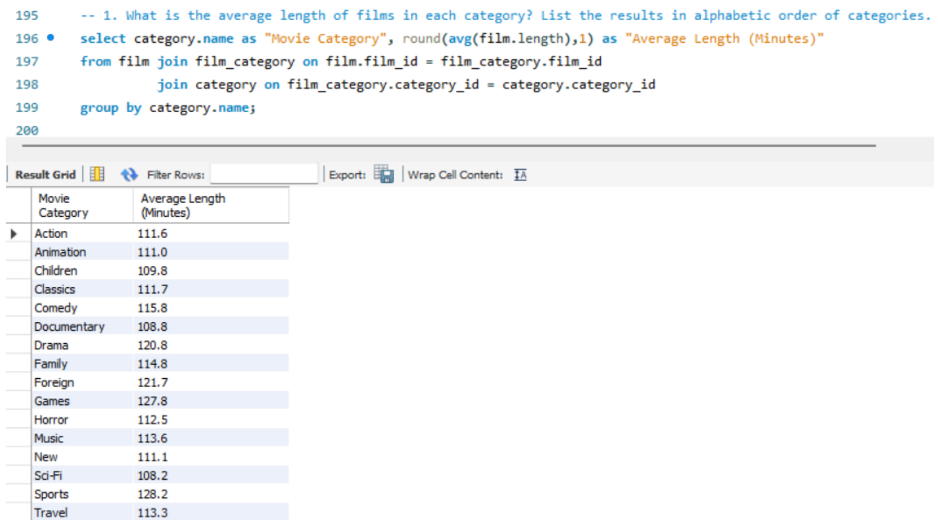
DATEDIFF() to find the difference between two dates.
Source:[MySQL :: MySQL 8.4 Reference Manual :: 14.7 Date and Time Functions]

ERD Diagram



1. What is the average length of films in each category? List the results in alphabetic order of categories.

The query selects category name and average film length rounded to one decimal point. It joins film, film_category, and category using the foreign keys film_id and category_id. It then groups the average film length by each category in alphabetical order.

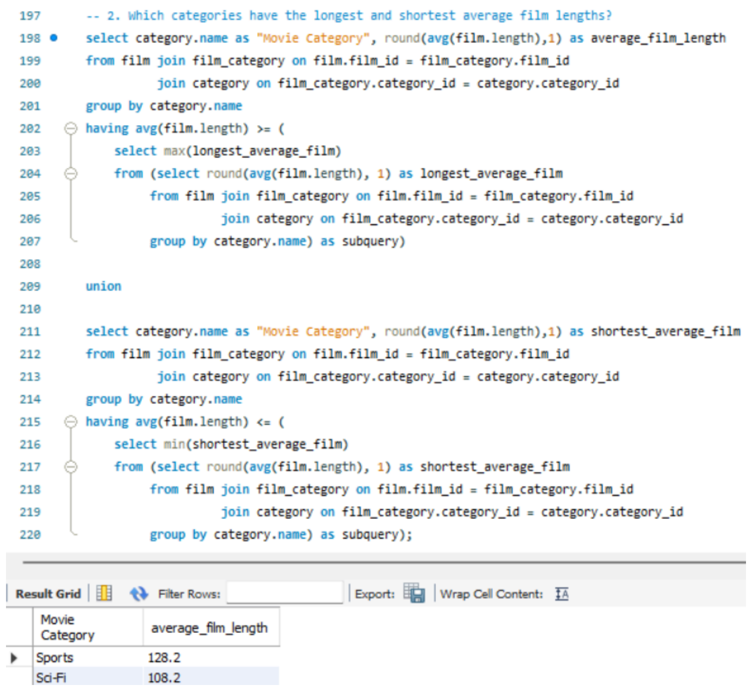


```sql
-- 1. What is the average length of films in each category? List the results in alphabetic order of categories.
select category.name as "Movie Category", round(avg(film.length),1) as "Average Length (Minutes)"
from film join film_category on film.film_id = film_category.film_id
          join category on film_category.category_id = category.category_id
group by category.name;
```

Movie Category	Average Length (Minutes)
Action	111.6
Animation	111.0
Children	109.8
Classics	111.7
Comedy	115.8
Documentary	108.8
Drama	120.8
Family	114.8
Foreign	121.7
Games	127.8
Horror	112.5
Music	113.6
New	111.1
Sci-Fi	108.2
Sports	128.2
Travel	113.3

2. Which categories have the longest and shortest average film lengths?

The query selects category name and average film length rounded to one decimal point. It joins film, film_category, and category by using the foreign keys film_id and category_id. It uses having and a subquery to find the longest and shortest length on average and grouping by category. These query results are combined using union.

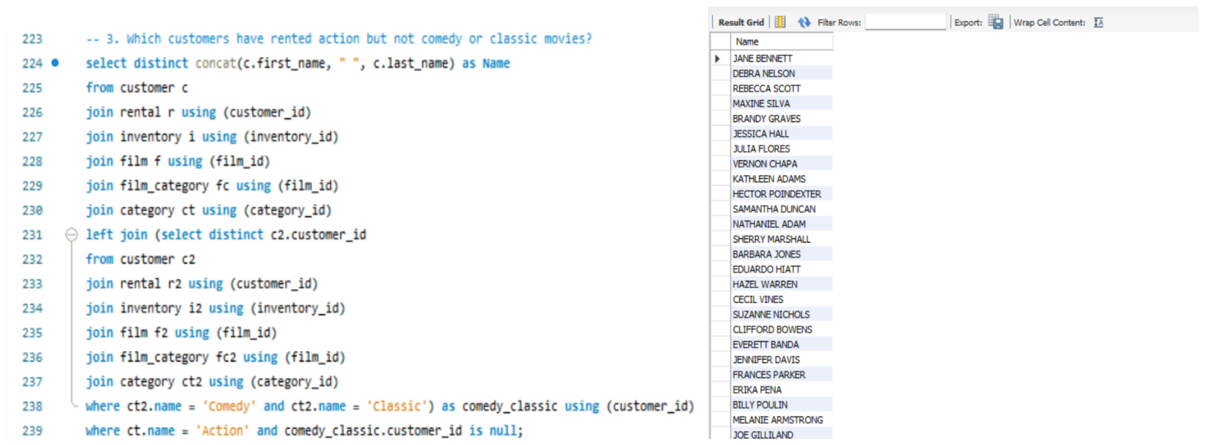


```sql
-- 2. Which categories have the longest and shortest average film lengths?
select category.name as "Movie Category", round(avg(film.length),1) as average_film_length
from film join film_category on film.film_id = film_category.film_id
          join category on film_category.category_id = category.category_id
group by category.name
having avg(film.length) >= (
    select max(longest_average_film)
    from (select round(avg(film.length), 1) as longest_average_film
        from film join film_category on film.film_id = film_category.film_id
                  join category on film_category.category_id = category.category_id
        group by category.name) as subquery)

union

select category.name as "Movie Category", round(avg(film.length),1) as shortest_average_film
from film join film_category on film.film_id = film_category.film_id
          join category on film_category.category_id = category.category_id
group by category.name
having avg(film.length) <= (
    select min(shortest_average_film)
    from (select round(avg(film.length), 1) as shortest_average_film
        from film join film_category on film.film_id = film_category.film_id
                  join category on film_category.category_id = category.category_id
        group by category.name) as subquery);
```

Movie Category	average_film_length
Sports	128.2
Sci-Fi	108.2

3. Which customers have rented action but not comedy or classic movies?

Shortened to alias' for this query because of the amount of joins. The query selects distinct customer name and joins customer, rental, inventory, film, film_category, and category by using the foreign keys customer_id, inventory_id, film_id, category_id, and customer_id. A subquery and left join are used to find customers who rented category name Action but not Comedy or Classic by using the subquery when comedy_classic.customer_id is null.



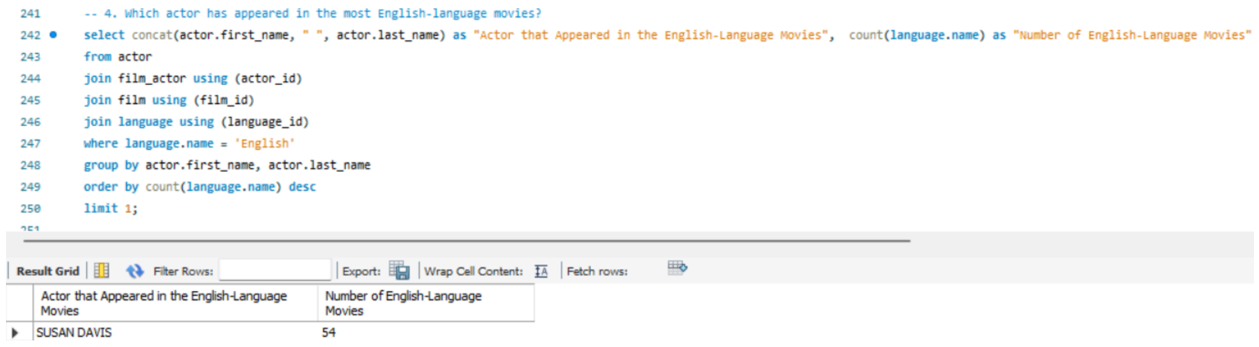
```sql
-- 3. Which customers have rented action but not comedy or classic movies?
select distinct concat(c.first_name, " ", c.last_name) as Name
from customer c
join rental r using (customer_id)
join inventory i using (inventory_id)
join film f using (film_id)
join film_category fc using (film_id)
join category ct using (category_id)
left join (select distinct c2.customer_id
from customer c2
join rental r2 using (customer_id)
join inventory i2 using (inventory_id)
join film f2 using (film_id)
join film_category fc2 using (film_id)
join category ct2 using (category_id)
where ct2.name = 'Comedy' and ct2.name = 'Classic') as comedy_classic using (customer_id)
where ct.name = 'Action' and comedy_classic.customer_id is null;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Name
JANE BENNETT
DEBRA NELSON
REBECCA SCOTT
MAXINE SILVA
BRANDY GRAVES
JESSICA HALL
JULIA FLORES
VERNON CHAPA
KATHLEEN ADAMS
HECTOR POINDEXTER
SAMANTHA DUNCAN
NATHANIEL ADAM
SHERRY MARSHALL
BARBARA JONES
EDUARDO HIATT
HAZEL WARREN
CECIL VINES
SUZANNE NICHOLS
CLIFFORD BOWENS
EVERETT BANDA
JENNIFER DAVIS
FRANCES PARKER
ERIKA PENA
BILLY POULIN
MELANIE ARMSTRONG
JOE GILLILAND

4. Which actor has appeared in the most English-language movies?

The query selects a combined first and last name using concat and a count of the language name. It joins actor, film_actor, and film using the foreign keys actor_id, film_id, and language_id. It then finds what movie language equals English. It groups by actor's first and last name and orders by highest count of English-language movies and limits the search by 1 to show the actor that has appeared in the most English-language movies.



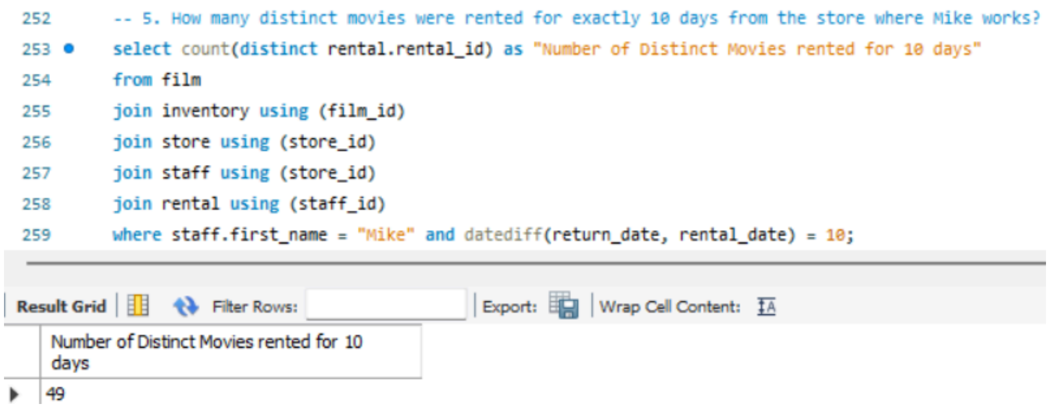
```sql
-- 4. Which actor has appeared in the most English-language movies?
select concat(actor.first_name, " ", actor.last_name) as "Actor that Appeared in the English-Language Movies",  count(language.name) as "Number of English-Language Movies"
from actor
join film_actor using (actor_id)
join film using (film_id)
join language using (language_id)
where language.name = 'English'
group by actor.first_name, actor.last_name
order by count(language.name) desc
limit 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

Actor that Appeared in the English-Language Movies	Number of English-Language Movies
SUSAN DAVIS	54

5. How many distinct movies were rented for exactly 10 days from the store where Mike works?

The query selects a distinct count of rental_ids. It joins film, inventory, store, staff and rental using the foreign keys film_id, store_id, and staff_id. It filters the results by searching for the staff named Mike and the difference between return and rental dates by using datediff to find the difference between the dates by the amount of 10 days.

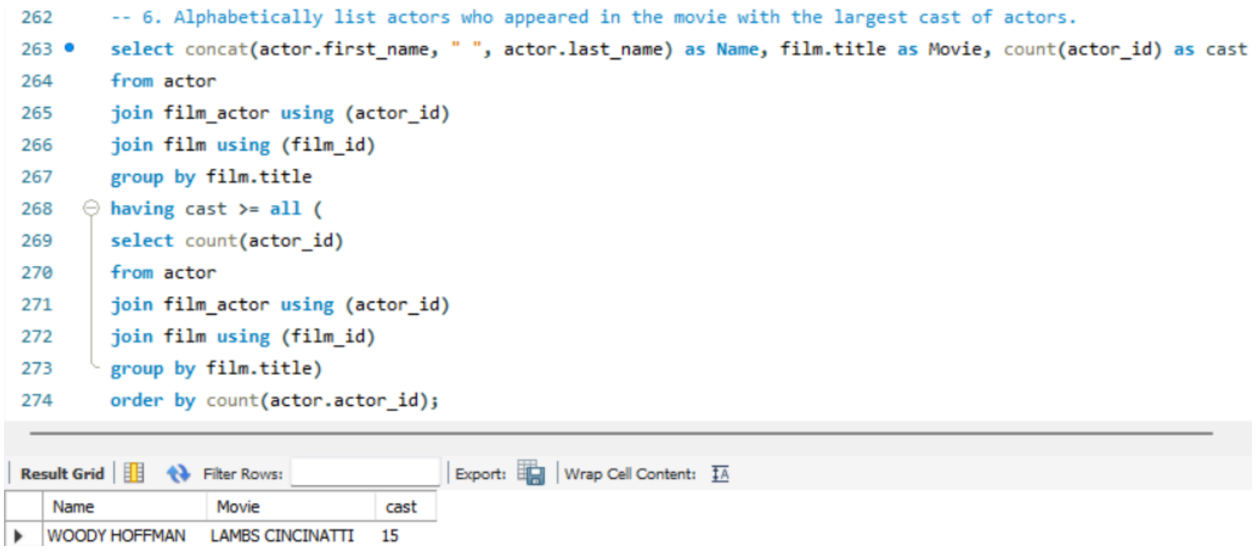


```sql
-- 5. How many distinct movies were rented for exactly 10 days from the store where Mike works?
select count(distinct rental.rental_id) as "Number of Distinct Movies rented for 10 days"
from film
join inventory using (film_id)
join store using (store_id)
join staff using (store_id)
join rental using (staff_id)
where staff.first_name = "Mike" and datediff(return_date, rental_date) = 10;
```

```
Number of Distinct Movies rented for 10
days
49
```

6. Alphabetically list actors who appeared in the movie with the largest cast of actors.

The query selects Name concat with first and last name, film title as movie, and count of actors as cast. It joins actor, film_actor, and film using the foreign keys actor_id and film_id. It groups by film title and selects the cast with the largest number of cast members. I was able to create a subquery that would select the movie with the largest cast but was not able to output the list of all the actors in the movie.



```sql
-- 6. Alphabetically list actors who appeared in the movie with the largest cast of actors.
select concat(actor.first_name, " ", actor.last_name) as Name, film.title as Movie, count(actor_id) as cast
from actor
join film_actor using (actor_id)
join film using (film_id)
group by film.title
having cast >= all (
select count(actor_id)
from actor
join film_actor using (actor_id)
join film using (film_id)
group by film.title)
order by count(actor.actor_id);
```

```
Name            Movie              cast
WOODY HOFFMAN   LAMBS CINCINATTI   15
```