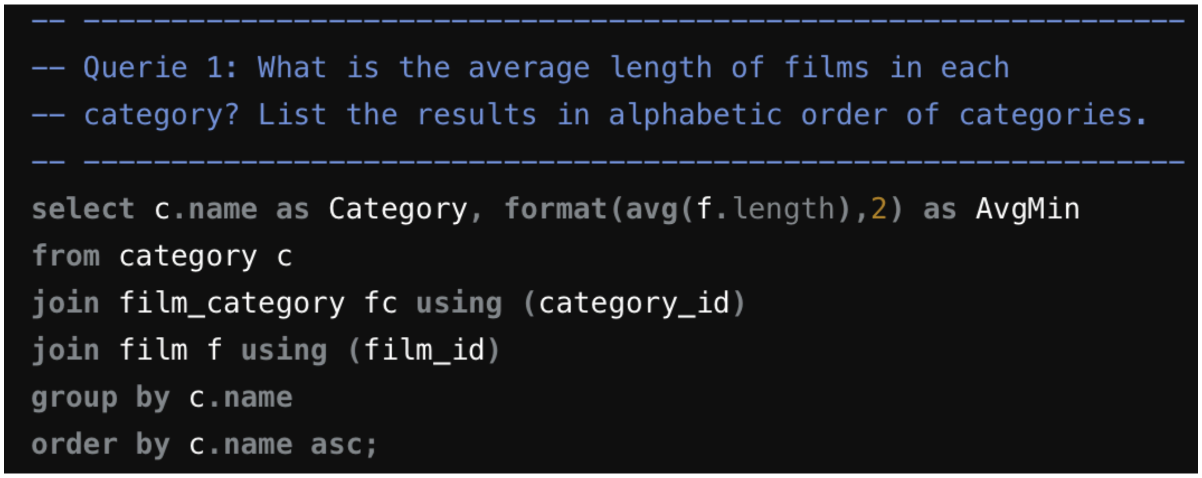
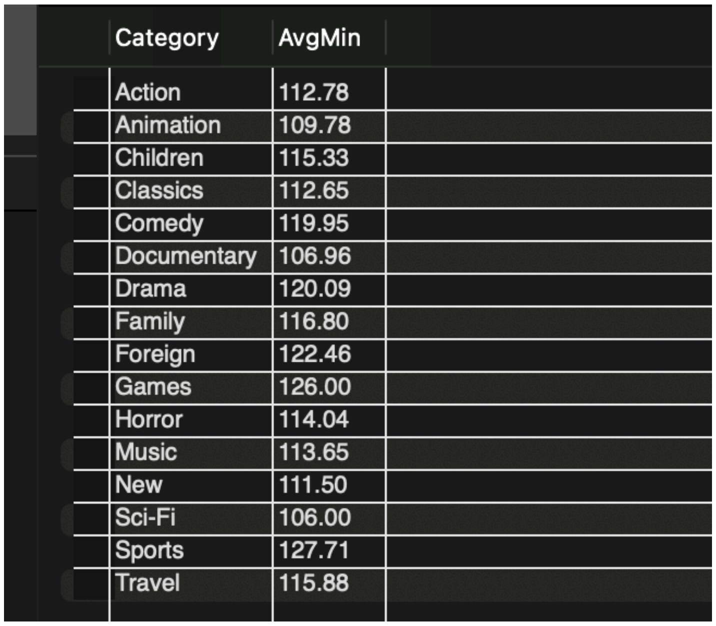
DB Assignment 4
Brittany Klose
10/22/24

<u>**SQL queries:**</u>

1. **What is the average length of films in each category? List the results in alphabetic order of categories.**
   a. To find this query I joined together the category, film, and film category table. Then I used the average operator on film length to calculate the average film time for each category. I formatted the answer to 2 decimal places for ease of readability. Lastly I order the results by category name in ascending order to get the films in alphabetical order.
   b. Query & Results

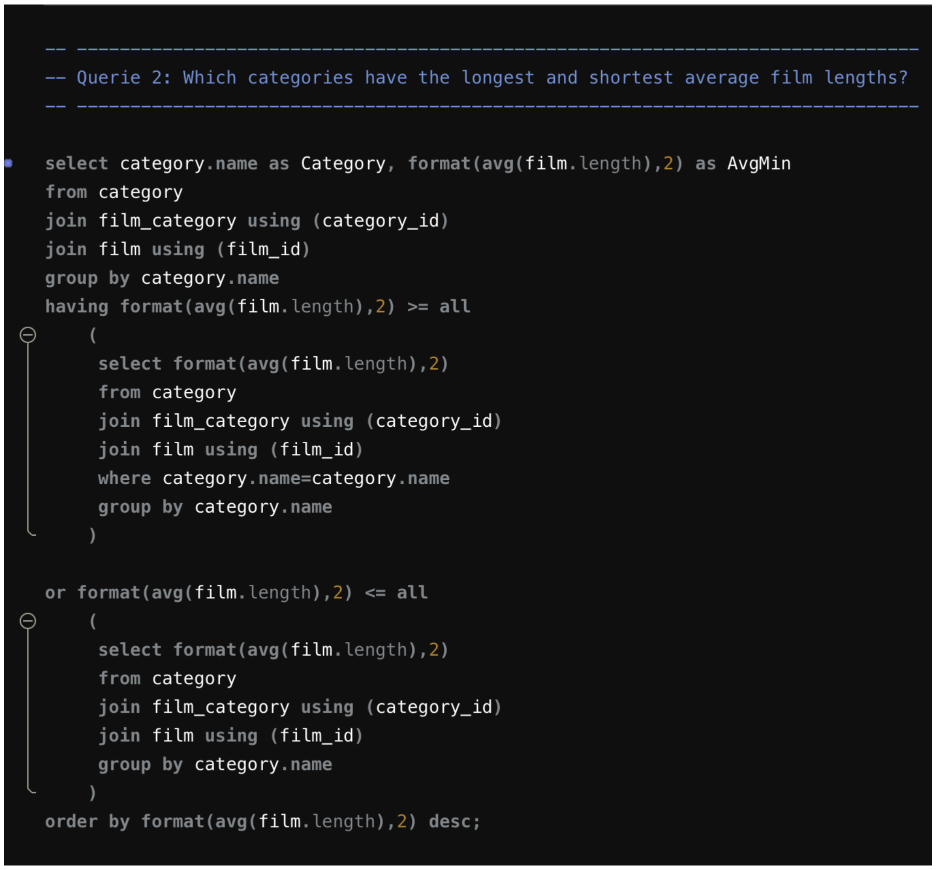


```
-- ----------------------------------------------------------------
-- Querie 1: What is the average length of films in each
-- category? List the results in alphabetic order of categories.
-- ----------------------------------------------------------------
select c.name as Category, format(avg(f.length),2) as AvgMin
from category c
join film_category fc using (category_id)
join film f using (film_id)
group by c.name
order by c.name asc;
```

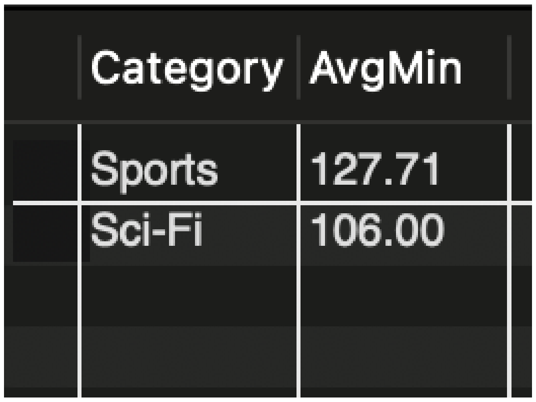


Category	AvgMin
Action	112.78
Animation	109.78
Children	115.33
Classics	112.65
Comedy	119.95
Documentary	106.96
Drama	120.09
Family	116.80
Foreign	122.46
Games	126.00
Horror	114.04
Music	113.65
New	111.50
Sci-Fi	106.00
Sports	127.71
Travel	115.88

2. **Which categories have the longest and shortest average film lengths?**
    a. To solve this query I used the query I created in the problem above. I then added in two different cases with ‘having’ to first select the category with an average time greater than all the others, i.e. with the longest. And then I used ‘or’ to write another statement to select the category with an average film length that was less than all the others i.e. the shortest. I ordered the results in descending order to put the longest at the top and shortest below to follow the order of the prompt. Additionally, included in my SQL file I wrote a side query to print out the script from the first query and order it instead by average time, longest to shortest. This provided a quick check for me that my results were correct since there were only 16 categories. If this were for something else with 100s of categories I would have done the query twice, once as ascending and then as descending with a limit of 1 under each to verify my results if needed.
    b. Query & Results



```
-- ---------------------------------------------------------------------------------
-- Querie 2: Which categories have the longest and shortest average film lengths?
-- ---------------------------------------------------------------------------------

select category.name as Category, format(avg(film.length),2) as AvgMin
from category
join film_category using (category_id)
join film using (film_id)
group by category.name
having format(avg(film.length),2) >= all
    (
     select format(avg(film.length),2)
     from category
     join film_category using (category_id)
     join film using (film_id)
     where category.name=category.name
     group by category.name
    )

or format(avg(film.length),2) <= all
    (
     select format(avg(film.length),2)
     from category
     join film_category using (category_id)
     join film using (film_id)
     group by category.name
    )
order by format(avg(film.length),2) desc;
```

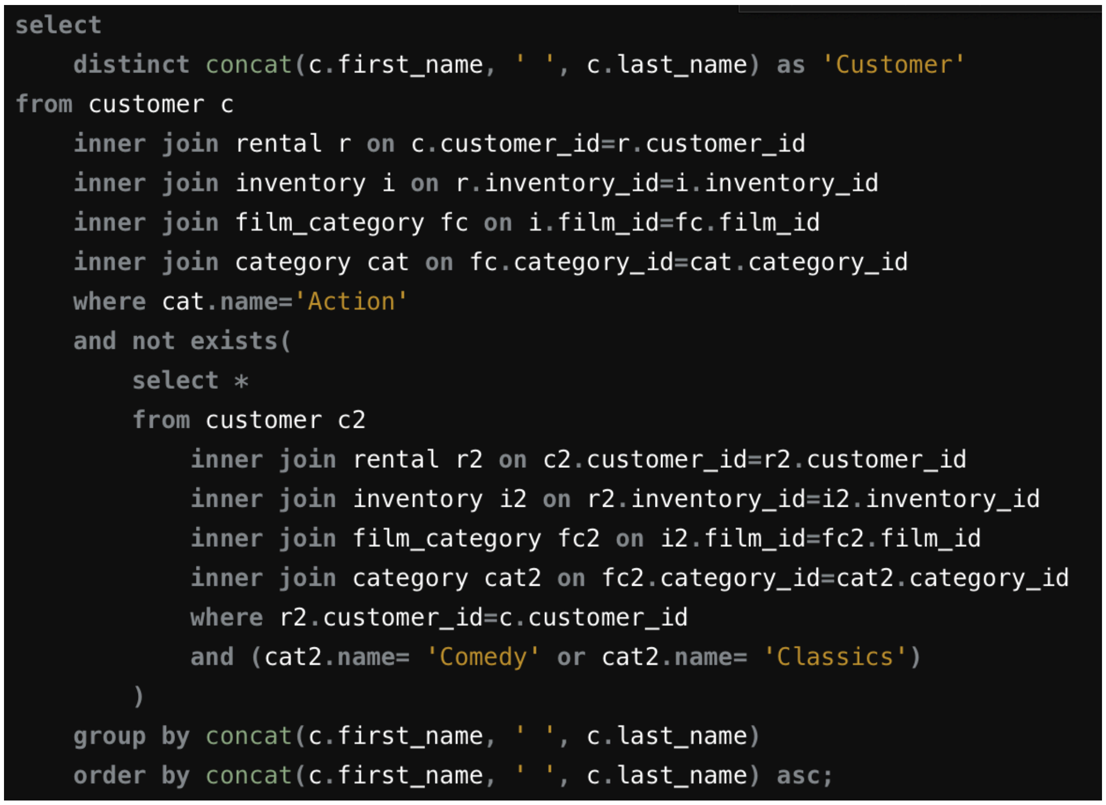


Category	AvgMin
Sports	127.71
Sci-Fi	106.00

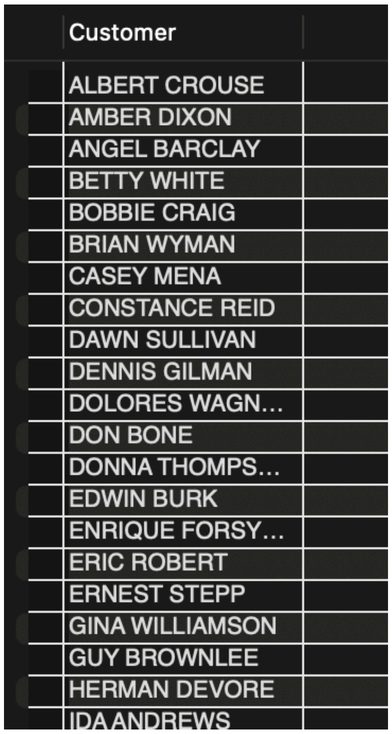
3. **Which customers have rented action but not comedy or classic movies?**

a. This query proved to be the most challenging for me as the keyword ‘Except’ is not compatible with my version of MySQL. Instead I used ‘Not Exists’. To start I selected customer names and concatenated first and last names together for personal preference. Then I joined the rental, inventory, film_category, and category tables together. I used a where subquery to locate where the category names for that customer included action but not comedy and classics, i.e. where those two categories didn’t exist. I then ordered the names alphabetically for a nice organized list to be outputted. After I was done I wanted a way to somewhat check my answers and tried to combine the group_concat feature I used in a later problem to print out the same names and also include their rented categories. To do so I had to write a longer query and use ‘find_in_set’ instead of where=action, and ‘not in’ instead of ‘where not exists’. Overall I would say the extra language to include something I didn’t necessarily need proved to be inefficient, but was helpful to figure out to help grow my skills.

b. Query & Results V1:

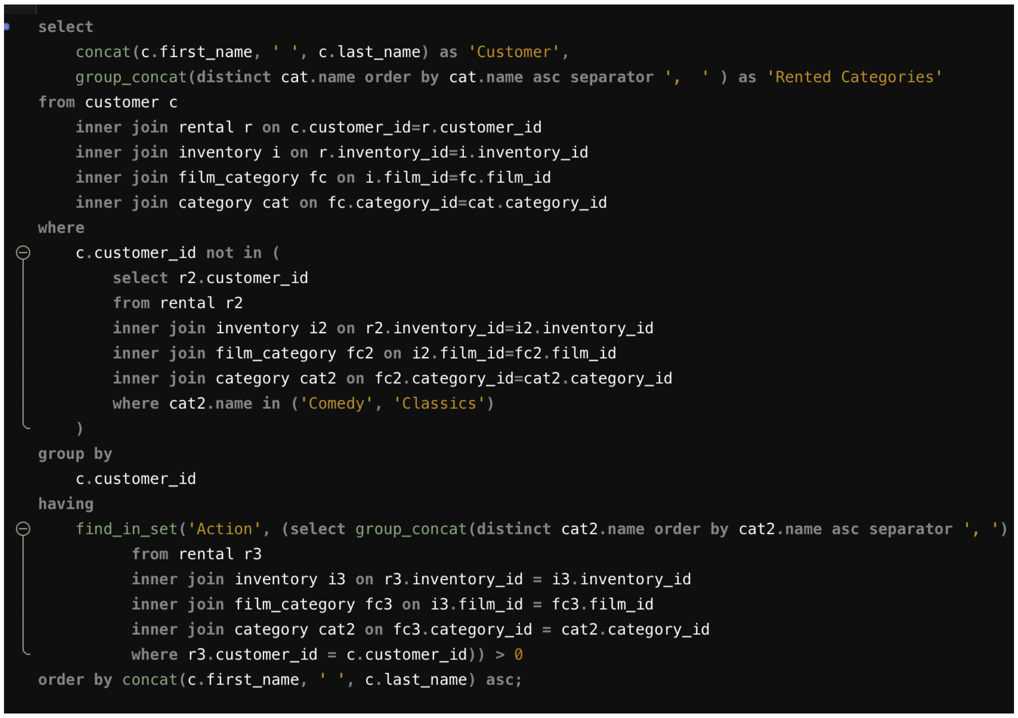


```
select
    distinct concat(c.first_name, ' ', c.last_name) as 'Customer'
from customer c
    inner join rental r on c.customer_id=r.customer_id
    inner join inventory i on r.inventory_id=i.inventory_id
    inner join film_category fc on i.film_id=fc.film_id
    inner join category cat on fc.category_id=cat.category_id
    where cat.name='Action'
    and not exists(
        select *
        from customer c2
            inner join rental r2 on c2.customer_id=r2.customer_id
            inner join inventory i2 on r2.inventory_id=i2.inventory_id
            inner join film_category fc2 on i2.film_id=fc2.film_id
            inner join category cat2 on fc2.category_id=cat2.category_id
            where r2.customer_id=c.customer_id
            and (cat2.name= 'Comedy' or cat2.name= 'Classics')
        )
    group by concat(c.first_name, ' ', c.last_name)
    order by concat(c.first_name, ' ', c.last_name) asc;
```

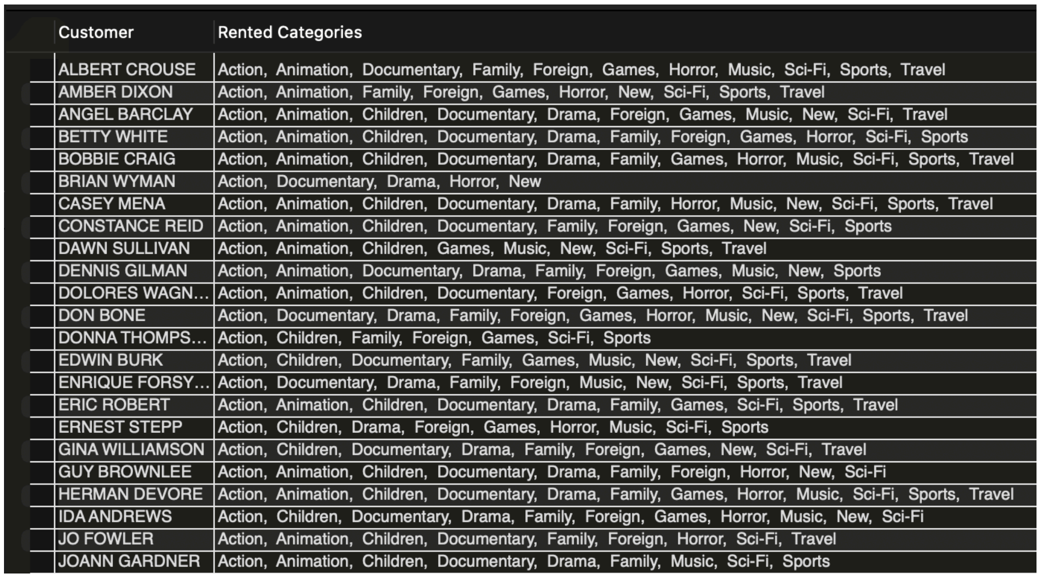


```
Customer
ALBERT CROUSE
AMBER DIXON
ANGEL BARCLAY
BETTY WHITE
BOBBIE CRAIG
BRIAN WYMAN
CASEY MENA
CONSTANCE REID
DAWN SULLIVAN
DENNIS GILMAN
DOLORES WAGN...
DON BONE
DONNA THOMPS...
EDWIN BURK
ENRIQUE FORSY...
ERIC ROBERT
ERNEST STEPP
GINA WILLIAMSON
GUY BROWNLEE
HERMAN DEVORE
IDA ANDREWS
```

c. Query & Results V2:



```sql
select
    concat(c.first_name, ' ', c.last_name) as 'Customer',
    group_concat(distinct cat.name order by cat.name asc separator ',  ' ) as 'Rented Categories'
from customer c
    inner join rental r on c.customer_id=r.customer_id
    inner join inventory i on r.inventory_id=i.inventory_id
    inner join film_category fc on i.film_id=fc.film_id
    inner join category cat on fc.category_id=cat.category_id
where
    c.customer_id not in (
        select r2.customer_id
        from rental r2
        inner join inventory i2 on r2.inventory_id=i2.inventory_id
        inner join film_category fc2 on i2.film_id=fc2.film_id
        inner join category cat2 on fc2.category_id=cat2.category_id
        where cat2.name in ('Comedy', 'Classics')
    )
group by
    c.customer_id
having
    find_in_set('Action', (select group_concat(distinct cat2.name order by cat2.name asc separator ', ')
          from rental r3
          inner join inventory i3 on r3.inventory_id = i3.inventory_id
          inner join film_category fc3 on i3.film_id = fc3.film_id
          inner join category cat2 on fc3.category_id = cat2.category_id
          where r3.customer_id = c.customer_id)) > 0
order by concat(c.first_name, ' ', c.last_name) asc;
```

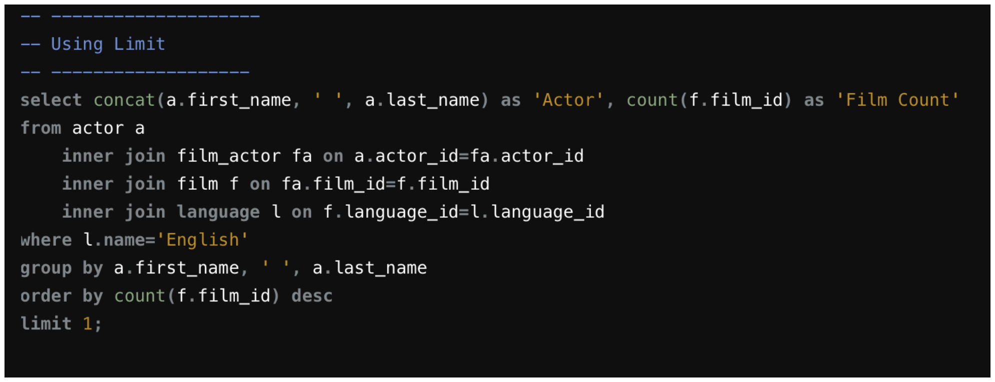


Customer	Rented Categories
ALBERT CROUSE	Action, Animation, Documentary, Family, Foreign, Games, Horror, Music, Sci-Fi, Sports, Travel
AMBER DIXON	Action, Animation, Family, Foreign, Games, Horror, New, Sci-Fi, Sports, Travel
ANGEL BARCLAY	Action, Animation, Children, Documentary, Drama, Foreign, Games, Music, New, Sci-Fi, Travel
BETTY WHITE	Action, Animation, Children, Documentary, Drama, Family, Foreign, Games, Horror, Sci-Fi, Sports
BOBBIE CRAIG	Action, Animation, Children, Documentary, Drama, Family, Games, Horror, Music, Sci-Fi, Sports, Travel
BRIAN WYMAN	Action, Documentary, Drama, Horror, New
CASEY MENA	Action, Animation, Children, Documentary, Drama, Family, Horror, Music, New, Sci-Fi, Sports, Travel
CONSTANCE REID	Action, Animation, Children, Documentary, Family, Foreign, Games, New, Sci-Fi, Sports
DAWN SULLIVAN	Action, Animation, Children, Games, Music, New, Sci-Fi, Sports, Travel
DENNIS GILMAN	Action, Animation, Documentary, Drama, Family, Foreign, Games, Music, New, Sports
DOLORES WAGN…	Action, Animation, Children, Documentary, Foreign, Games, Horror, Sci-Fi, Sports, Travel
DON BONE	Action, Documentary, Drama, Family, Foreign, Games, Horror, Music, New, Sci-Fi, Sports, Travel
DONNA THOMPS…	Action, Children, Family, Foreign, Games, Sci-Fi, Sports
EDWIN BURK	Action, Children, Documentary, Family, Games, Music, New, Sci-Fi, Sports, Travel
ENRIQUE FORSY…	Action, Documentary, Drama, Family, Foreign, Music, New, Sci-Fi, Sports, Travel
ERIC ROBERT	Action, Animation, Children, Documentary, Drama, Family, Games, Sci-Fi, Sports, Travel
ERNEST STEPP	Action, Children, Drama, Foreign, Games, Horror, Music, Sci-Fi, Sports
GINA WILLIAMSON	Action, Children, Documentary, Drama, Family, Foreign, Games, New, Sci-Fi, Travel
GUY BROWNLEE	Action, Animation, Children, Documentary, Drama, Family, Foreign, Horror, New, Sci-Fi
HERMAN DEVORE	Action, Animation, Children, Documentary, Drama, Family, Games, Horror, Music, Sci-Fi, Sports, Travel
IDA ANDREWS	Action, Children, Documentary, Drama, Family, Foreign, Games, Horror, Music, New, Sci-Fi
JO FOWLER	Action, Animation, Children, Documentary, Family, Foreign, Horror, Sci-Fi, Travel
JOANN GARDNER	Action, Animation, Children, Documentary, Drama, Family, Music, Sci-Fi, Sports

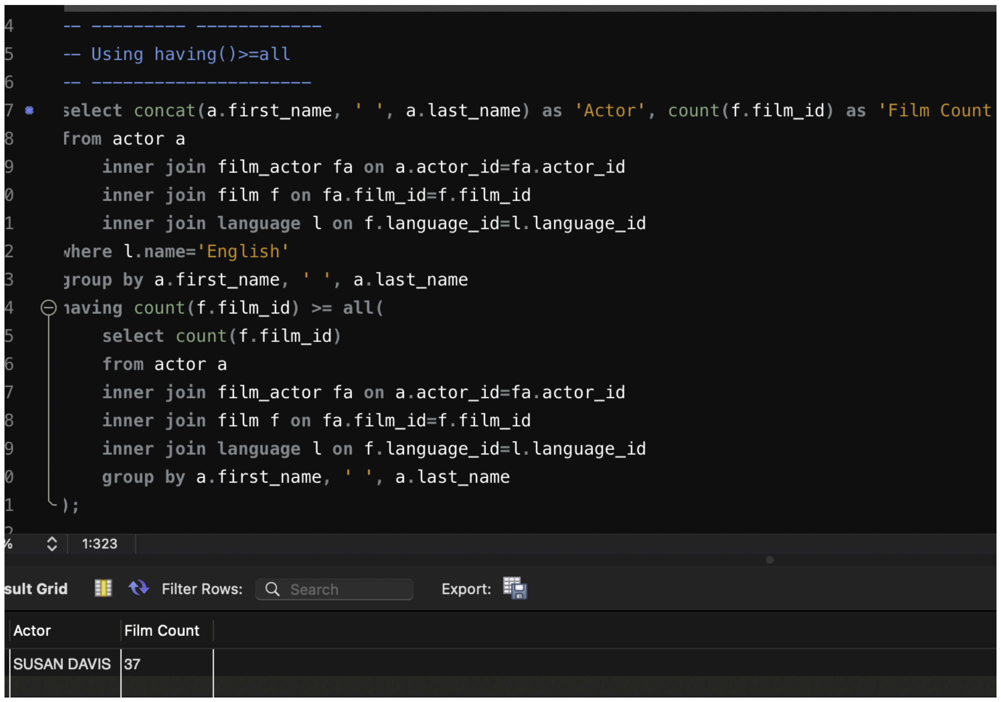
4. **Which actor has appeared in the most English-language movies?**
    a. I began this query by concatenating the actor's first and last names together as one Actor name mostly for ease. I then created a count on film IDs to track the number of movies each actor appeared in. I used inner join to connect film, film_actor, and language to be able to see which films

each actor was in and to then narrow the search down to only movies with English as the language. In this case it was easier to order results by highest film count to lowest with a limit of 1. However I included the same query using 'having () >= all' in the case that multiple actors were tied for the greatest number. It was not the case here, but it's always good practice to check your answers.

b. Query & Results



```
-- --------------------
-- Using Limit
-- -------------------
select concat(a.first_name, ' ', a.last_name) as 'Actor', count(f.film_id) as 'Film Count'
from actor a
    inner join film_actor fa on a.actor_id=fa.actor_id
    inner join film f on fa.film_id=f.film_id
    inner join language l on f.language_id=l.language_id
where l.name='English'
group by a.first_name, ' ', a.last_name
order by count(f.film_id) desc
limit 1;
```



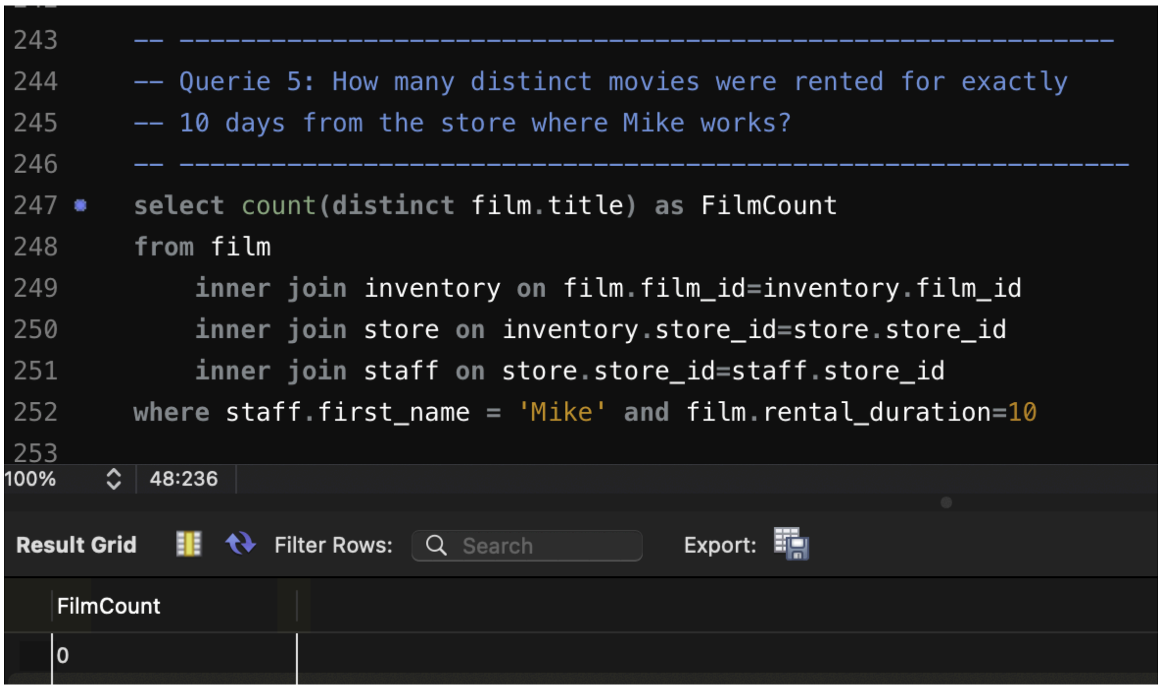
```
-- ---------- ------------
-- Using having()>=all
-- ---------------------
select concat(a.first_name, ' ', a.last_name) as 'Actor', count(f.film_id) as 'Film Count
from actor a
    inner join film_actor fa on a.actor_id=fa.actor_id
    inner join film f on fa.film_id=f.film_id
    inner join language l on f.language_id=l.language_id
where l.name='English'
group by a.first_name, ' ', a.last_name
having count(f.film_id) >= all(
    select count(f.film_id)
    from actor a
    inner join film_actor fa on a.actor_id=fa.actor_id
    inner join film f on fa.film_id=f.film_id
    inner join language l on f.language_id=l.language_id
    group by a.first_name, ' ', a.last_name
);
```

```
1:323

sult Grid   Filter Rows:  Search   Export:

Actor        Film Count
SUSAN DAVIS  37
```

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

a. This query was relatively simple. I created a count for film titles and used the distinct keyword in front to ensure the count didn't include the same movies more than once. Then I established inner joins to connect tables: inventory, store, film, and staff. Finally I narrowed the count down using ‘where’ to select the count just where the rental duration was 10 days and Mike was the staff member. There were no cases where a film’s rental duration was 10 days alone so the count was 0. In this case there wasn’t a need to check my results since there was a constraint given that rental duration was 2-8. However I included additional checks in my code that can be seen in my SQL file for good practice in case there wasn’t a constraint or if the prompt asked for a number within the domain. One check simply selected all distinct rental durations in ascending order so I could see all the numbers and make sure the number in question wasn’t there. This was only efficient though because the list was short. Lastly, I did another check using return date and rental date in case that differed from rental duration. In the end though the answer was still 0.

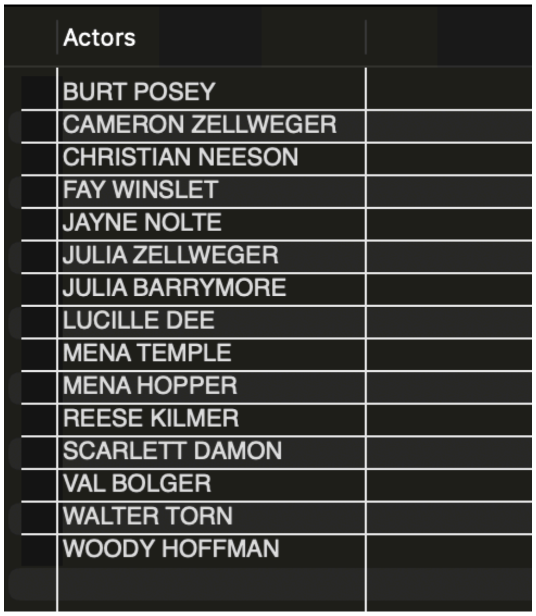
b. Query & Results



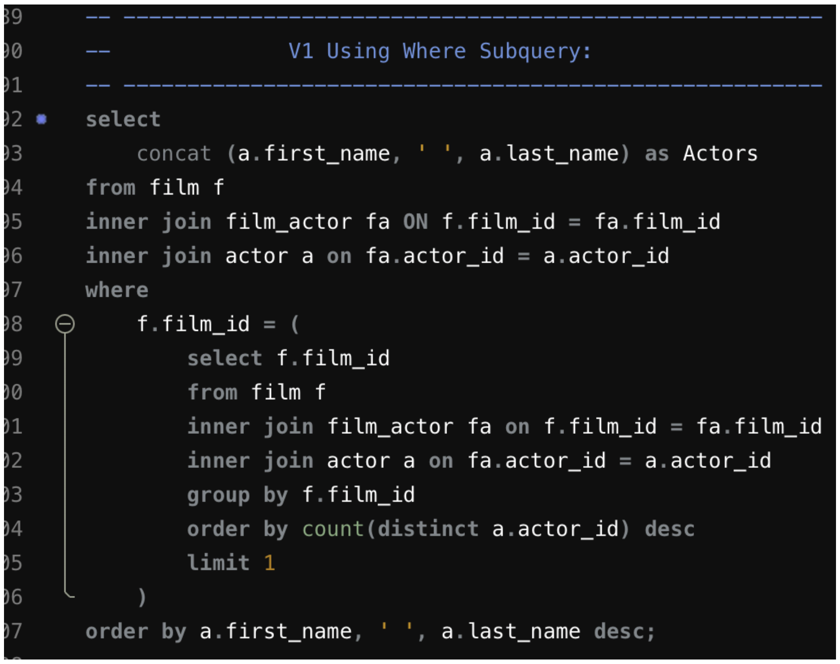
```
-- ---------------------------------------------------------------
-- Querie 5: How many distinct movies were rented for exactly
-- 10 days from the store where Mike works?
-- ----------------------------------------------------------------
select count(distinct film.title) as FilmCount
from film
    inner join inventory on film.film_id=inventory.film_id
    inner join store on inventory.store_id=store.store_id
    inner join staff on store.store_id=staff.store_id
where staff.first_name = 'Mike' and film.rental_duration=10
```

```
FilmCount
0
```

6. **Alphabetically list actors who appeared in the movie with the largest cast of actors.**
   a. For this query I tried two different approaches. One being using a 'where' subquery and the other using group-concat. In both queries I began by seeing I had to join film_actor, actor, and film. In the first query I selected the names of the actors and used a subquery with 'where' to select the list of actors from the subquery which found the movie with the greatest cast. To find the movie with the biggest cast I made a count in the subquery to find the number of distinct actors in each movie in descending order and used limit 1 to only select the top row. I then ordered the results in alphabetical order. It went by first name since I used a concat to combine first and last names. The next version of the query was very similar in language minus the presence of the subquery. I also adapted my selection to include the film title, the cast count, and the list of actors. To have the full list of actor names in one column I used group_concat and ordered it by first name since that was the order used in the previous query. I liked that this version allowed me to see additional information and all in one row, such as in the case where I wanted to see the cast and count for all the movies. However the prior query definitely is much better suited for viewing the actual cast names, especially in the case where the list is much longer than 15.
   b. Query & Results 1: Using Where Subquery

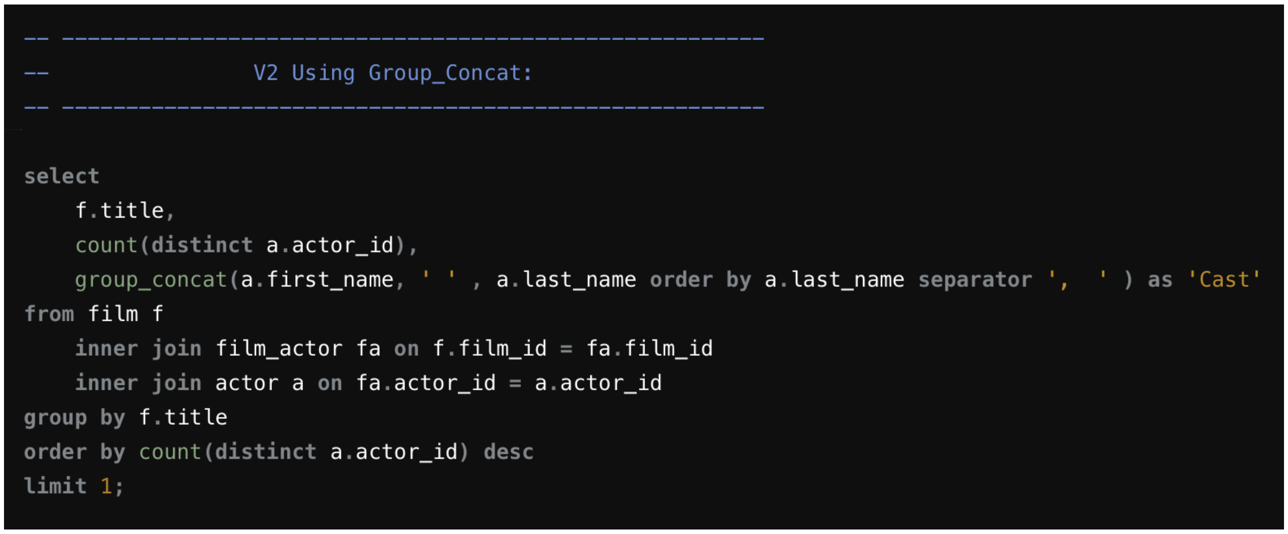


Actors
BURT POSEY
CAMERON ZELLWEGER
CHRISTIAN NEESON
FAY WINSLET
JAYNE NOLTE
JULIA ZELLWEGER
JULIA BARRYMORE
LUCILLE DEE
MENA TEMPLE
MENA HOPPER
REESE KILMER
SCARLETT DAMON
VAL BOLGER
WALTER TORN
WOODY HOFFMAN

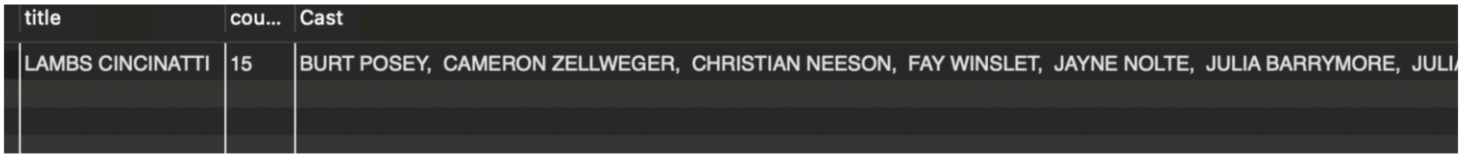


```
-- -----------------------------------------------------------
--              V1 Using Where Subquery:
-- -----------------------------------------------------------
select
    concat (a.first_name, ' ', a.last_name) as Actors
from film f
inner join film_actor fa ON f.film_id = fa.film_id
inner join actor a on fa.actor_id = a.actor_id
where
    f.film_id = (
        select f.film_id
        from film f
        inner join film_actor fa on f.film_id = fa.film_id
        inner join actor a on fa.actor_id = a.actor_id
        group by f.film_id
        order by count(distinct a.actor_id) desc
        limit 1
    )
order by a.first_name, ' ', a.last_name desc;
```

c. Query & Results 2: Using Group_concat

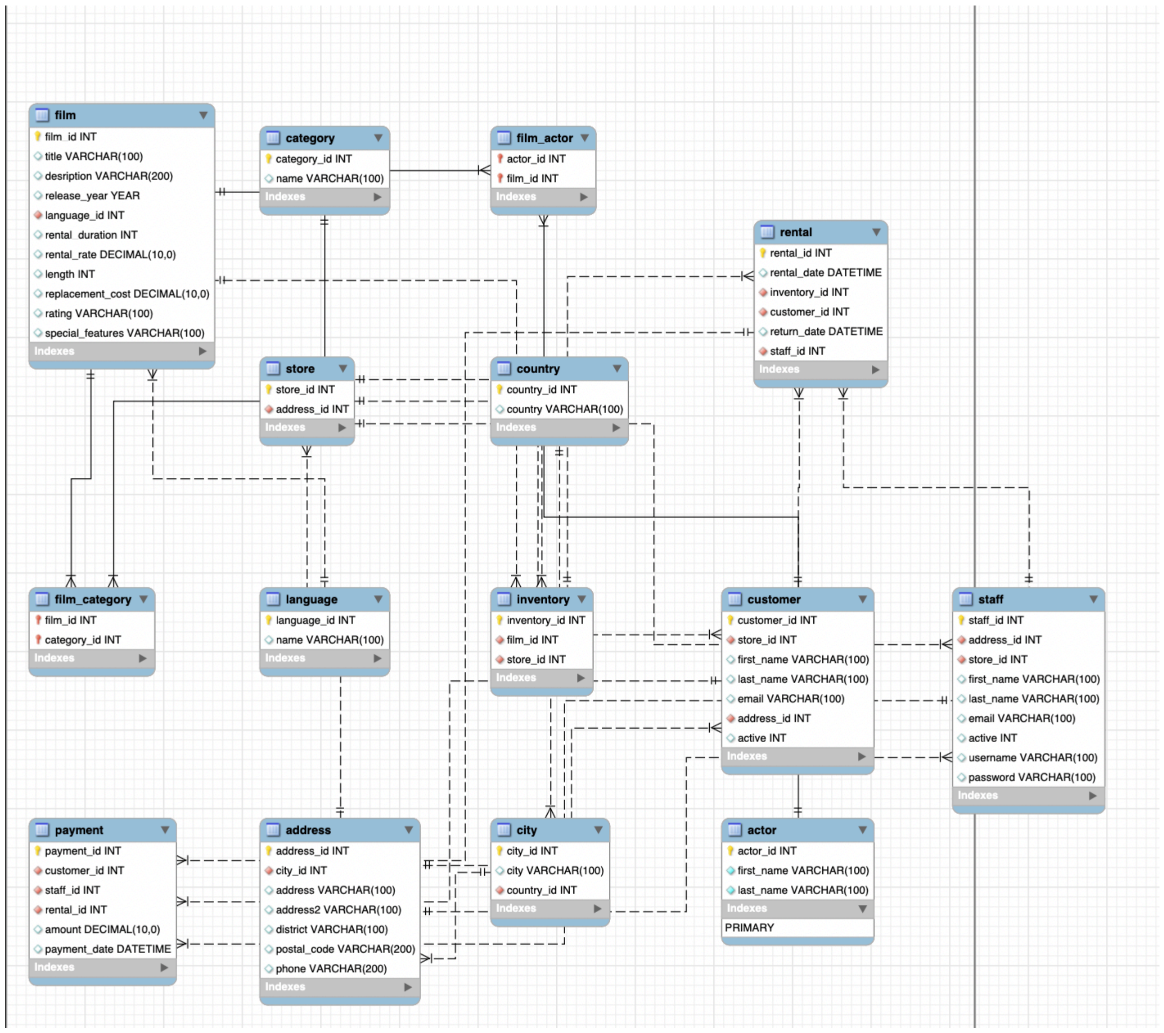


```
-- -------------------------------------------------------
--                V2 Using Group_Concat:
-- -------------------------------------------------------

select
    f.title,
    count(distinct a.actor_id),
    group_concat(a.first_name, ' ' , a.last_name order by a.last_name separator ',  ' ) as 'Cast'
from film f
    inner join film_actor fa on f.film_id = fa.film_id
    inner join actor a on fa.actor_id = a.actor_id
group by f.title
order by count(distinct a.actor_id) desc
limit 1;
```



title	cou...	Cast
LAMBS CINCINATTI	15	BURT POSEY, CAMERON ZELLWEGER, CHRISTIAN NEESON, FAY WINSLET, JAYNE NOLTE, JULIA BARRYMORE, JULI

7. **ERD diagram**
    a. Below is MySQL Workbench generated based on the tables I created in my schema.


film
film_id INT
title VARCHAR(100)
desription VARCHAR(200)
release_year YEAR
language_id INT
rental_duration INT
rental_rate DECIMAL(10,0)
length INT
replacement_cost DECIMAL(10,0)
rating VARCHAR(100)
special_features VARCHAR(100)
Indexes
category
category_id INT
name VARCHAR(100)
Indexes
film_actor
actor_id INT
film_id INT
Indexes
rental
rental_id INT
rental_date DATETIME
inventory_id INT
customer_id INT
return_date DATETIME
staff_id INT
Indexes
store
store_id INT
address_id INT
Indexes
country
country_id INT
country VARCHAR(100)
Indexes
film_category
film_id INT
category_id INT
Indexes
language
language_id INT
name VARCHAR(100)
Indexes
inventory
inventory_id INT
film_id INT
store_id INT
Indexes
customer
customer_id INT
store_id INT
first_name VARCHAR(100)
last_name VARCHAR(100)
email VARCHAR(100)
address_id INT
active INT
Indexes
staff
staff_id INT
address_id INT
store_id INT
first_name VARCHAR(100)
last_name VARCHAR(100)
email VARCHAR(100)
active INT
username VARCHAR(100)
password VARCHAR(100)
Indexes
payment
payment_id INT
customer_id INT
staff_id INT
rental_id INT
amount DECIMAL(10,0)
payment_date DATETIME
Indexes
address
address_id INT
city_id INT
address VARCHAR(100)
address2 VARCHAR(100)
district VARCHAR(100)
postal_code VARCHAR(200)
phone VARCHAR(200)
Indexes
city
city_id INT
city VARCHAR(100)
country_id INT
Indexes
actor
actor_id INT
first_name VARCHAR(100)
last_name VARCHAR(100)
Indexes
PRIMARY