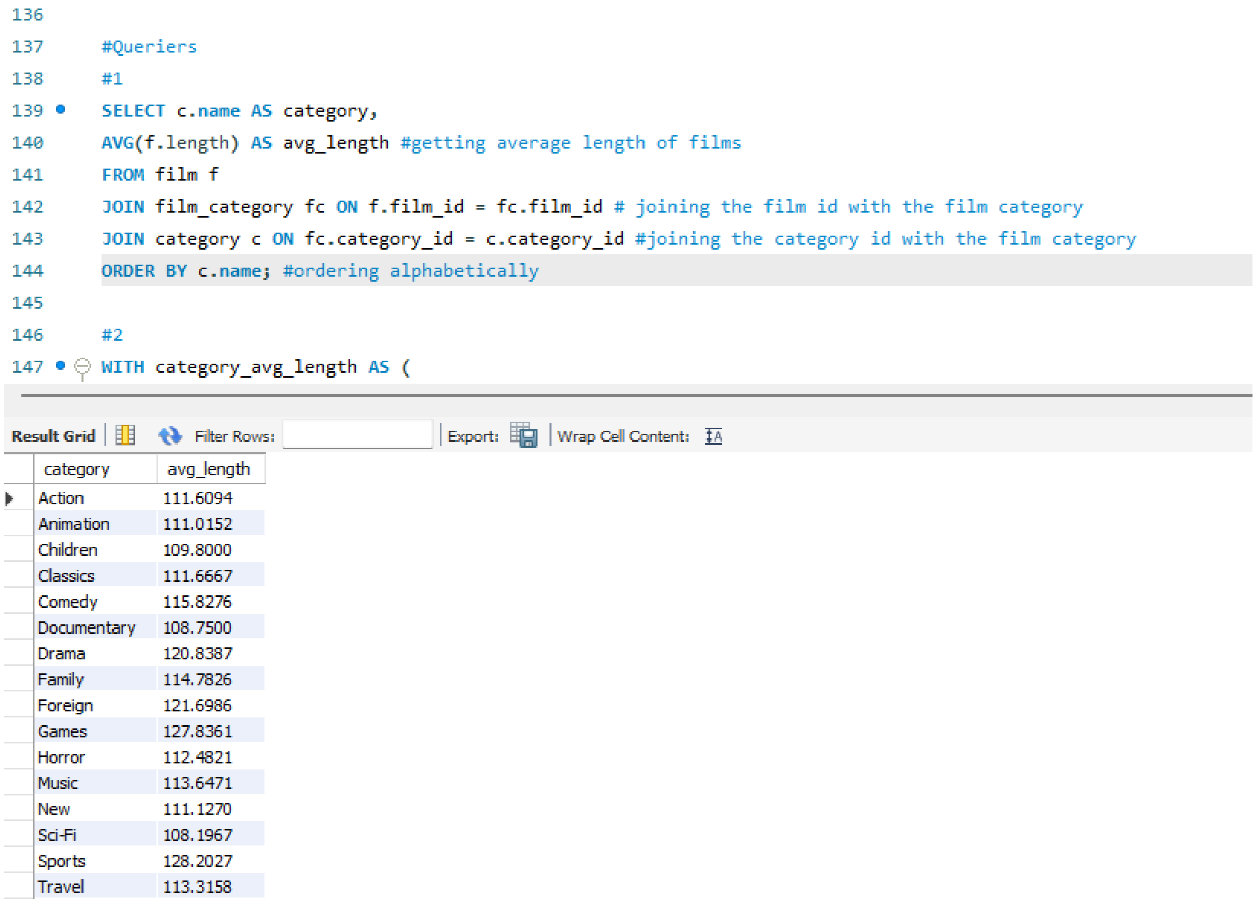
#1

Querrey 1 is fairly simple, we just join the movie categories together and then we get the average for each category (we do this by joining the categories together to check if they are valid)



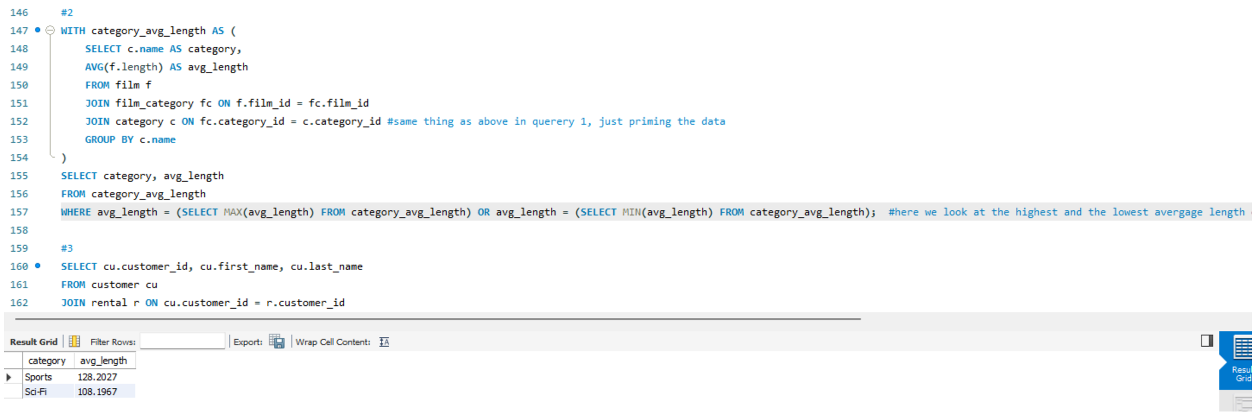
```
#Queriers
#1
SELECT c.name AS category,
AVG(f.length) AS avg_length #getting average length of films
FROM film f
JOIN film_category fc ON f.film_id = fc.film_id # joining the film id with the film category
JOIN category c ON fc.category_id = c.category_id #joining the category id with the film category
ORDER BY c.name; #ordering alphabetically

#2
WITH category_avg_length AS (
```

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

category	avg_length
Action	111.6094
Animation	111.0152
Children	109.8000
Classics	111.6667
Comedy	115.8276
Documentary	108.7500
Drama	120.8387
Family	114.7826
Foreign	121.6986
Games	127.8361
Horror	112.4821
Music	113.6471
New	111.1270
Sci-Fi	108.1967
Sports	128.2027
Travel	113.3158

#2

Very similar to Querery 1, however now we just get the highest and lowest of both categories and then just display those two.



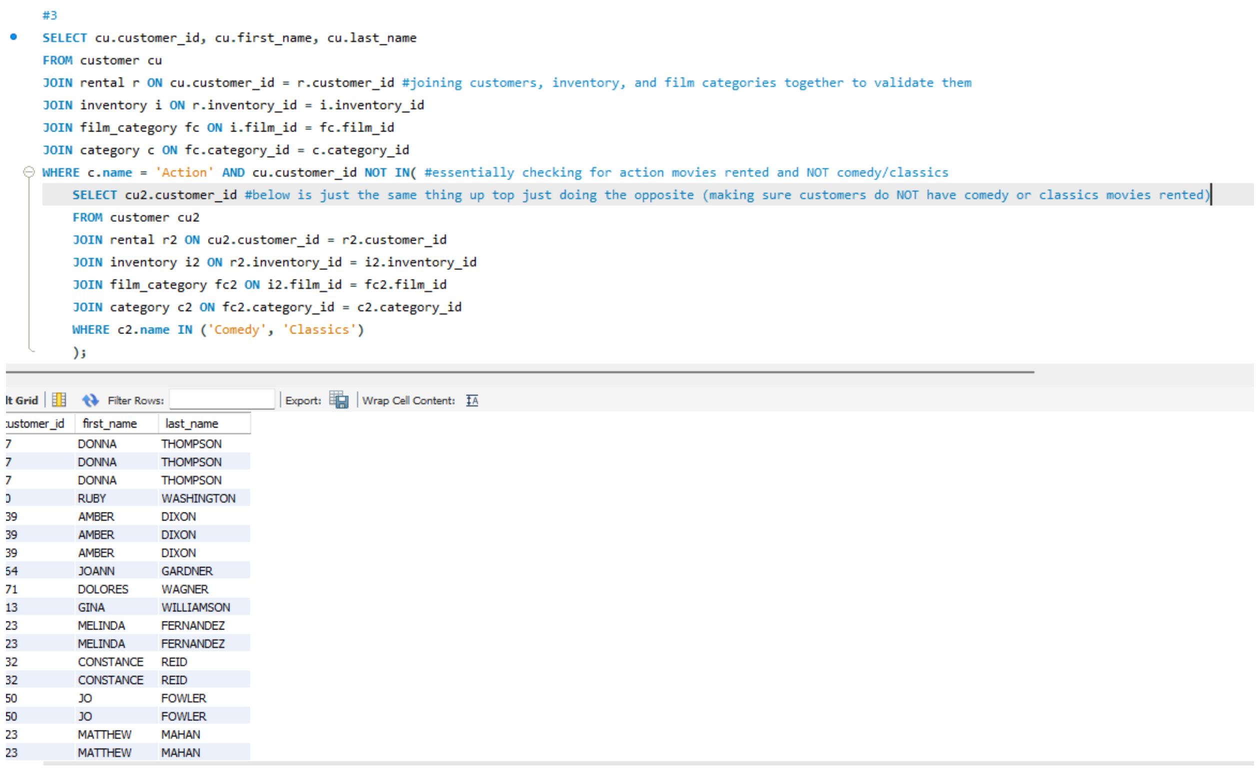
```
#2
WITH category_avg_length AS (
    SELECT c.name AS category,
    AVG(f.length) AS avg_length
    FROM film f
    JOIN film_category fc ON f.film_id = fc.film_id
    JOIN category c ON fc.category_id = c.category_id #same thing as above in querery 1, just priming the data
    GROUP BY c.name
)
SELECT category, avg_length
FROM category_avg_length
WHERE avg_length = (SELECT MAX(avg_length) FROM category_avg_length) OR avg_length = (SELECT MIN(avg_length) FROM category_avg_length);  #here we look at the highest and the lowest avergage length

#3
SELECT cu.customer_id, cu.first_name, cu.last_name
FROM customer cu
JOIN rental r ON cu.customer_id = r.customer_id
```

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

category	avg_length
Sports	128.2027
Sci-Fi	108.1967

#3

For Querrey 3, we essentially just look at each customer's purchases and then determine through the where clause if they have purchased action and NOT purchased Comedy or Classics (For some reason I was getting duplicates here, so maybe the same people purchase multiple action movies?)



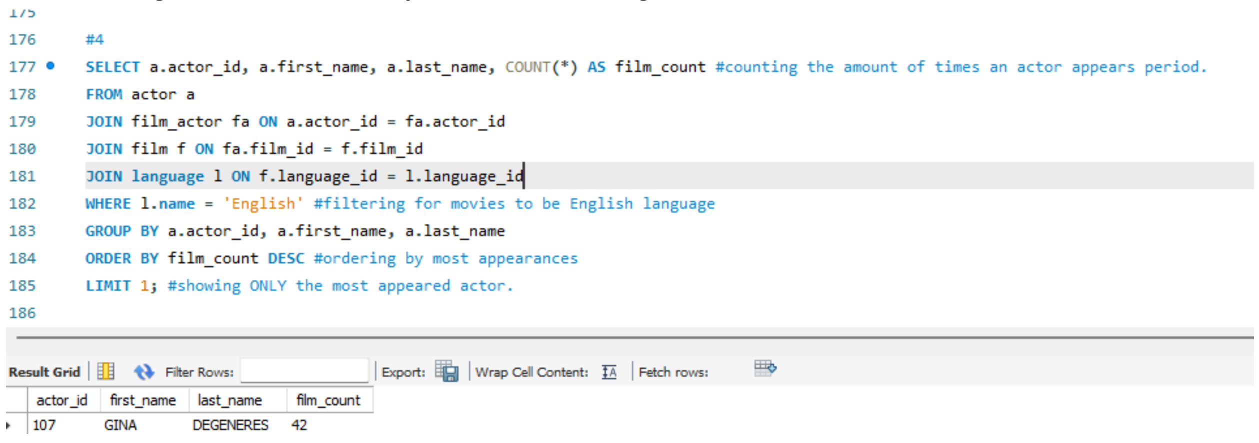
```
#3
SELECT cu.customer_id, cu.first_name, cu.last_name
FROM customer cu
JOIN rental r ON cu.customer_id = r.customer_id #joining customers, inventory, and film categories together to validate them
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film_category fc ON i.film_id = fc.film_id
JOIN category c ON fc.category_id = c.category_id
WHERE c.name = 'Action' AND cu.customer_id NOT IN( #essentially checking for action movies rented and NOT comedy/classics
    SELECT cu2.customer_id #below is just the same thing up top just doing the opposite (making sure customers do NOT have comedy or classics movies rented)
    FROM customer cu2
    JOIN rental r2 ON cu2.customer_id = r2.customer_id
    JOIN inventory i2 ON r2.inventory_id = i2.inventory_id
    JOIN film_category fc2 ON i2.film_id = fc2.film_id
    JOIN category c2 ON fc2.category_id = c2.category_id
    WHERE c2.name IN ('Comedy', 'Classics')
    );
```

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

ustomer_id	first_name	last_name
7	DONNA	THOMPSON
7	DONNA	THOMPSON
7	DONNA	THOMPSON
0	RUBY	WASHINGTON
39	AMBER	DIXON
39	AMBER	DIXON
39	AMBER	DIXON
54	JOANN	GARDNER
71	DOLORES	WAGNER
13	GINA	WILLIAMSON
23	MELINDA	FERNANDEZ
23	MELINDA	FERNANDEZ
32	CONSTANCE	REID
32	CONSTANCE	REID
50	JO	FOWLER
50	JO	FOWLER
23	MATTHEW	MAHAN
23	MATTHEW	MAHAN

#4

For Querrey 4, we get the count of all the actors, then we then look at actors that have shown up in English language movies, then we order by descending to get the most appearances and limit to 1 to get our one and only winner, Gina Degeners



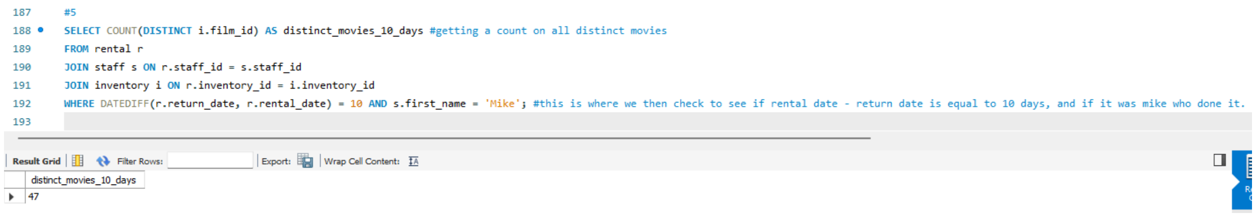
```
#4
SELECT a.actor_id, a.first_name, a.last_name, COUNT(*) AS film_count #counting the amount of times an actor appears period.
FROM actor a
JOIN film_actor fa ON a.actor_id = fa.actor_id
JOIN film f ON fa.film_id = f.film_id
JOIN language l ON f.language_id = l.language_id
WHERE l.name = 'English' #filtering for movies to be English language
GROUP BY a.actor_id, a.first_name, a.last_name
ORDER BY film_count DESC #ordering by most appearances
LIMIT 1; #showing ONLY the most appeared actor.
```

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

actor_id	first_name	last_name	film_count
107	GINA	DEGENERES	42

#5

We simply just count distinct movies that Mike has rented (by using the rental table) to then look at the rent and return date to see if that is equal to 10 days.

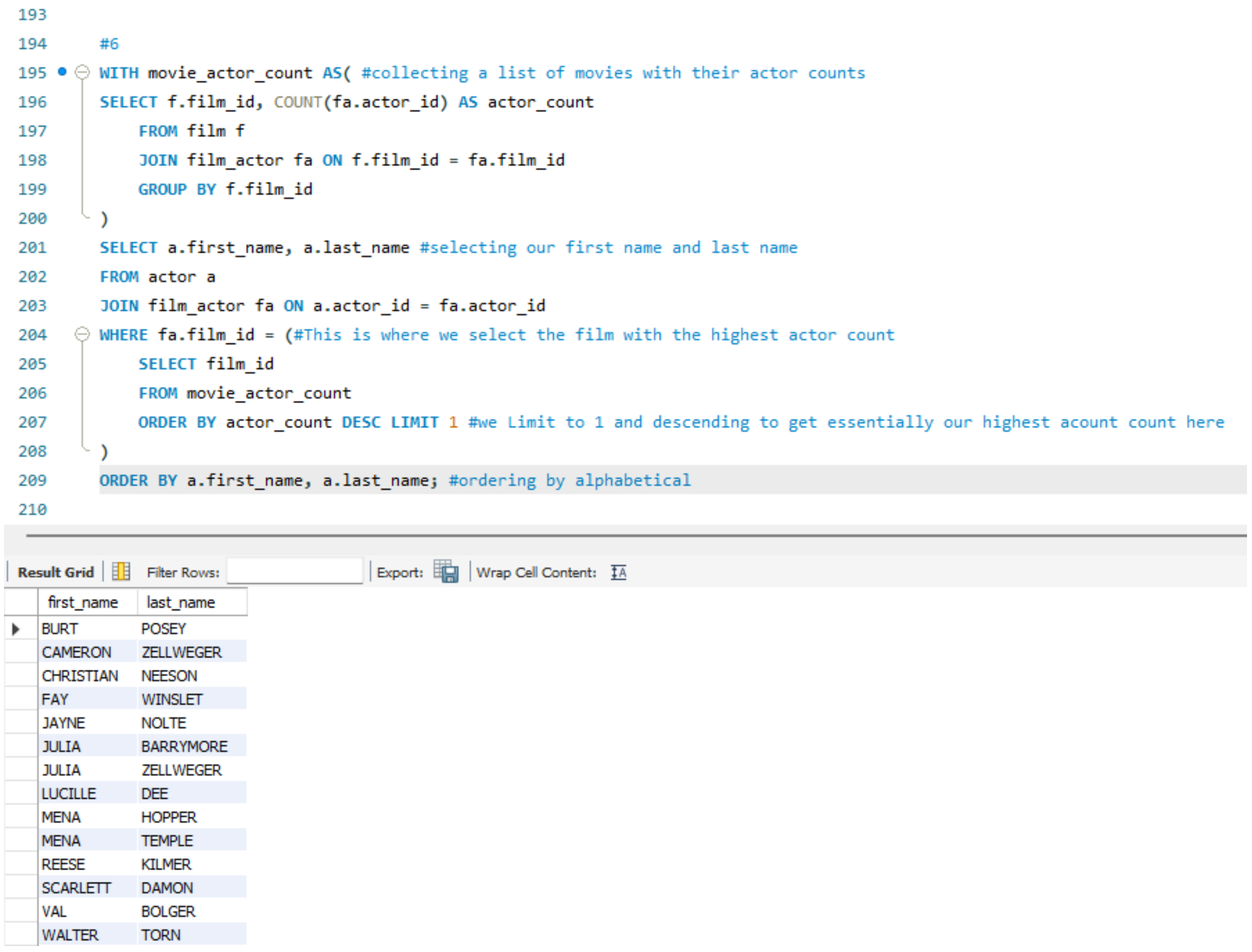


```
#5
SELECT COUNT(DISTINCT i.film_id) AS distinct_movies_10_days #getting a count on all distinct movies
FROM rental r
JOIN staff s ON r.staff_id = s.staff_id
JOIN inventory i ON r.inventory_id = i.inventory_id
WHERE DATEDIFF(r.return_date, r.rental_date) = 10 AND s.first_name = 'Mike'; #this is where we then check to see if rental date - return date is equal to 10 days, and if it was mike who done it.
```

distinct_movies_10_days
47

#6

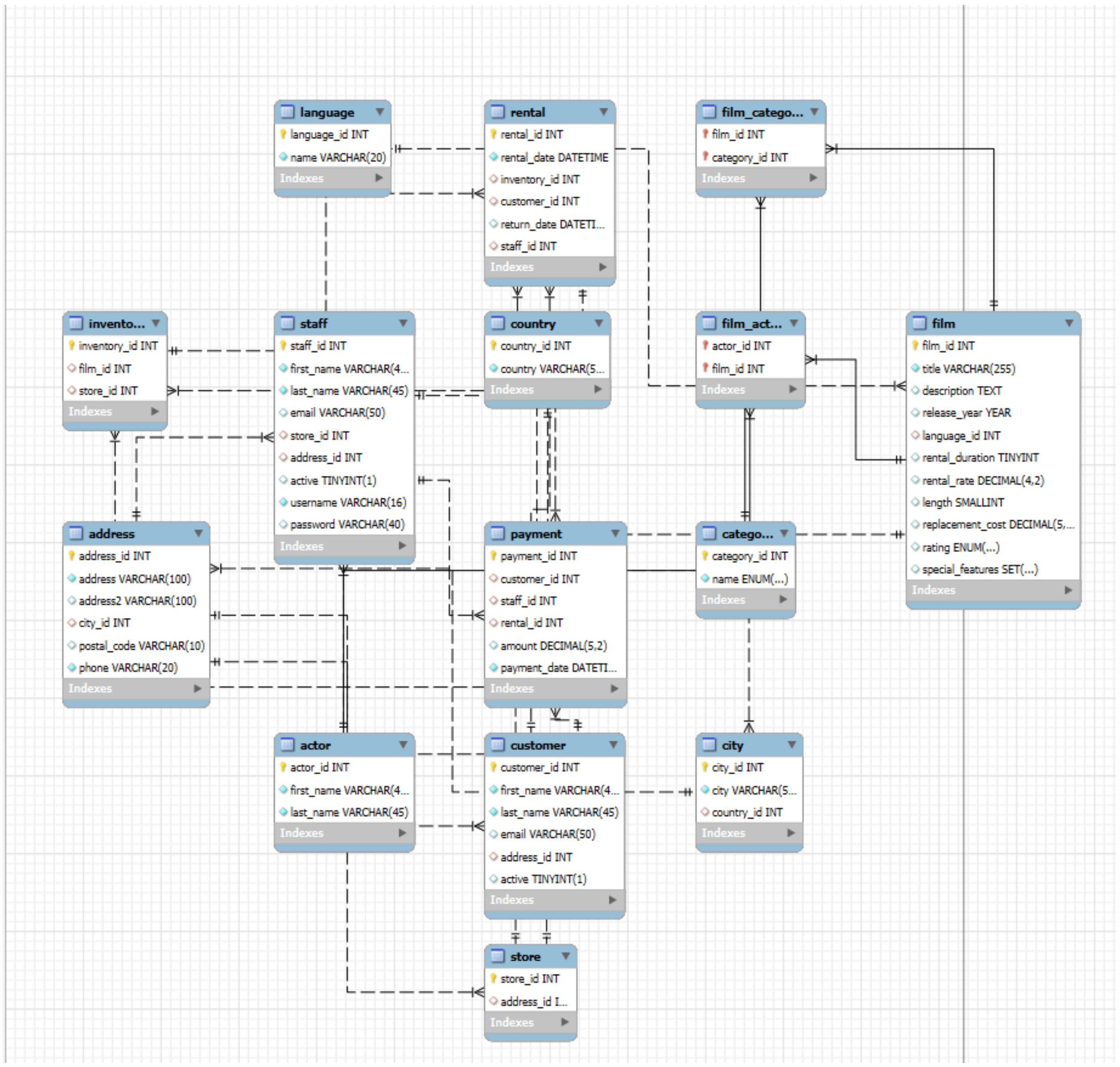
Here we gather a list of the #of actors per movie, and then we get the highest count of actors in a movie to use that as our baseline; then from there, we list of all the actors who were in that movie. (Listing alphabetically)



```

#6
WITH movie_actor_count AS( #collecting a list of movies with their actor counts
SELECT f.film_id, COUNT(fa.actor_id) AS actor_count
    FROM film f
    JOIN film_actor fa ON f.film_id = fa.film_id
    GROUP BY f.film_id
)
SELECT a.first_name, a.last_name #selecting our first name and last name
FROM actor a
JOIN film_actor fa ON a.actor_id = fa.actor_id
WHERE fa.film_id = (#This is where we select the film with the highest actor count
    SELECT film_id
    FROM movie_actor_count
    ORDER BY actor_count DESC LIMIT 1 #we Limit to 1 and descending to get essentially our highest acount count here
)
ORDER BY a.first_name, a.last_name; #ordering by alphabetical

```

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

ERD diagram


language
language_id INT
name VARCHAR(20)
Indexes
rental
rental_id INT
rental_date DATETIME
inventory_id INT
customer_id INT
return_date DATETI...
staff_id INT
Indexes
film_catego...
film_id INT
category_id INT
Indexes
invento...
inventory_id INT
film_id INT
store_id INT
Indexes
staff
staff_id INT
first_name VARCHAR(4...
last_name VARCHAR(45)
email VARCHAR(50)
store_id INT
address_id INT
active TINYINT(1)
username VARCHAR(16)
password VARCHAR(40)
Indexes
country
country_id INT
country VARCHAR(5...
Indexes
film_act...
actor_id INT
film_id INT
Indexes
film
film_id INT
title VARCHAR(255)
description TEXT
release_year YEAR
language_id INT
rental_duration TINYINT
rental_rate DECIMAL(4,2)
length SMALLINT
replacement_cost DECIMAL(5,...
rating ENUM(...)
special_features SET(...)
Indexes
address
address_id INT
address VARCHAR(100)
address2 VARCHAR(100)
city_id INT
postal_code VARCHAR(10)
phone VARCHAR(20)
Indexes
payment
payment_id INT
customer_id INT
staff_id INT
rental_id INT
amount DECIMAL(5,2)
payment_date DATETI...
Indexes
catego...
category_id INT
name ENUM(...)
Indexes
actor
actor_id INT
first_name VARCHAR(4...
last_name VARCHAR(45)
Indexes
customer
customer_id INT
first_name VARCHAR(4...
last_name VARCHAR(45)
email VARCHAR(50)
address_id INT
active TINYINT(1)
Indexes
city
city_id INT
city VARCHAR(5...
country_id INT
Indexes
store
store_id INT
address_id I...
Indexes