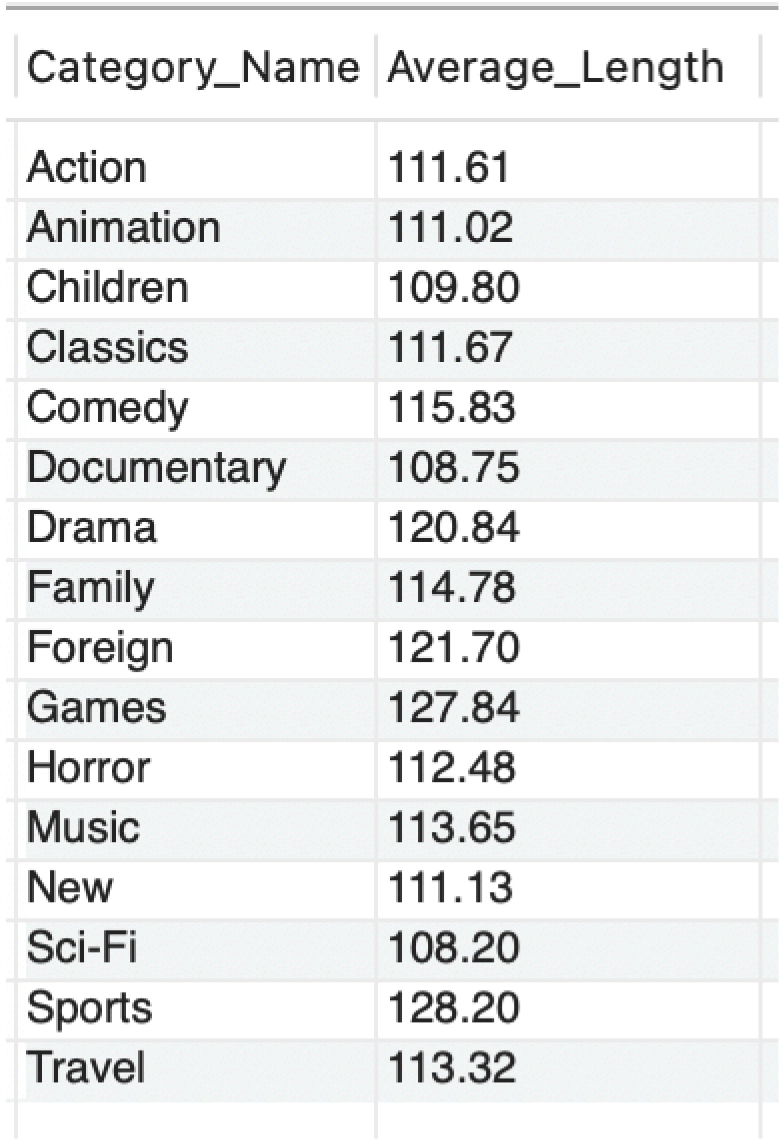
Title Page

- **Title:** DB Assignment 4
- **Your Name:** Cooper Simon
- **Date:** 11/04/25

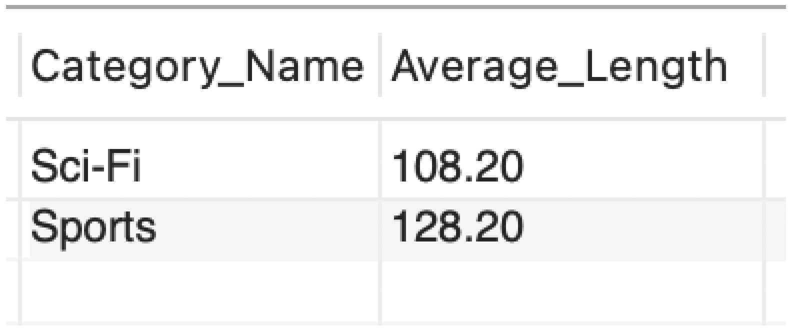
```
/*
******************************************************************************************************************
**************************************************
Query 1. What is the average length of films in each category? List the results in alphabetic
order of categories.
The query joins three tables: film, film_categorycategory, and category
then groups by category name and sorts alphabetically.
This query selects the category name and also finds the average film length for each category.
While listing the result alphabetically.

******************************************************************************************************************
**************************************************** */
select category.name as Category_Name, round(avg(film.length), 2) as Average_Length --
displays the cateogry and computes avg film length
from film
inner join film_category using(film_id)
inner join category using(category_id)
group by category.name
order by category.name; -- orders alphabetically
```



Category_Name	Average_Length
Action	111.61
Animation	111.02
Children	109.80
Classics	111.67
Comedy	115.83
Documentary	108.75
Drama	120.84
Family	114.78
Foreign	121.70
Games	127.84
Horror	112.48
Music	113.65
New	111.13
Sci-Fi	108.20
Sports	128.20
Travel	113.32

```
/*
*******************************************************************************************************************
***************************************************
Query 2. Which categories have the longest and shortest average film lengths?
The query joins three tables: film, film_categorycategory, and category
Then finds the longest and shortest average film length. Uses subqueries to gather this result;
this is extremely similar to the previous query except now we must find just a max and min. We
can do this using subqueries.
*******************************************************************************************************************
***************************************************** */
select category.name as Category_Name, round(avg(film.length), 2) as Average_Length
from film
```

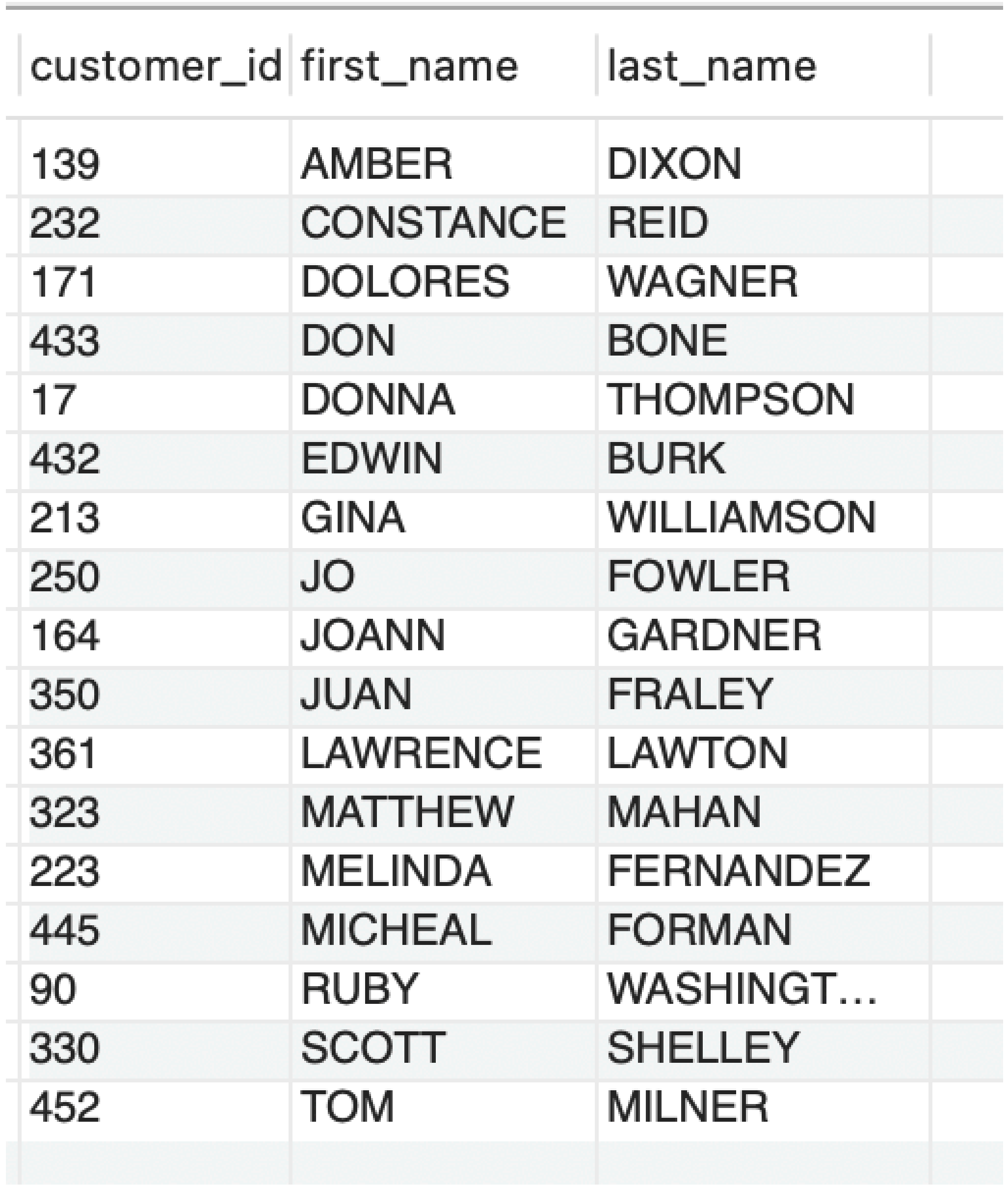
```
inner join film_category using(film_id)
inner join category using(category_id)
group by category.name
having round(avg(film.length), 2) = (
    select max(avg_length) -- subquerry for max length
    from (
        select round(avg(f2.length), 2) as avg_length
                    from film f2
        inner join film_category film_cat2 using(film_id)
                    inner join category category2 using(category_id)
            group by category2.name
    ) as subquery
)
or round(avg(film.length), 2) = (
    select min(avg_length) -- subquerry for min length
    from (
        select round(avg(f3.length), 2) as avg_length
        from film f3
        inner join film_category film_cat3 using(film_id)
                    inner join category category3 using(category_id)
            group by category3.name
    ) as subquery
);
```



Category_Name	Average_Length
Sci-Fi	108.20
Sports	128.20

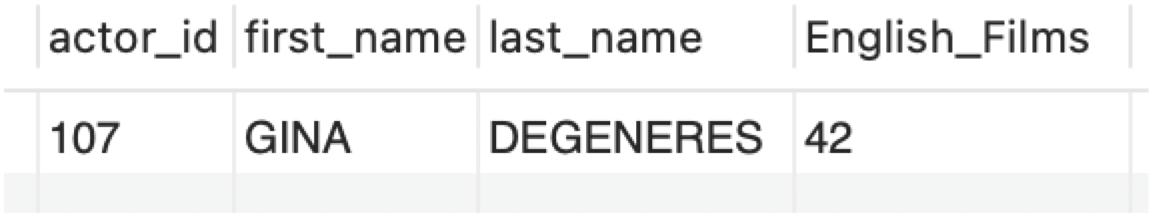
```
/*
****************************************************************************************************************
**************************************************
Query 3. Which customers have rented action but not comedy or classic movies?
We will be using 6 tables for this querry, customer, rental, inverntory, film, film_category, and
category. This will include inner joins and left joins
There will be a main query and a subquery. The main query will find customers who rented
action movies.
The subquery contains customers who rented comedy or classic movies. We will then left join
these two to find who has rented action movies, but not comedy or classics.
****************************************************************************************************************
*************************************************** */
select distinct customer.customer_id, customer.first_name, customer.last_name
```

```
from customer
inner join rental using (customer_id)
inner join inventory using (inventory_id)
inner join film using (film_id)
inner join film_category using (film_id)
inner join category using (category_id)
left join(
	select distinct c2.customer_id  -- subquerry to find customer who rented comedys or
classic
	from customer c2
	inner join rental r2 using (customer_id)
	inner join inventory i2 using (inventory_id)
	inner join film f2 using (film_id)
	inner join film_category fc2 using (film_id)
	inner join category cat2 using (category_id)
   where trim(cat2.name) in ('Comedy' , 'Classics')
) as containsC using(customer_id)
where lower(category.name) in ('action') -- includes people who rented action movies
and containsC.customer_id is null -- discludes people who rented comedy and classics.
order by customer.first_name, customer.last_name;
```



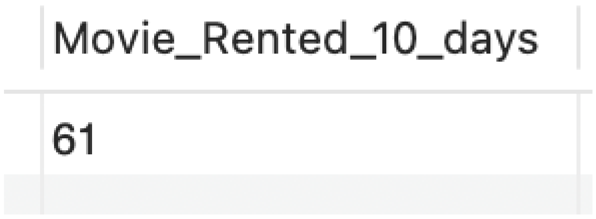
customer_id	first_name	last_name
139	AMBER	DIXON
232	CONSTANCE	REID
171	DOLORES	WAGNER
433	DON	BONE
17	DONNA	THOMPSON
432	EDWIN	BURK
213	GINA	WILLIAMSON
250	JO	FOWLER
164	JOANN	GARDNER
350	JUAN	FRALEY
361	LAWRENCE	LAWTON
323	MATTHEW	MAHAN
223	MELINDA	FERNANDEZ
445	MICHEAL	FORMAN
90	RUBY	WASHINGT…
330	SCOTT	SHELLEY
452	TOM	MILNER

```
/*
***********************************************************************************************************
************************************************
Query 4. Which actor has appeared in the most English-language movies?
This will have an inner join between the tables actor, film_actor, film, and language
We will need to count the filtered films for only English films and then group them by actors.
Finally, order by the count descending and limit it by one to get the top actor.
***********************************************************************************************************
************************************************** */
select actor.actor_id, actor.first_name, actor.last_name, count(film.film_id) as English_Films
from actor
inner join film_actor using(actor_id)
inner join film using (film_id)
inner join language using (language_id)
where language.name = 'English' -- language must be english
group by actor.actor_id, actor.first_name, actor.last_name
order by English_Films desc
limit 1; -- limits top actor
```



actor_id	first_name	last_name	English_Films
107	GINA	DEGENERES	42

```
/*
***************************************************************************************************************
************************************************
Query 5. How many distinct movies were rented for exactly 10 days from the store where Mike
works?
For this query we will need to connect four of our tables using an inner join, rental, inventory,
film and staff.
Then we will filter our data so we are only dealing with staff whos first name was mike.
SQL has a function called DATEDIFF, which can calculate the difference between two dates.
Shows the count of films which had been rented for exactly 10 days from the store where Mike
worked.
***************************************************************************************************************
************************************************** */
select count(distinct film.film_id) as Movie_Rented_10_days
from rental
inner join inventory using (inventory_id)
inner join film using (film_id)
inner join staff on inventory.store_id = staff.store_id
where staff.first_name = 'MIKE' -- filters for staff with first name mike
and datediff(rental.return_date, rental.rental_date) = 10; -- special SQL function which does the
math automatically
```

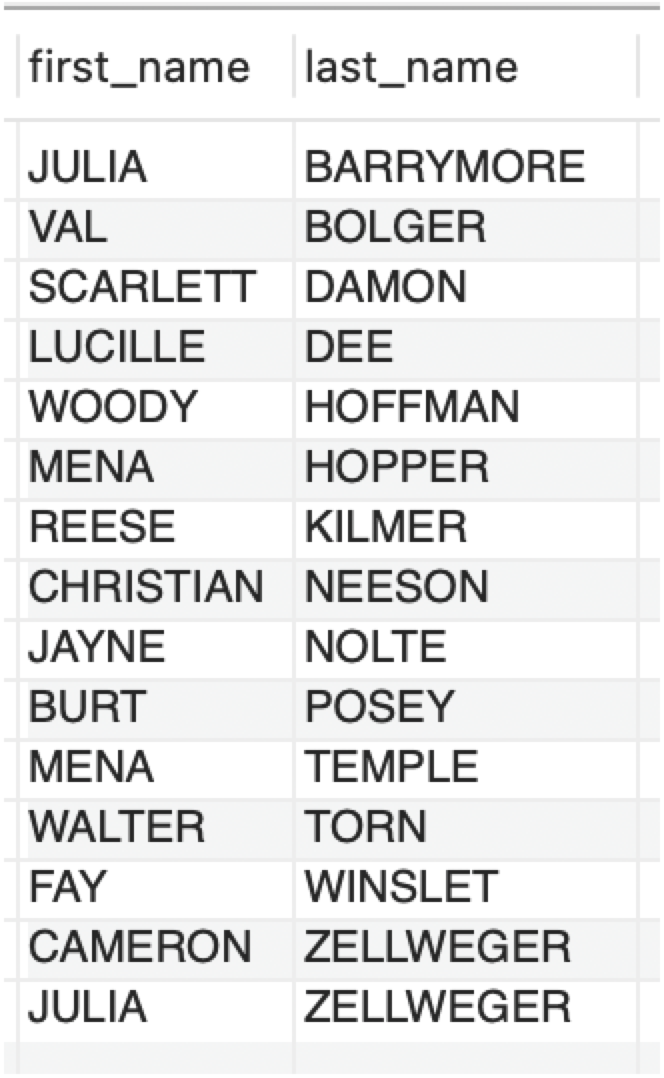


Movie_Rented_10_days
61

```
/*
***************************************************************************************************************
************************************************
Query 6. Alphabetically list actors who appeared in the movie with the largest cast of actors.
Tables Joined: actor and film_actor
```

Count the number of actors per film. Find the film(s) with the maximum actor count. Return all actors who appear in that film. Sort alphabetically by last name and first name.
Expected Output: List of all actors (first, last) in the movie that has the most actors.

```
************************************************************************************************************
************************************************** */
select actor.first_name, actor.last_name
from actor
inner join film_actor using (actor_id)
where film_actor.film_id in(
        select film_id -- finds number of people in each cast
   from film_actor
   group by film_id
   having count(actor_id) =( -- selects the film with the largest cast
                select max(actor_count)
      from (
      select count(actor_id) as actor_count
      from film_actor
      group by film_id
        ) as subquerry
)
)
order by actor.last_name, actor.first_name;
```



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