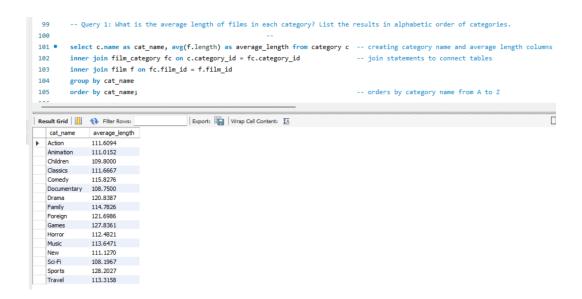
DB Assignment 4 Sean Gor 10.27.2024

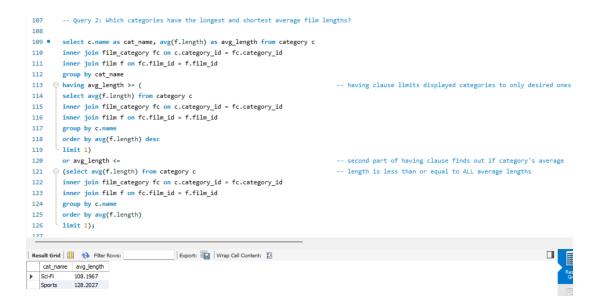
(ER diagram is attached separately in my Github Repo).

Query 1:



This query selects the category name and finds the average length of each category using the avg() function. It then uses the Group By function to match each category with its respective average film length. Finally, the order by function is used to sort the categories based on their names alphabetically.

Query 2:



This query selects category names and associates them with their average lengths like the first query. However, to find categories with just the highest and lowest lengths, I used a having clause with subqueries to find the category with the highest length, and the category with the lowest length (respectively). This ensures that only the categories with the highest and lowest lengths are printed (I also included the average lengths of the categories for clarity).

Query 3:

```
-- Query 3: Which customers have rented action but not comedy or classic movies?
124 • select distinct c.first_name as cust_first_name, c.last_name as cust_last_name from customer c
                                                                                       -- join statements to get necessary data
125
       inner join rental on c.customer id = rental.customer id
126
       inner join inventory on rental.inventory id = inventory.inventory id
       inner join film_category on inventory.film_id = film_category.film_id
       inner join category cat on film_category.category_id = cat.category_id
       where cat.name = 'Action'
                                                                                      -- where clause filters customers' names only if
-- they rented movies
131
       select c2.customer_id from customer c2
       inner join rental on c2.customer id = rental.customer id
132
       inner join inventory on rental.inventory_id = inventory.inventory_id
       inner join film_category on inventory.film_id = film_category.film_id
135
       inner join category cat2 on film_category.category_id = cat2.category_id
136
       where cat2.name = 'Comedy' or cat2.name = 'Classic
                                                                                       -- other where clause limits customers names to
137
                                                                                       -- those who did not rent comedy or classic movies
```

Output(could not fit it all in one screenshot)

	cust_first_name	cust_last_name
	PATRICIA	JOHNSON
	SUSAN	WILSON
	LISA	ANDERSON
	DONNA	THOMPSON
•	CHRISTINE	ROBERTS
	JOYCE	EDWARDS
	MILDRED	BAILEY
	JUDY	GRAY
	KATHY	JAMES
	BEVERLY	BROOKS
	LOUISE	JENKINS
	RUBY	WASHINGTON
	CARMEN	OWENS
	WENDY	HARRISON
	SHANNON	FREEMAN
	SHEILA	WELLS
	ELLEN	SIMPSON
	ANITA	MORALES
	AMBER	DIXON
	APRIL	BURNS
	JOANN	GARDNER
	DOLORES	WAGNER
	MARION	SNYDER
	IDA	ANDREWS
	ROBERTA	HARPER
	TARA	RYAN
	WILMA	RICHARDS
	GINA	WILLIAMSON
	AGNES	BISHOP
	MELINDA	FERNANDEZ
	CONSTANCE	REID

cust_first_name	cust_last_name
NELLIE	GARRETT
MINNIE	ROMERO
GLENDA	FRAZIER
MARIAN	MENDOZA
JO	FOWLER
MARSHA	DOUGLAS
PATSY	DAVIDSON
NORA	HERRERA
JENNY	CASTRO
FELICIA	SUTTON
BOBBIE	CRAIG
MISTY	LAMBERT
JOHN	FARNSWORTH
DAVID	ROYAL
MATTHEW	MAHAN
SCOTT	SHELLEY
JOSHUA	MARK
PETER	MENARD
JUAN	FRALEY
TERRY	GRISSOM
GERALD	FULTZ
RALPH	MADRIGAL
LAWRENCE	LAWTON
WAYNE	TRUONG
VICTOR	BARKLEY
TODD	TAN
JIMMY	SCHRADER
DANNY	ISOM
LEONARD	SCHOFIELD
DALE	RATCLIFF
CHAD	CARBONE
ALFRED	CASILLAS

FRANCIS	SIKES
BRADLEY	MOTLEY
EDWIN	BURK
DON	BONE
ALEXANDER	FENNELL
BERNARD	COLBY
MICHEAL	FORMAN
JAY	ROBB
JIM	REA
TOM	MILNER
RONNIE	RICKETTS
DARRELL	POWER
PEDRO	CHESTNUT
MAURICE	CRAWLEY
HECTOR	POINDEXTER
BRENT	HARKINS
REGINALD	KINDER
CHESTER	BENNER
ELMER	NOE
CORY	MEEHAN
ERIK	GUILLEN
CHRISTIAN	JUNG
JULIO	NOLAND
PERRY	SWAFFORD
SERGIO	STANFIELD
MARION	OCAMPO
SETH	HANNON

This query selects the necessary customers by using join statements to obtain the category names. It also uses a where statement to filter rows being selected (limited to those consisting of action category names, but not category or classic). It also uses a subquery that selects all customer ids that are not to be included in the query (based on the query constraints).

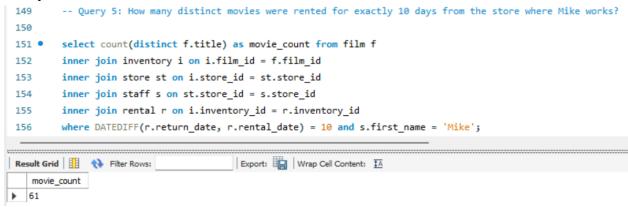
Query 4:

```
-- Query 4: Which actor has appeared in the most English-language movies?
 140
 141 • select first_name, last_name from actor
      inner join film_actor fa on actor.actor_id = fa.actor_id
 142
       inner join film f on fa.film_id = f.film_id
      inner join language 1 on f.language_id = l.language_id
 144
 145 where l.name = 'English'
                                                                                   -- filters movies that are in english language only
 group by first_name, last_name
                                                                                   -- (this is denoted by the name in the language table being
        order by count(f.title) desc
                                                                                   -- english)
      limit 1;
 148
                                                                                   -- limit 1 displays only the maximum result
 Export: Wrap Cell Content: 🖽 | Fetch rows:
   first_name last_name
▶ SUSAN
           DAVIS
```

This query selects all actor's names where the film's language id = 1. Since there is no language name in the film table, I assumed that language ID being 1 would be associated with an

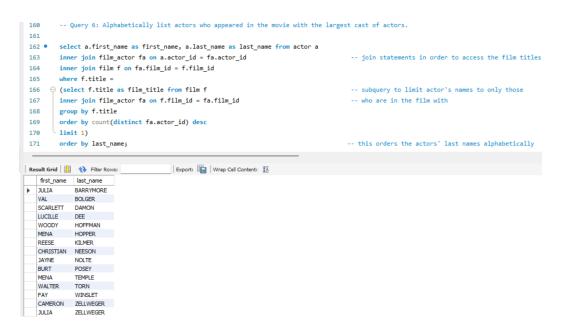
English-language film. The order by function orders by the number of english film titles each actor participated in. Finally, to obtain only actor with the highest # of film titles, limit 1 is used.

Query 5:



This query selects each film title and uses join statements in order to access necessary information from the staff and rental tables. The function DATEDIFF() is used in order to calculate the difference between when the person rented and when the person returned a movie in days. The where clause filters rows to only count film titles that were rented for 10 days at the store Mike works for.

Query 6:



This query selects the first and last names of actors as headings, then uses join statements to access the film titles (to make the desired comparison). The where clause contains a subquery which finds the movie with the most amount of actors (using the count() function to count actors in each movie). Finally, the order by clause sorts the filtered actors' last names alphabetically.