DB Assignment 4
Sean Gor
10.27.2024

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

Query 1:

```
-- Query 1: What is the average length of films in each category? List the results in alphabetic order of categories.
--
select c.name as cat_name, avg(f.length) as average_length from category c  -- creating category name and average length columns
inner join film_category fc on c.category_id = fc.category_id               -- join statements to connect tables
inner join film f on fc.film_id = f.film_id
group by cat_name
order by cat_name;                                                          -- orders by category name from A to Z
```

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

cat_name	average_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

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:

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

select c.name as cat_name, avg(f.length) as avg_length from category c
inner join film_category fc on c.category_id = fc.category_id
inner join film f on fc.film_id = f.film_id
group by cat_name
having avg_length >= (                                  -- having clause limits displayed categories to only desired ones
select avg(f.length) from category c
inner join film_category fc on c.category_id = fc.category_id
inner join film f on fc.film_id = f.film_id
group by c.name
order by avg(f.length) desc
limit 1)
or avg_length <=                                        -- second part of having clause finds out if category's average
(select avg(f.length) from category c                   -- length is less than or equal to ALL average lengths
inner join film_category fc on c.category_id = fc.category_id
inner join film f on fc.film_id = f.film_id
group by c.name
order by avg(f.length)
limit 1);
```

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

cat_name	avg_length
Sci-Fi	108.1967
Sports	128.2027

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:

```sql
-- Query 3: Which customers have rented action but not comedy or classic movies?

select distinct c.first_name as cust_first_name, c.last_name as cust_last_name from customer c
inner join rental on c.customer_id = rental.customer_id                  -- join statements to get necessary data
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
and c.customer_id not in (                                               -- they rented movies
select c2.customer_id from customer c2
inner join rental on c2.customer_id = rental.customer_id
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 cat2 on film_category.category_id = cat2.category_id
where cat2.name = 'Comedy' or cat2.name = 'Classic'                      -- other where clause limits customers names to
);                                                                       -- 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
TANYA	GILBERT

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?

select first_name, last_name from actor
inner join film_actor fa on actor.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'                                  -- 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;                                                  -- limit 1 displays only the maximum result
```

Result Grid | Filter Rows: | 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:

```
-- Query 5: How many distinct movies were rented for exactly 10 days from the store where Mike works?

select count(distinct f.title) as movie_count from film f
inner join inventory i on i.film_id = f.film_id
inner join store st on i.store_id = st.store_id
inner join staff s on st.store_id = s.store_id
inner join rental r on i.inventory_id = r.inventory_id
where DATEDIFF(r.return_date, r.rental_date) = 10 and s.first_name = 'Mike';
```

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

movie_count
61

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:

```
-- Query 6: Alphabetically list actors who appeared in the movie with the largest cast of actors.

select a.first_name as first_name, a.last_name as last_name from actor a
inner join film_actor fa on a.actor_id = fa.actor_id                    -- join statements in order to access the film titles
inner join film f on fa.film_id = f.film_id
where f.title =
(select f.title as film_title from film f                               -- subquery to limit actor's names to only those
inner join film_actor fa on f.film_id = fa.film_id                      -- who are in the film with
group by f.title
order by count(distinct fa.actor_id) desc
limit 1)
order by last_name;                                                     -- this orders the actors' last names alphabetically
```

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

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

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.