

HW 4

Title: DB Assignment 4

Name: Brennen Cramp

Date: 11/4/2024

Problem 1:

First, I wanted to combine the 3 tables (film, film_category, category) with inner joins to be able to get what the categories of each film were. A projection is then used to grab the name (aliased as category) and the average length of the films when grouping by the category. Finally, the results are sorted by the category name (in alphabetical order).

```
153 -- Problem 1:
154 -- What is the average length of films in each category? List the results in alphabetic order of categories
155
156 • SELECT name AS category, AVG(length) AS avgLength -- Grabs the category name and
157 FROM (film as f INNER JOIN film_category AS fc ON f.film_id = fc.film_id -- Combines films and film_cat
158      INNER JOIN category AS c ON c.category_id = fc.category_id) -- Resulting previous table cor
159 GROUP BY category -- Groups by the film's categor
160 ORDER BY category; -- In ascending order of the n
161
```

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

	category	avgLength
▶	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

Problem 2:

Using a CTE for the problem, I first combined 3 tables (film, film_category, category) with inner joins to be able to get what the categories of each film were. Next, a projection is used to grab the name (aliased as category) and the average length of the films when grouping by the category. With that query forming my CTE, my next queries (combined using a union) retrieve the category and average film length where the first filters for the maximum average film length and the second filters for the minimum average film length.

```
152      -- Problem 2:
153      -- Which categories have the longest and shortest average film lengths?
154
155      WITH avgFilmLengthPerCat AS (
156          SELECT name AS category, AVG(length) AS avgLength
157          FROM (film AS f INNER JOIN film_category AS fc ON f.film_id = fc.film_id
158              INNER JOIN category AS c ON c.category_id = fc.category_id)
159          GROUP BY category
160      )
161      -- Query that gets the category with the longest average length
162      SELECT category, avgLength      -- Grabs the category and the average film le
163      FROM avgFilmLengthPerCat      -- Uses the CTE formed above
164      WHERE avgLength = (           -- Filters where the average length is equal
165          SELECT MAX(avgLength)      -- Grabs the max average film length per cate
166          FROM avgFilmLengthPerCat  -- Uses the CTE formed above
167      )
168
169      UNION                        -- Uses a union to combine the longest and sh
170
171      -- Query that gets the category with the shortest average length
172      SELECT category, avgLength      -- Grabs the category and the average film le
173      FROM avgFilmLengthPerCat      -- Uses the CTE formed above
174      WHERE avgLength = (           -- Filters where the average length is equal
175          SELECT MIN(avgLength)      -- Grabs the min average film length per cate
176          FROM avgFilmLengthPerCat  -- Uses the CTE formed above
177      );
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
category	avgLength			
Sports	128.2027			
Sci-Fi	108.1967			

Problem 3:

Forming a CTE for the problem, I combined 6 tables (category, film_category, film, inventory, rental, customer) with inner joins to get a table tracking customers and the categories of movies they rented by projecting the customers full name (using CONCAT) and the category of the movie. The following query gets the customers who have rented action movies and then I use an except to remove customers who have also rented either a comedy or classics (final query), ordering by the customer's full name.

```
176      -- Problem 3:
177      -- Which customers have rented action but not comedy or classic movies?
178
179  WITH custCategories AS (
180      SELECT CONCAT(first_name, ' ', last_name) AS full_name, name AS category
181      FROM (category AS cat JOIN film_category AS fc ON cat.category_id = fc.category_id
182            JOIN film AS f ON f.film_id = fc.film_id
183            JOIN inventory AS i ON i.film_id = f.film_id
184            JOIN rental AS r ON r.inventory_id = i.inventory_id
185            JOIN customer AS c ON c.customer_id = r.customer_id)
186  )
187  -- Query that gets the customers who have rented Action movies
188  SELECT full_name                                -- Grabs the full name of the cu
189  FROM custCategories                             -- Uses the CTE formed above
190  WHERE category = "Action"                       -- Filters where the category is
191
192  EXCEPT                                       -- Uses an except to remove cust
193
194  -- Query that gets the customers who have rented Comedy or Classics movies
195  SELECT full_name                                -- Grabs the full name of the cus
196  FROM custCategories                             -- Uses the CTE formed above
197  WHERE category = "Comedy" OR category = "Classics" -- Filters where the category is
198  ORDER BY full_name;                           -- Alphabetically orders by the c
```

full_name
AMBER DIXON
CONSTANCE REID
DOLORES WAGNER
DON BONE
DONNA THOMPSON
EDWIN BURK
GINA WILLIAMSON
JO FOWLER
JOANN GARDNER
JUAN FRALEY
LAWRENCE LAWTON
MATTHEW MAHAN
MELINDA FERNAN...
MICHEAL FORMAN
RUBY WASHINGTON
SCOTT SHELLEY
TOM MILNER

Problem 4:

First, I wanted to combine 4 tables (actor, film_actor, film, language) with inner joins to be able to get which actors were in which films and the language the film was. Next, a projection was used, concatenating the first and last name of the actor to make the full name and counting the number of movie titles, grouping by the actor's name, to find the total number of movies the actor was in. I filtered on the movies where the language was English. Finally, I ordered the number of movies the actor was in descending order and limited the output to 1 to grab which actor appeared in the most English-language movies.

```
206      -- Problem 4:
207      -- Which actor has appeared in the most English-language movies?
208
209 •   SELECT CONCAT(first_name, ' ', last_name) AS actor, COUNT(title) AS moviesIn
210      FROM (actor AS a INNER JOIN film_actor AS fa ON a.actor_id = fa.actor_id
211           INNER JOIN film AS f ON f.film_id = fa.film_id
212           INNER JOIN language AS l ON l.language_id = f.language_id)
213      WHERE name = "English"
214      GROUP BY actor
215      ORDER BY moviesIn DESC LIMIT 1;
216
```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
	actor	moviesIn				
▶	SUSAN DAVIS	54				

Problem 5:

Creating a CTE for the problem, I first combined 4 tables (rental, inventory, staff, film) with inner joins to be able to get which staffer is connected to the store and what movies are sold at those stores. Next, I used a projection to grab distinct movie titles and filtered on where the first name of the staffer is "Mike" and where the difference in the return and rental dates (using the DATEDIFF function) is 10 days. After the CTE had been formed, I used a projection on the new table to count all the movie titles that were rented for exactly 10 days.

```
212 -- Problem 5:
213 -- How many distinct movies were rented for exactly 10 days from the store where Mike works?
214
215 WITH distMoviesRentedForTenDays AS ( -- Creates a
216     SELECT DISTINCT title -- Grabs the
217     FROM (rental AS r JOIN inventory AS i ON i.inventory_id = r.inventory_id -- Combines i
218         JOIN staff AS s ON s.store_id = i.store_id -- Resulting
219         JOIN film AS f ON f.film_id = i.film_id) -- Resulting
220     WHERE first_name = "Mike" AND DATEDIFF(return_date, rental_date) = 10 -- Filters on
221 )
222 SELECT COUNT(title) AS distinctMoviesRentedForTenDays -- Grabs the
223 FROM distMoviesRentedForTenDays; -- Uses the CTE
224
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
	distinctMoviesRentedForTenDays		
▶	61		

Problem 6:

Using a CTE for the problem, I first combined 3 tables (actor, film_actor, film) with inner joins to get which movies had the most actors in them using a projection to retrieve the title and count the actor's full name grouping by the movie title. With that query forming my CTE, my next query uses the same table combinations that were utilized in the CTE, grabbing the actor's full name and the movie title. From here, I used a filter to grab the movie title that used a subquery which grabs the title of the movie from the CTE that was filtered on the number of actors where another subquery was used to find the max number of actors from the CTE; all ending in finding which actors were in the movie that had the most number of actors and ordering the actors names alphabetically (by first name).

```

226 -- Problem 6:
227 -- Alphabetically list actors who appeared in the movie with the largest cast of actors.
228
229 WITH actorsPerMovie AS (                                -- Creates
230     SELECT title, COUNT(CONCAT(first_name, ' ', last_name)) AS numOfActors -- Grabs t
231     FROM (actor AS a JOIN film_actor AS fa ON a.actor_id = fa.actor_id -- Combine
232          JOIN film AS f ON f.film_id = fa.film_id) -- Resulti
233     GROUP BY title -- Groups
234 )
235     SELECT CONCAT(first_name, ' ', last_name) AS actor, title -- Grabs the conca
236     FROM (actor AS a JOIN film_actor AS fa ON a.actor_id = fa.actor_id -- Combines actor
237          JOIN film AS f ON f.film_id = fa.film_id) -- Resulting previ
238     WHERE title = ( -- Filters on wher
239         SELECT title -- Grabs the movie
240         FROM actorsPerMovie -- Uses the CTE fo
241         WHERE numOfActors = ( -- Filters on wher
242             SELECT MAX(numOfActors) -- Grabs the max n
243             FROM actorsPerMovie -- Uses the CTE fo
244         )
245     )
246     ORDER BY actor; -- Orders on the a

```

Result Grid		Filter Rows:	Ex
	actor	title	
▶	BURT POSEY	LAMBS CINCINATTI	
	CAMERON ZELLWEGER	LAMBS CINCINATTI	
	CHRISTIAN NEESON	LAMBS CINCINATTI	
	FAY WINSLET	LAMBS CINCINATTI	
	JAYNE NOLTE	LAMBS CINCINATTI	
	JULIA BARRYMORE	LAMBS CINCINATTI	
	JULIA ZELLWEGER	LAMBS CINCINATTI	
	LUCILLE DEE	LAMBS CINCINATTI	
	MENA HOPPER	LAMBS CINCINATTI	
	MENA TEMPLE	LAMBS CINCINATTI	
	REESE KILMER	LAMBS CINCINATTI	
	SCARLETT DAMON	LAMBS CINCINATTI	
	VAL BOLGER	LAMBS CINCINATTI	
	WALTER TORN	LAMBS CINCINATTI	
	WOODY HOFFMAN	LAMBS CINCINATTI	

ERD Diagram:

Important item to note is that the imported data for the film table's special_features field was not structured correctly since it separated the data by commas, causing the values being put in non-existent columns. In order to account for this error, I created 3 extra columns (labeled special_features_2, special_features_3, and special_features_4) to be able to store those values.

