## HW<sub>4</sub>

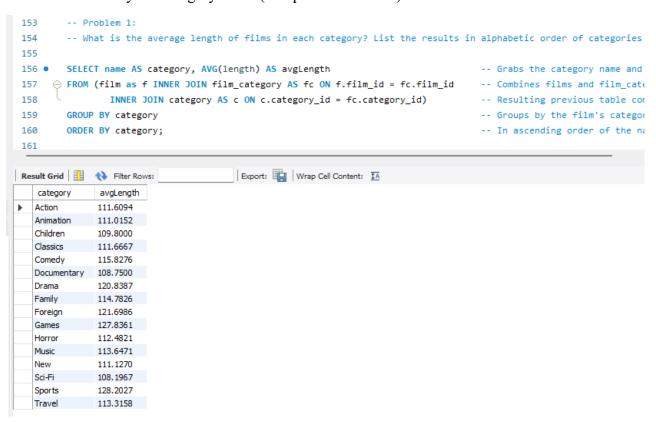
Title: DB Assignment 4

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).



#### 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.

```
-- Problem 2:
152
        -- Which categories have the longest and shortest average film lengths?
153
154
155 • ⊝ WITH avgFilmLengthPerCat AS (
             SELECT name AS category, AVG(length) AS avgLength
156
            FROM (film as f INNER JOIN film category AS fc ON f.film id = fc.film id
157
                     INNER JOIN category AS c ON c.category_id = fc.category_id)
158
159
            GROUP BY category
        )
160
        -- Query that gets the category with the longest average length
161
        SELECT category, avgLength
                                           -- Grabs the category and the average film le
162
        FROM avgFilmLengthPerCat
                                           -- Uses the CTE formed above
163
164

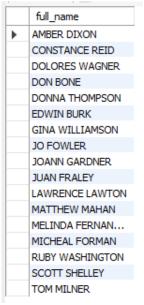
    ○ WHERE avgLength = (
                                           -- Filters where the average length is equal
            SELECT MAX(avgLength)
                                           -- Grabs the max average film length per cate
165
            FROM avgFilmLengthPerCat
                                           -- Uses the CTE formed above
166
        )
167
168
                                           -- Uses a union to combine the longest and sh
169
        UNION
170
171
        -- Query that gets the category with the shortest average length
        SELECT category, avgLength
                                           -- Grabs the category and the average film le
172
                                           -- Uses the CTE formed above
        FROM avgFilmLengthPerCat
173
                                           -- Filters where the average length is equal
174

    ○ WHERE avgLength = (
            SELECT MIN(avgLength)
                                           -- Grabs the min average film length per cate
175
            FROM avgFilmLengthPerCat
                                           -- Uses the CTE formed above
176
177
        );
Result Grid Filter Rows:
                                     Export: Wrap Cell Content: IA
   category
            avgLength
           128,2027
  Sports
  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:
        -- Which customers have rented action but not comedy or classic movies?
177
178
179 • ⊝ WITH custCategories AS (
            SELECT CONCAT(first_name, ' ', last_name) AS full_name, name AS category
180
            FROM (category AS cat JOIN film_category AS fc ON cat.category_id = fc.category_id
181
182
                    JOIN film AS f ON f.film_id = fc.film_id
183
                    JOIN inventory AS i ON i.film_id = f.film_id
                    JOIN rental AS r ON r.inventory_id = i.inventory_id
184
185
                    JOIN customer AS c ON c.customer_id = r.customer_id)
186
187
        -- Ouery that gets the customers who have rented Action movies
        SELECT full_name
188
                                                                 -- Grabs the full name of the cu
        FROM custCategories
                                                                 -- Uses the CTE formed above
189
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
        WHERE category = "Comedy" OR category = "Classics"
197
                                                                -- Filters where the category is
198
        ORDER BY full_name;
                                                                -- Alphabetically orders by the c
```



#### 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:
         -- Which actor has appeared in the most English-language movies?
207
208
         SELECT CONCAT(first name, ' ', last name) AS actor, COUNT(title) AS moviesIn
209 •

→ FROM (actor AS a INNER JOIN film actor AS fa ON a.actor id = fa.actor id
210
                 INNER JOIN film AS f ON f.film id = fa.film id
211
                 INNER JOIN language AS 1 ON l.language id = f.language id)
212
        WHERE name = "English"
213
214
        GROUP BY actor
215
        ORDER BY moviesIn DESC LIMIT 1;
216
                                          Export: Wrap Cell Content: TA Fetch rows:
                                                                                     4
Result Grid
              Filter 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
             FROM (rental AS r JOIN inventory AS i ON i.inventory_id = r.inventory_id
217
                                                                                              -- Combines
                      JOIN staff AS s ON s.store_id = i.store_id
218
                                                                                              -- Resulting
                      JOIN film AS f ON f.film_id = i.film_id)
219
                                                                                              -- Resulting
             WHERE first_name = "Mike" AND DATEDIFF(return_date, rental_date) = 10
                                                                                              -- Filters or
220
221
         SELECT COUNT(title) AS distinctMoviesRentedForTenDays
222
                                                                                              -- Grabs the
         FROM distMoviesRentedForTenDays;
                                                                                              -- Uses the (
223
224
Result Grid | Filter Rows:
                                      Export: Wrap Cell Content: IA
    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
            SELECT title, COUNT(CONCAT(first_name, ' ', last_name)) AS numOfActors
230
                                                                                       -- Grabs t
            FROM (actor AS a JOIN film_actor AS fa ON a.actor_id = fa.actor_id
231
                                                                                       -- Combine
                    JOIN film AS f ON f.film id = fa.film id)
232
                                                                                       -- Resulti
233
            GROUP BY title
234
235
        SELECT CONCAT(first_name, ' ', last_name) AS actor, title
                                                                               -- Grabs the conca

⊖ FROM (actor AS a JOIN film_actor AS fa ON a.actor_id = fa.actor_id
236
                                                                              -- Combines actor
                    JOIN film AS f ON f.film_id = fa.film_id)
237
                                                                               -- Resulting previ
238
     -- Filters on wher
239
            SELECT title
                                                                               -- Grabs the movie
240
            FROM actorsPerMovie
                                                                               -- Uses the CTE fo
            WHERE numOfActors = (
                                                                               -- Filters on wher
241
                SELECT MAX(numOfActors)
                                                                               -- Grabs the max n
242
243
                FROM actorsPerMovie
                                                                               -- Uses the CTE fo
            )
245
        ORDER BY actor;
                                                                               -- Orders on the a
246
```

	actor	title
Þ	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.

