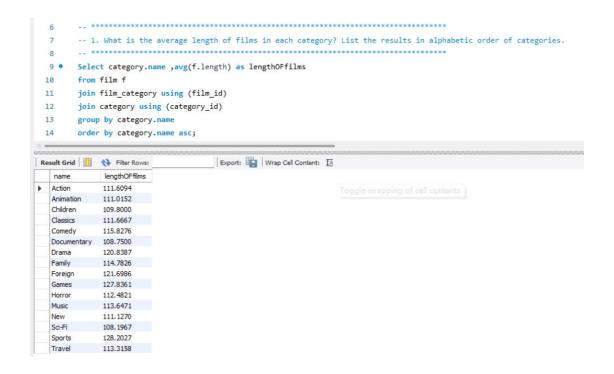
Report of HW4 Name: Deare Date:2024/10/21

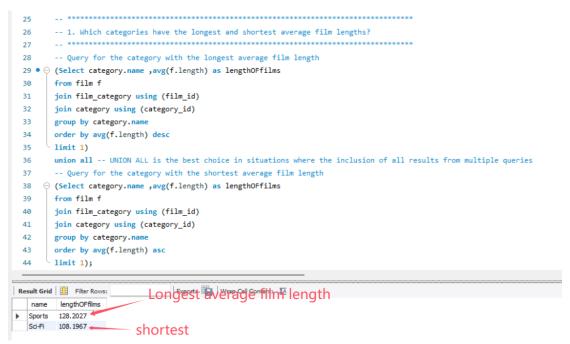
1. What is the average length of films in each category? List the results in alphabetic order of categories.

SQL Component	Explanation
SELECT category.name, AVG(f.length) AS lengthOFfilms	Selects the category name and calculates the average length of films within each category, labeling it as lengthOFfilms.
FROM film AS f	Specifies the source table for film data, using f as an alias for the film.
JOIN film_category USING (film_id)	Joins the film table to the film_category table by matching entries based on the film_id column.
JOIN category USING (category_id)	Joins the film_category table to the category table by matching entries based on the category_id column.
GROUP BY category.name	Group the results by the category name to calculate the average length for films in each category.
ORDER BY category.name ASC	Orders the grouped results alphabetically by category name for organized output.



- 2. Which categories have the longest and shortest average film lengths?
- $-\!-$ UNION ALL is the best choice in situations where the inclusion of all results from multiple queries
- -- Query for the category with the shortest average film length

SQL Part	Description
(Select category.name,	This part of the query selects the name of the
avg(f.length) as	film category and calculates the average
lengthOFfilms	length of films within that category.
from film f	Specifies that the data is sourced from the
	film table, aliased as f.
join film_category using	Joins the film table with the film_category
(film_id)	table on the film_id, linking films to their
	categories.
join category using	Joins the film_category table with the
(category_id)	category table on the category_id, providing
	access to category names.
group by category.name	Groups the results by the film category name
	to enable aggregation (average calculation)
	per category.
order by avg(f.length)	Orders the results by the average film length
desc	in descending order, so the category with the
	longest films appears first.
limit 1)	Limits the results to the single category with
	the longest average film length.
union all	Combines the results of the two queries. UNION
	ALL includes all results from both queries,
	including duplicates.
(Select category. name,	Starts the second part of the query,
avg(f.length) as	structurally similar to the first, to select
lengthOFfilms	the category with the shortest films.
order by avg(f.length)	Changes the order to ascending to find the
asc	category with the shortest films.
limit 1);	Limits the results to the single category with
	the shortest average film length.



- 3. Which customers have rented action but not comedy or classic movies?
- -- Assuming combination of first_name and last_name is customer's name
- -- Assuming the customer's name must be distinct
- -- Assuming the catagory name 'classic' comes from the constraint set {Classics}
- -- Assuming represented result by customer's name and customer_id

SQL Part	Description
select distinct	Selects unique combinations of customer first
	and last names, along with their customer IDs,
	formatted as "Customer_name" and "cid".
from customer	Indicates that the data retrieval starts from
	the customer table, establishing the base
	table for further joins.
join rental	Joins the customer table to the rental table
using(customer_id)	using the customer_id field, including data
	about each customer's rentals.
join inventory	Joins the rental table to the inventory table
using(inventory_id)	using the inventory_id, providing access to
	details of rented items.
join film using(film_id)	Joins the inventory table to the film table
	using the film_id, allowing access to film
	details related to the inventory.
join film_category	Joins the film table to the film_category
using(film_id)	table, linking films to their categories using
	the film_id.
join category	Joins the film_category table to the category
using(category_id)	table using the category_id, facilitating

where
category.name='Action'
and customer_id
not in

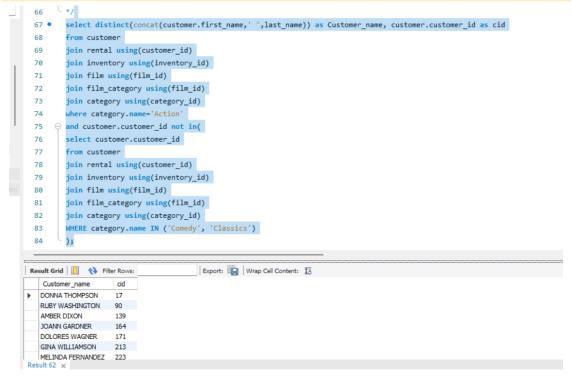
filtering of films by category.

Filters the results to include only entries
where the film's category is 'Action'.

Adds a condition to exclude certain customers.

This part is incomplete and would typically
include a subquery to specify customers to
exclude (e.g., those who have rented 'Comedy'

or 'Classics').



- 4. Which actor has appeared in the most English-language movies?
- -- Assuming combination of first name and last name is actor's full name
- -- Assuming representing result by actor_id and actor full name and number of films they participate in
- -- Due to some actors having the same first name and last name like SUSAN DAVIS, it is better to represent the result by actor_id(PK), actor full name, number of films they participated in

SQL	Description
SELECT CONCAT (actor.first_name, '',	Selects the full name of the
actor.last_name) AS actor_name,	actor by concatenating the first
COUNT (film. film_id) AS num_films,	and last names, counts the number
actor.actor_id as actor_id	of films each actor has appeared
	in, and selects the actor's ID.
FROM actor	Specifies the actor table as the
	source of data.
JOIN film_actor USING(actor_id)	Joins the actor table with the
	film_actor table using the

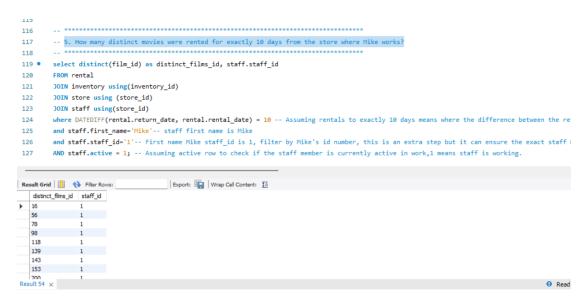
	actor_id to match records.
JOIN film USING(film_id)	Joins the film_actor table with the film table using the film_id to match records.
JOIN language USING(language_id)	Joins the film table with the language table using the language_id to match records.
WHERE language_id = '1'	Filters the results to include only films in language_id '1'.
GROUP BY actor_name, actor_id	Groups the results by actor name and actor ID, which is necessary for counting the films per actor.
ORDER BY num_films DESC	Orders the results by the number of films in descending order, so the actor with the most films is listed first.
LIMIT 1	Limits the results to only the top entry (actor with the most films).



- 5. How many distinct movies were rented for exactly 10 days from the store where Mike works?
- -- Assuming active row to check if the staff member is currently active in work, 1 means staff is working.
- -- Assuming rentals for exactly 10 days means where the difference between the return_date and the rental_date.

SQL Part	Description
SELECT DISTINCT (film. film_id) AS	Selects unique film IDs and the staff
distinct_films_id,	ID, ensuring no duplicates in the list
staff.staff_id	of film IDs.
FROM rental	Specifies that the data is being
	selected from the rental table.
JOIN inventory ON	Joins the rental table with the
rental.inventory_id =	inventory table, linking them via the

inventory.inventory_id	inventory_id.
JOIN store ON inventory.store_id	Joins the inventory table with the
= store.store_id	store table, using store_id to link
	inventory items to their stores.
JOIN staff ON store.store_id =	Joins the store table with the staff
staff.store_id	table, linking them by store_id to
	associate staff members with stores.
WHERE	Filters for rentals where the
DATEDIFF (rental.return_date,	difference between the return_date
rental.rental_date) = 10	and rental_date is exactly 10 days.
AND staff.first_name = 'Mike'	Narrows down records to those handled
	by staff members named Mike.
AND staff.staff_id = '1'	Further filters to ensure the specific
	staff member, Mike, has the staff_id
	of '1' First name Mike staff_id is
	1, filter by Mike's id number, this is
	an extra step but it can ensure the
	exact staff Mike, because lot of
	staff's first name could be Mike
AND staff.active = 1	Includes only those entries where the
	staff member is currently active
	(active = 1).



6. Alphabetically list actors who appeared in the movie with the largest cast of actors.

SQL Part	Description
WITH MovieCastCounts AS ()	Common Table Expression (CTE): This
	section calculates the cast size for
	each film by counting the number of

	distinct actors associated with each film.
SELECT film.film_id, COUNT(actor.actor_id) AS cast_size	This command selects the film ID and counts the number of unique actor IDs associated with each film, effectively measuring the size of the cast per film.
FROM film JOIN film_actor USING(film_id) JOIN actor USING(actor_id)	Joins the film, film_actor, and actor tables to link films with their actors. These joins ensure that all necessary data about films and their actors can be collated.
GROUP BY film.film_id	Groups the results by film ID, which is necessary for the COUNT function to calculate the number of actors per film.
MaxCastMovie AS ()	CTE: Identifies the film(s) with the largest cast. It uses the results of the MovieCastCounts CTE to find the maximum cast size.
SELECT film_id FROM MovieCastCounts WHERE cast_size = (SELECT MAX(cast_size) FROM MovieCastCounts)	Selects the film ID from the previous CTE where the cast size equals the maximum cast size found across all movies. This identifies the movie with the largest number of actors.
SELECT DISTINCT CONCAT(actor.first_name, ', actor.last_name) AS actor_name	Selects distinct actor names by concatenating first and last names, ensuring that each actor is listed only once.
FROM actor JOIN film_actor USING(actor_id) JOIN MaxCastMovie USING(film_id)	After identifying the movie with the largest cast, this joins the actor and film_actor tables, and then with the MaxCastMovie to only include actors from that specific movie.
ORDER BY actor_name ASC	Orders the list of actors alphabetically by their full name.

```
____133 ● ⊝ WITH MovieCastCounts AS (
   134
             SELECT film.film_id,COUNT(actor.actor_id) AS cast_size -- Count of actors in each movie
    135
             FROM film
    136
             JOIN film_actor using(film_id)
             join actor using(actor_id)
    137
             GROUP BY film.film_id
   138
            ),
    139
    140 

MaxCastMovie AS (
    141
             SELECT film_id
             from MovieCastCounts
    142
             WHERE cast_size = (SELECT MAX(cast_size) FROM MovieCastCounts) -- Identifies the movie with the largest cast
    143
    144
    145
             -- Final Query to list actors from the movie with the largest cast
    146
             SELECT DISTINCT CONCAT(actor.first_name, ' ', actor.last_name) AS actor_name
   147
    148
             FROM actor
             JOIN film_actor using (actor_id)
             JOIN MaxCastMovie using(film_id)
    150
             ORDER BY actor_name ASC; -- Orders the actor names alphabetically
    151
    Result Grid | Filter Rows:
                                       Export: Wrap Cell Content: IA
      actor_name
   ▶ BURT POSEY

CAMERON ZELLWEGER
      CHRISTIAN NEESON
      FAY WINSLET
      JAYNE NOLTE
      JULIA BARRYMORE
      JULIA ZELLWEGER
    Result 61 ×
```