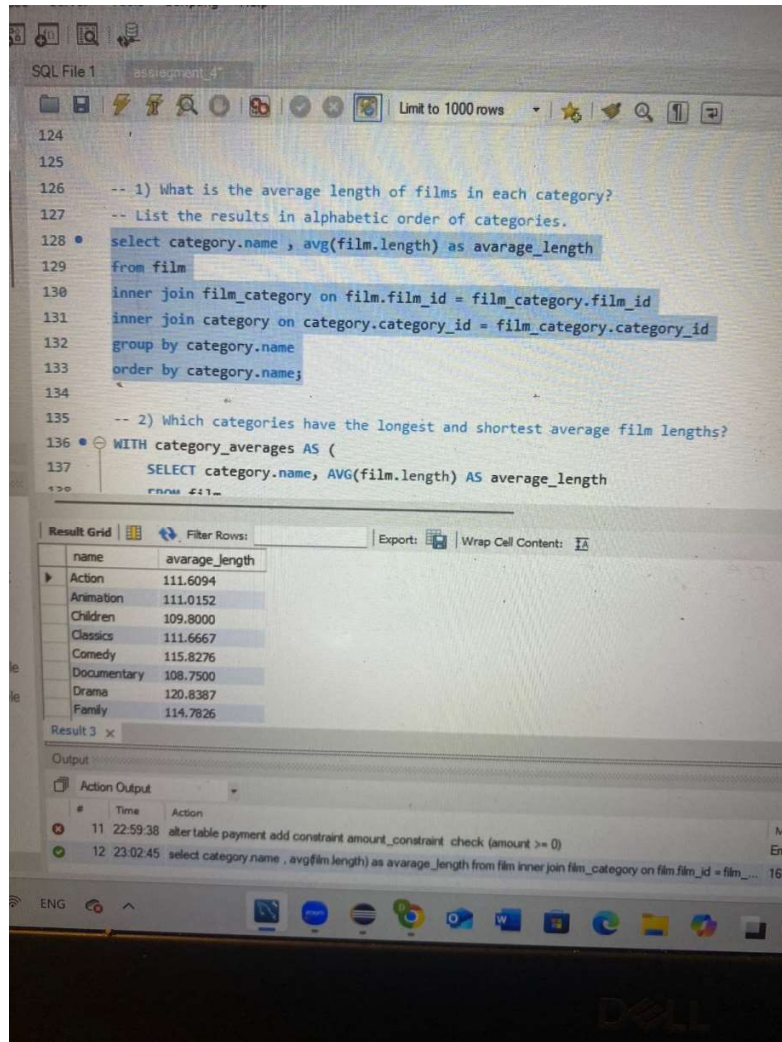


## Assignment 4

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



The screenshot shows a SQL IDE window titled 'SQL File 1' with a file named 'assignment\_4'. The query editor contains the following SQL code:

```
124
125
126 -- 1) What is the average length of films in each category?
127 -- List the results in alphabetic order of categories.
128 select category.name , avg(film.length) as avarage_length
129 from film
130 inner join film_category on film.film_id = film_category.film_id
131 inner join category on category.category_id = film_category.category_id
132 group by category.name
133 order by category.name;
134
135 -- 2) Which categories have the longest and shortest average film lengths?
136 WITH category_averages AS (
137     SELECT category.name, AVG(film.length) AS average_length
138     from film
139     inner join film_category on film.film_id = film_category.film_id
140     inner join category on category.category_id = film_category.category_id
141     group by category.name
142 )
143 select * from category_averages
144 order by average_length desc;
```

The 'Result Grid' shows the following data:

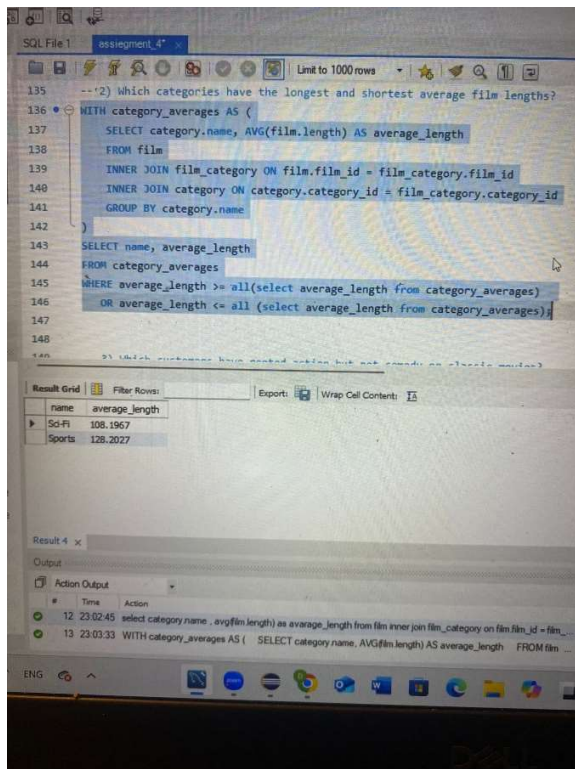
name	avarage_length
Action	111.6094
Animation	111.0152
Children	109.8000
Classics	111.6667
Comedy	115.8276
Documentary	108.7500
Drama	120.8387
Family	114.7826

The 'Output' pane shows the following log entries:

```
11 22:59:38 alter table payment add constraint amount_constraint check (amount >= 0)
12 23:02:45 select category.name , avg(film.length) as average_length from film inner join film_category on film.film_id = film_category.film_id
```

**Explanation:** This query calculates the average film length for each category. It joins the film, film\_category, and category tables to relate films with their respective categories. The AVG function computes the average length, and the results are grouped by category name. Finally, the results are sorted in alphabetical order of categories.

2) Which categories have the longest and shortest average film lengths?

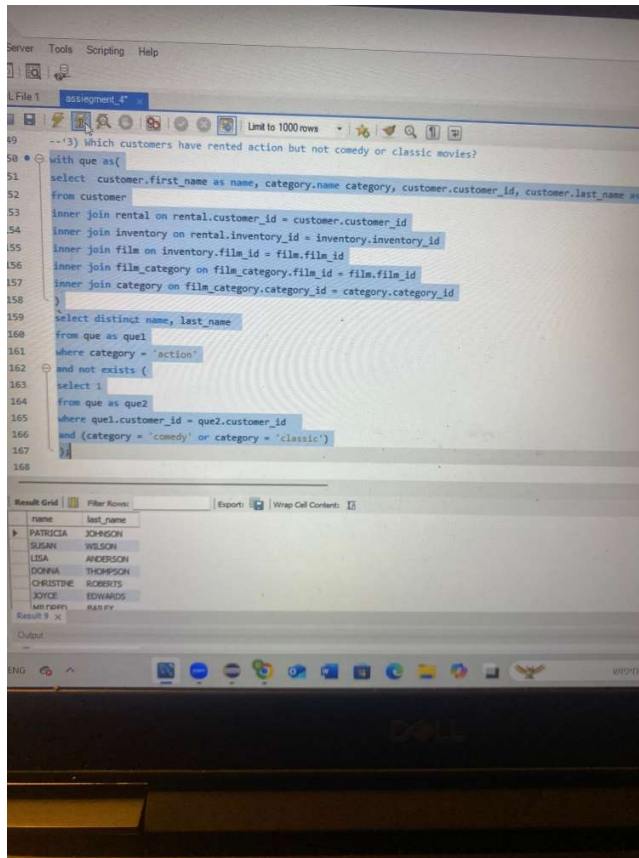


```
--2) Which categories have the longest and shortest average film lengths?
WITH category_averages AS (
  SELECT category.name, AVG(film.length) AS average_length
  FROM film
  INNER JOIN film_category ON film.film_id = film_category.film_id
  INNER JOIN category ON category.category_id = film_category.category_id
  GROUP BY category.name
)
SELECT name, average_length
FROM category_averages
WHERE average_length >= all(select average_length from category_averages)
OR average_length <= all (select average_length from category_averages);
```

name	average_length
Sci-Fi	108.1967
Sports	128.2027

**Explanation:** This query first creates a common table expression (CTE) named `category_averages` that computes the average length of films for each category. The outer query selects the categories that have the longest and shortest average lengths by comparing each category's average length against all others using the `ALL` keyword.

3) Which customers have rented action but not comedy or classic movies?



**Explanation:** This query identifies customers who have rented action films but have not rented any comedy or classic films. It uses a CTE (que) to join several tables and gather relevant data. The main query then selects distinct names and last names of customers from que who rented action films and excludes any customer who has rented films in the comedy or classic categories by checking with the NOT EXISTS clause.

4) Which actor has appeared in the most English-language movies?

The screenshot shows a SQL IDE window titled "assignment\_4". The SQL editor contains the following code:

```
164 from que as que2
165 where que1.customer_id = que2.customer_id
166 and (category = 'comedy' or category = 'classic')
167 );
168
169 -- 4) Which actor has appeared in the most English-language movies?
170 • select actor.first_name, actor.last_name, count(language.name) as times
171 from actor
172 inner join film_actor on actor.actor_id = film_actor.actor_id
173 inner join film on film_actor.film_id = film.film_id
174 inner join language on film.language_id = language.language_id
175 where language.name = 'English'
176 group by actor.first_name, actor.last_name
177 order by times desc
178 limit 1;
179
180 -- 5) How many distinct movies were rented for exactly 10 days
181 -- from the store where Mike works?
182 • select count(distinct film.title) as distinct_files
183 from rental
```

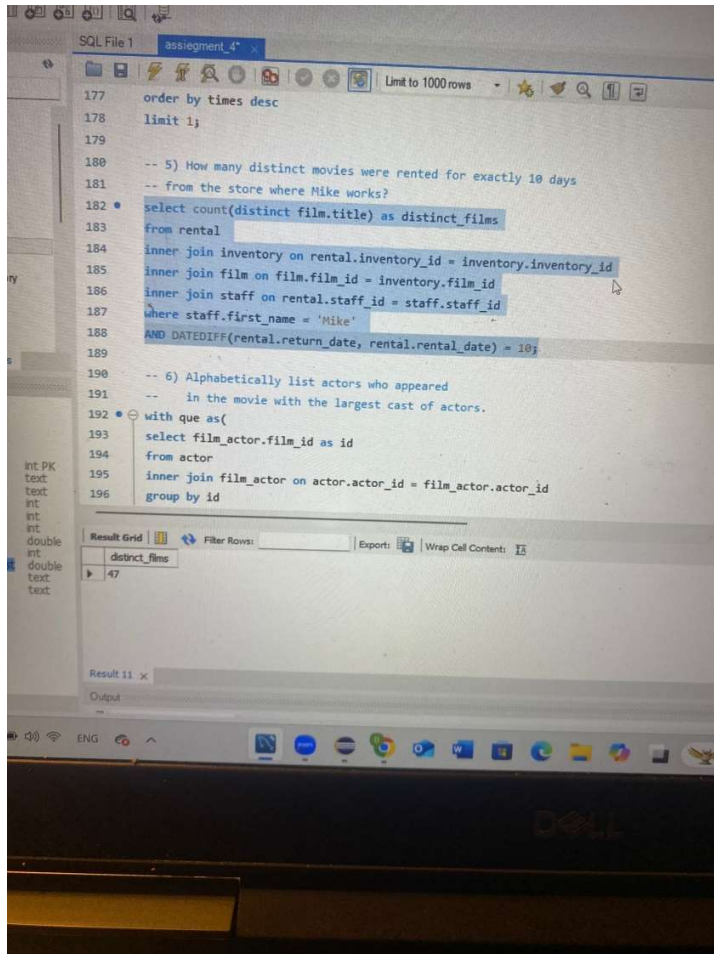
The "Result Grid" shows the following data:

first_name	last_name	times
SUSAN	DAVIS	54

The IDE interface includes a menu bar (Database, Server, Tools, Scripting, Help), a toolbar, and a status bar at the bottom.

**Explanation:** This query finds the actor who has appeared in the most English-language films. It joins the actor, film\_actor, film, and language tables. The WHERE clause filters for films with the language 'English', and the COUNT function counts the occurrences of each actor in these films. The results are grouped by actor and ordered by the count in descending order, limiting the output to the top actor.

5) How many distinct movies were rented for exactly 10 days from the store where Mike works?



**Explanation:** This query counts the distinct titles of movies that were rented for exactly 10 days from the store where Mike works. It joins the rental, inventory, film, and staff tables. The WHERE clause filters for rentals processed by staff named Mike and checks that the difference between the return and rental dates is exactly 10 days. The COUNT(DISTINCT film.title) ensures that only unique movie titles are counted.

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



```
185 inner join film on film.film_id = inventory.film_id
186 inner join staff on rental.staff_id = staff.staff_id
187 where staff.first_name = 'Mike'
188 AND DATEDIFF(rental.return_date, rental.rental_date) = 10;
189
190 -- 6) Alphabetically list actors who appeared
191 -- in the movie with the largest cast of actors.
192 with que as(
193 select film_actor.film_id as id
194 from actor
195 inner join film_actor on actor.actor_id = film_actor.actor_id
196 group by id
197 order by count(film_actor.actor_id) desc limit 1
198 )
199 select actor.first_name as name, actor.last_name
200 from actor
201 inner join film_actor on film_actor.actor_id = actor.actor_id
202 inner join que on que.id = film_actor.film_id
203 order by name;
```

Result Grid

name	last_name
BURT	POSEY
CAMERON	ZELLWEGER
CHRISTIAN	HEESON
FAY	WINSLET
JAYME	NOLTE
JULIA	BARRYMORE
WETA	WILLIAMS

**Explanation:** This query retrieves actors from the film that has the largest cast. It first creates a CTE (que) that identifies the film ID with the highest number of actors by counting the film\_actor entries. The outer query then selects the first and last names of actors who appeared in that film by joining the actor and film\_actor tables with the film ID from the CTE. Finally, the results are ordered alphabetically by the actor's first name.

ERD Diagram:

