

Database Management Systems Assignment 1

Published: 10.11.2025
Due: 24.11.25, 23:59

Data

In this exercise, We'll use the *Sakila* schema, which can be found on TAU's server. The documentation for this schema can be found [here](#). A guide for connecting to the Sakila database is available [here](#).

Requirements

For each question in part 2, you are required to provide the following:

- The SQL query you used. (.sql files)
- The query output - (in case the query exceeds 20 rows, please list only the first 20. You can use LIMIT at end of the query to get that – by the order you were asked).
- Any assumptions you made. If you think that the question can be understood in more than one way, explain according to which interpretation you solved it.

Important Notes

- Your query must return the answer to the question exactly, with no more and no fewer attributes or rows.
- Do NOT return duplicated rows in the answer unless you are specifically asked to do so.
- Do NOT use the “LIMIT” keyword in your queries, unless the final result exceeds the 20-result limit stated above, in which case you should return the top 20 results by the order you were asked or arbitrary 20 results if no such order is specified.

Submission

- Your solution should be submitted in a single zip file named YOUR-USERNAME-hw1.zip, through Moodle.
- The zip should include the answers document in pdf format, named YOUR-USERNAME-hw1.pdf
- For every questions in part 2, include a .sql file with the SQL query. Name the files q01.sql, q02.sql, and so on according to the question numbers.
- Please make sure that the queries are well formatted (use tabs and new-lines, parenthesis, etc.) to make them readable (See the example format).
- Submission is individual.

1 Explaining SQL Queries - 10 Points

Explain in your words what the following queries output. It is recommended to execute the queries on the Sakila database to validate your answers.

1.1

```
SELECT c.first_name, c.last_name, COUNT(DISTINCT country.country)
FROM rental r, inventory i, customer c, address sa,
address ca, city sc, city cc, country, store s
WHERE r.inventory_id = i.inventory_id
    AND i.store_id = s.store_id
    AND r.customer_id = c.customer_id
    AND s.address_id = sa.address_id
    AND sa.city_id = sc.city_id
    AND sc.country_id = country.country_id
    AND ca.address_id = c.address_id
    AND ca.city_id = cc.city_id
    AND cc.city = 'Tel Aviv-Jaffa'
GROUP BY c.customer_id;
```

1.2

```
SELECT film.title
FROM film,
((SELECT f.film_id
  FROM rental r1,
       rental r2,
       inventory i1,
       inventory i2,
       film f,
       film_actor fa,
       actor a
 WHERE r1.inventory_id = i1.inventory_id
   AND r2.inventory_id = i2.inventory_id
   AND i1.film_id = i2.film_id
   AND f.film_id = i1.film_id
   AND f.film_id = fa.film_id
   AND fa.actor_id = a.actor_id
   AND r1.rental_id != r2.rental_id
   AND YEAR(r1.rental_date) = '2005'
   AND YEAR(r2.rental_date) = '2005'
   AND a.first_name = 'TOM')
INTERSECT
(
(
    SELECT film_id
    FROM film
)
EXCEPT
(
    SELECT f.film_id
    FROM film f, inventory i1, inventory i2, inventory i3
    WHERE f.film_id = i1.film_id
      AND f.film_id = i2.film_id
      AND f.film_id = i3.film_id
      AND i1.inventory_id != i2.inventory_id
      AND i1.inventory_id != i3.inventory_id
      AND i2.inventory_id != i3.inventory_id
)
))
A
WHERE A.film_id = film.film_id;
```

2 Writing SQL Queries - 90 Points

1. Write an SQL query that returns all film categories for which every store has strictly more than 150 films available in that category. For each such category, also compute max_films, defined as the maximum number of films available in that category in any single store (i.e., the maximum across all stores for that category). For example: if store 1 has 100 action films and store 2 has 270 action films, then max_films for action is 270.

The results should be ordered based on max_films, such that the category with the largest max_films comes first.

The output schema should be (category, max_films).

2. Write a query that identifies actors who have acted in strictly less films than the average actor has acted in. The query should return their first and last names, along with their film counts.

The results should be ordered based on the film_count, with the one having the highest number of films coming first.

The output schema should be (fname, lname, film_count).

3. Write an SQL query that returns the actors who have acted in the fewest film categories and the actors who have acted in the most film categories (include all actors in case of ties).

The query should return the actor's first and last name, as well as the number of categories the actor has acted in

The results should be ordered by the actor's first and last name.

The output schema should be (fname, lname, num_categories)

4. Write a query that identifies film categories meeting the following criteria:

(a) Contain at least 65 films.

(b) Have an average rental duration exceeding 3 days. The average should be computed over all films in the category that have been rented and **returned**. Use *DATEDIFF* to compute the rental duration.

(c) Have at least 400 customers who rented films in this category.

Retrieve the names of these categories, along with their average rental durations, film counts and number of customers, sorted in descending order by the film count.

The output schema should be (category, avg_rental_duration, film_count, num_customers).

5. Let C be a customer and let A(C) be the actor who has participated in the largest number of films that C has rented (If there is more than one such actor, choose the one with the largest actor id).

Write a query that, for each customer C that lives in Argentina, finds A(C) and displays C's first and last name, A(C)'s name, and the number of rented movies featuring A(C).

The results should be ordered based on the customer's first and last name.

The output schema should be (customer_fname, customer_lname, actor_fname, actor_lname, numrented).

6. In this question you are **not allowed to use aggregation (i.e. GROUP BY)**.

Write a query that finds all films that have exactly three copies in the inventory. Display the title of these films along with the actor with the maximal actor_id who participated in the film.

The results should be ordered by the film title.

The output schema should be (title, fname, lname)