USE sakila;

SELECT \* FROM actor;

-- 1a. Display the first and last names of all actors from the table `actor`.

SELECT first\_name, last\_name FROM actor;

-- 1b. Display the first and last name of each actor in a single column in upper case letters. Name the column `Actor Name`.

SELECT concat(first\_name, ' ', last\_name) as 'Actor Name' FROM actor;

-- 2a. You need to find the ID number, first name, and last name of an actor, of whom you know only the first name, "Joe."

-- What is one query would you use to obtain this information?

SELECT actor\_id, first\_name, last\_name FROM actor WHERE first\_name = 'JOE';

-- 2b. Find all actors whose last name contain the letters `GEN`:

SELECT actor\_id, first\_name, last\_name FROM actor WHERE last\_name LIKE '%GEN%';

-- 2c. Find all actors whose last names contain the letters `LI`. This time, order the rows by last name and first name, in that order:

SELECT actor\_id, first\_name, last\_name FROM actor WHERE last\_name LIKE '%LI%' ORDER BY last\_name, first\_name;

-- 2d. Using `IN`, display the `country\_id` and `country` columns of the following countries: Afghanistan, Bangladesh, and China:

SELECT \* FROM country;

SELECT country\_id, country

FROM country

WHERE country IN ('Afghanistan', 'Bangladesh', 'China');

SELECT \* FROM actor;

-- 3a. You want to keep a description of each actor. You don't think you will be performing queries on a description,

-- so create a column in the table `actor` named `description` and use the data type `BLOB`

SELECT \* FROM actor;

ALTER TABLE actor ADD COLUMN description BLOB;

-- 3b. Very quickly you realize that entering descriptions for each actor is too much effort. Delete the `description` column.

SELECT \* FROM actor;

ALTER TABLE actor DROP COLUMN description;

-- 4a. List the last names of actors, as well as how many actors have that last name.

SELECT \* FROM actor;

SELECT last\_name, COUNT(\*) FROM actor GROUP BY last\_name;

-- 4b. List last names of actors and the number of actors who have that last name, but only for names that are shared by

-- at least two actors

SELECT \* FROM actor;

SELECT last\_name, COUNT(\*) as 'name' FROM actor GROUP BY last\_name HAVING COUNT(\*) >1;

-- 4c. The actor `HARPO WILLIAMS` was accidentally entered in the `actor` table as `GROUCHO WILLIAMS`. Write a query to fix the record.

UPDATE actor SET first\_name = 'HARPO' WHERE first\_name = 'GROUCHO' AND last\_name = 'WILLIAMS';

SELECT actor\_id, first\_name, last\_name FROM actor WHERE last\_name = 'WILLIAMS';

-- \*\*\*4d. Perhaps we were too hasty in changing `GROUCHO` to `HARPO`. It turns out that `GROUCHO` was the correct name after all! In a

-- single query, if the first name of the actor is currently `HARPO`, change it to `GROUCHO`.

UPDATE actor SET first\_name = 'GROUCHO' WHERE first\_name = 'HARPO';

-- \*\*\*5a. You cannot locate the schema of the `address` table. Which query would you use to re-create it?

-- SELECT \* FROM address

CREATE TABLE new\_address AS

SELECT address\_id, address, address2, district, city\_id, postal\_code, phone, location, last\_update

FROM address;

SELECT \* FROM new\_address;

-- 6a. Use `JOIN` to display the first and last names, as well as the address, of each staff member. Use the tables `staff` and `address`:

-- SELECT \* FROM address

-- SELECT \* FROM staff

SELECT staff.address\_id, staff.first\_name, staff.last\_name, address.address

FROM staff

INNER JOIN address ON

staff.address\_id = address.address\_id;

-- 6b. Use `JOIN` to display the total amount rung up by each staff member in August of 2005. Use tables `staff` and `payment`.

-- SELECT \* FROM staff

-- SELECT \* FROM payment

SELECT staff.staff\_id,

SUM(payment.amount) AS 'Sum of Payment'

FROM staff

JOIN payment USING (staff\_id)

WHERE payment\_date BETWEEN '2005/08/01' AND '2005/08/31'

GROUP BY staff\_id;

-- 6.c List each film and the number of actors who are listed for that film. Use tables `film\_actor` and `film`. Use inner join.

-- SELECT \* FROM film\_actor

-- SELECT \* FROM film

SELECT film.title,

COUNT(film\_actor.actor\_id) AS NumberOfActors

FROM film\_actor

INNER JOIN film USING (film\_id)

GROUP BY film\_id;

-- 6d. How many copies of the film `Hunchback Impossible` exist in the inventory system?

SELECT film.title,

COUNT(inventory.inventory\_id) AS 'Number of Copies'

FROM film

INNER JOIN inventory

USING (film\_id)

WHERE title = "Hunchback Impossible"

GROUP BY title;

-- 7a. The music of Queen and Kris Kristofferson have seen an unlikely resurgence. As an unintended consequence,

-- films starting with the letters `K` and `Q` have also soared in popularity. Use subqueries to display the titles of movies starting

-- with the letters `K` and `Q` whose language is English.

SELECT film.title

FROM film

WHERE title LIKE 'K%'

OR title LIKE 'Q%'

AND language\_id

IN (SELECT language\_id FROM language WHERE name = 'English');

-- 7b. Use subqueries to display all actors who appear in the film `Alone Trip`.

SELECT first\_name, last\_name FROM actor

WHERE actor\_id IN (SELECT actor\_id FROM film\_actor WHERE film\_id = (SELECT film\_id FROM film WHERE title = 'Alone Trip'));

-- 7c. You want to run an email marketing campaign in Canada, for which you will need the names and email addresses of all

-- Canadian customers Use joins to retrieve this information.

SELECT first\_name, last\_name, email

FROM customer

JOIN address USING(address\_id)

JOIN city USING(city\_id)

JOIN country USING(country\_id)

WHERE country = "Canada";

-- 7d. Sales have been lagging among young families, and you wish to target all family movies for a promotion.

-- Identify all movies categorized as \_family\_ films.

SELECT title, description, rating FROM film\_list

WHERE category = "Family";

-- 7e. Display the most frequently rented movies in descending order.

SELECT title, COUNT(\*) AS rental\_count

FROM film

JOIN inventory USING(film\_id)

JOIN rental USING(inventory\_id)

GROUP BY film\_id

HAVING rental\_count > 10

ORDER BY rental\_count DESC;

-- 7f. Write a query to display how much business, in dollars, each store brought in.

SELECT \* FROM sales\_by\_store;

-- 7g. Write a query to display for each store its store ID, city, and country.

SELECT store\_id, city, country

FROM store

JOIN address USING(address\_id)

JOIN city USING(city\_id)

JOIN country USING(country\_id);

-- 7h. List the top five genres in gross revenue in descending order. (\*\*Hint\*\*: you may need to use the following tables: category,

-- film\_category, inventory, payment, and rental.)

SELECT \* FROM sales\_by\_film\_category

ORDER BY total\_sales DESC LIMIT 5;

-- 8a. In your new role as an executive, you would like to have an easy way of viewing the Top five genres by gross revenue.

-- Use the solution from the problem above to create a view. If you haven't solved 7h, you can substitute another query to create a view.

DROP view if exists Top\_five\_genres;

CREATE view Top\_five\_genres (category, total\_sales) AS

SELECT \* FROM sales\_by\_film\_category

ORDER BY total\_sales desc

limit 5;

-- 8b. How would you display the view that you created in 8a?

SELECT \* FROM Top\_five\_genres;

-- 8c. You find that you no longer need the view `top\_five\_genres`. Write a query to delete it.

DROP VIEW Top\_five\_genres;