-- Question 1a

USE sakila;

SELECT first\_name, last\_name FROM actor;

-- Question 1b

USE sakila;

USE Sakila;

SELECT first\_name, last\_name, concat(first\_name," ", last\_name) AS Actor\_Name

FROM actor;

-- Question 2a

USE saklia;

SELECT actor\_id, first\_name, last\_name

FROM actor

WHERE first\_name = 'Joe%';

-- Question 2b

USE sakila;

SELECT last\_name

FROM actor

WHERE last\_name LIKE '%GEN%';

-- Question 2c

USE Sakila;

SELECT last\_name, first\_name

FROM actor

WHERE last\_name LIKE '%LI%'

ORDER BY last\_name, first\_name;

-- Question 2d

USE Sakila;

SELECT country\_id, country

FROM country

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

-- Question 3a

USE Sakila;

ALTER TABLE actor

ADD COLUMN middle\_name blob;

-- Question 3b

ALTER TABLE actor

MODIFY middle\_name BLOB;

-- Question 3c

ALTER TABLE actor

DROP COLUMN middle\_name;

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

SELECT last\_name, COUNT(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 last\_name, COUNT(\*) as A

FROM actor

GROUP BY last\_name

HAVING COUNT(last\_name) >=2;

-- 4c. Oh, no! The actor HARPO WILLIAMS was accidentally entered in the actor table as

-- GROUCHO WILLIAMS, the name of Harpo's second cousin's husband's yoga teacher.

-- Write a query to fix the record.

UPDATE actor

SET first\_name = 'Harpo'

WHERE first\_name = 'GROUCHO' AND last\_name like '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. Otherwise, change the first name to MUCHO GROUCHO, as that

-- is exactly what the actor will be with the grievous error. BE CAREFUL NOT TO CHANGE THE FIRST NAME

-- OF EVERY ACTOR TO MUCHO GROUCHO, HOWEVER! (Hint: update the record using a unique identifier.)--

UPDATE actor

SET first\_name = 'MUCHO GROUCHO'

AND last\_name like 'William'

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

SHOW CREATE TABLE sakila.address;

-- CREATE TABLE `address` (

-- `address\_id` smallint(5) unsigned NOT NULL AUTO\_INCREMENT,

-- `address` varchar(50) NOT NULL,

-- `address2` varchar(50) DEFAULT NULL,

-- `district` varchar(20) NOT NULL,

-- `city\_id` smallint(5) unsigned NOT NULL,

-- `postal\_code` varchar(10) DEFAULT NULL,

-- `phone` varchar(20) NOT NULL,

-- `location` geometry NOT NULL,

-- `last\_update` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

-- PRIMARY KEY (`address\_id`),

-- KEY `idx\_fk\_city\_id` (`city\_id`),

-- SPATIAL KEY `idx\_location` (`location`),

-- CONSTRAINT `fk\_address\_city` FOREIGN KEY (`city\_id`) REFERENCES `city` (`city\_id`) ON UPDATE CASCADE

-- ) ENGINE=InnoDB AUTO\_INCREMENT=606 DEFAULT CHARSET=utf8

-- 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 first\_name, last\_name, address

FROM staff

where staff.address\_id = adress.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 first\_name, last\_name, SUM(payment.amount)

FROM staff, payment

Where staff.staff\_id = payment.staff\_id

AND month(payment\_date) = 8

Group by staff\_id

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

-- Use inner join.

SELECT title, COUNT(actor\_id)

FROM film f

INNER JOIN film\_actor fa

ON f.film\_id = fa.film\_id

GROUP BY title;

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

SELECT title, COUNT(inventory\_id)

FROM film f

INNER JOIN inventory i

ON f.film\_id = i.film\_id

WHERE title = "Hunchback Impossible";

-- 6e. Using the tables payment and customer and the JOIN command, list the total paid by each customer. List the customers alphabetically

-- by last name:

SELECT last\_name, first\_name, SUM(amount)

FROM payment p

INNER JOIN customer c

ON p.customer\_id = c.customer\_id

GROUP BY p.customer\_id

ORDER BY last\_name ASC;

-- 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.

USE Sakila;

SELECT title FROM film

WHERE language\_id in

(SELECT language\_id

FROM language

WHERE name = "English" )

AND (title LIKE "K%") OR (title LIKE "Q%");

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

use sakila;

select fl.actor\_id, ac.first\_name, ac.last\_name

from actor ac

join film\_actor fl on ac.actor\_id = fl.actor\_id

join film fm on fl.film\_id = fm.film\_id

where fm.title like '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.

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

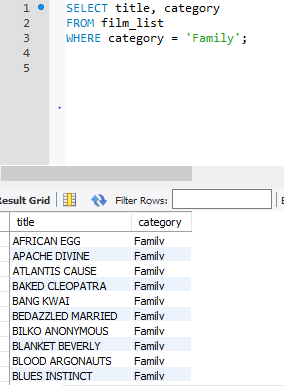
-- categorized as family films.

USE Sakila;

SELECT title, category

FROM film\_list

WHERE category = 'Family';



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

USE Sakila;

SELECT i.film\_id, f.title, COUNT(r.inventory\_id)

FROM inventory i

INNER JOIN rental r

ON i.inventory\_id = r.inventory\_id

INNER JOIN film\_text f

ON i.film\_id = f.film\_id

GROUP BY r.inventory\_id

ORDER BY COUNT(r.inventory\_id) DESC;

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

SELECT store.store\_id, SUM(amount)

FROM store

INNER JOIN staff

ON store.store\_id = staff.store\_id

INNER JOIN payment p

ON p.staff\_id = staff.staff\_id

GROUP BY store.store\_id

ORDER BY SUM(amount);

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

-- 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 sum(payment.amount)as 'sales', category.name

from payment join rental on (rental.rental\_id= payment.rental\_id)

join inventory on (inventory.inventory\_id = rental.inventory\_id)

join film\_category on (film\_category.film\_id = inventory.film\_id)

join category on (category.category\_id = film\_category.category\_id)

group by category.name

order by sum(payment.amount)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.

create view top5 as

select sum(payment.amount)as 'sales', category.name

from payment join rental on (rental.rental\_id= payment.rental\_id)

join inventory on (inventory.inventory\_id = rental.inventory\_id)

join film\_category on (film\_category.film\_id = inventory.film\_id)

join category on (category.category\_id = film\_category.category\_id)

group by category.name

order by sum(payment.amount)desc

limit 5;

select sum(payment.amount)as 'sales', category.name

from payment join rental on (rental.rental\_id= payment.rental\_id)

join inventory on (inventory.inventory\_id = rental.inventory\_id)

join film\_category on (film\_category.film\_id = inventory.film\_id)

join category on (category.category\_id = film\_category.category\_id)

group by category.name

order by sum(payment.amount)desc

limit 5;

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

create view top5 as

select sum(payment.amount)as 'sales', category.name

from payment join rental on (rental.rental\_id= payment.rental\_id)

join inventory on (inventory.inventory\_id = rental.inventory\_id)

join film\_category on (film\_category.film\_id = inventory.film\_id)

join category on (category.category\_id = film\_category.category\_id)

group by category.name

order by sum(payment.amount)desc

limit 5;

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

### Appendix: List of Tables in the Sakila DB

\* A schema is also available as `sakila\_schema.svg`. Open it with a browser to view.

```sql

'actor'

'actor\_info'

'address'

'category'

'city'

'country'

'customer'

'customer\_list'

'film'

'film\_actor'

'film\_category'

'film\_list'

'film\_text'

'inventory'

'language'

'nicer\_but\_slower\_film\_list'

'payment'

'rental'

'sales\_by\_film\_category'

'sales\_by\_store'

'staff'

'staff\_list'

'store'

```