The database table mentioned below contain customer, inventory, transactional and employee data for a fictional video rental company.

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

SELECT first\_name, last\_name

FROM actor;

Revised —

SELECT Concat(first\_name, ‘ ‘, last\_name) AS ‘Actor\_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 UPPER(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";

Revised —

SELECT actor\_id, CONCAT(last\_name, ‘, ‘ , first\_name) AS ‘Actor\_Name’

FROM actor

WHERE first\_name =‘Joe’;

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

SELECT first\_name, last\_name

FROM actor

WHERE last\_name LIKE '%GEN%';

Revised —

Select 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 last\_name, first\_name

FROM actor

WHERE last\_name LIKE '%LI%'

ORDER BY last\_name;

Revised —

SELECT last\_name, first\_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 country\_id, country

FROM country

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

#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`

#(Make sure to research the type `BLOB`, as the difference between it and `VARCHAR` are significant).

ALTER TABLE actor

ADD description BLOB;

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

ALTER TABLE actor

DROP COLUMN description;

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

SELECT DISTINCT last\_name AS "Last Name" , Count(DISTINCT last\_name) AS 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 actor names that are shared by at least two actors

SELECT last\_name, count(actor\_id) AS "# of Actors w. LN"

FROM actor

GROUP BY last\_name

HAVING count(actor\_id) > 1;

#4c. The actor `HARPO WILLIAMS` was accidentally entered in the `actor` table as `GROUCHO WILLIAMS`.

UPDATE actor

SET first\_name = "HARPO"

WHERE first\_name = "GROUCHO" and 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" and last\_name = "WILLIAMS";

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

SHOW CREATE TABLE address;

#Hint: [https://dev.mysql.com/doc/refman/5.7/en/show-create-table.html](https://dev.mysql.com/doc/refman/5.7/en/show-create-table.html)

#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

INNER JOIN address ON staff.address\_id = address.address\_id;

Revised —

SELECT CONCAT(s.first\_name, ‘ ‘, s.last\_name) AS ‘Name’, a.address as Address

FROM staff s JOIN address a ON s.address\_id = a.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 COUNT(amount), staff.staff\_id

FROM staff

INNER JOIN payment ON

staff.staff\_id = payment.staff\_id

GROUP BY staff.staff\_id ;

Revised —

SELECT s.staff\_id, SUM(p.amount) AS ‘Total’

FROM staff s JOIN payment p ON s.staff\_id = p.staff\_id

GROUP BY s.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) AS "Number of Actors in Film"

FROM film

INNER JOIN film\_actor ON

film.film\_id = film\_actor.film\_id

GROUP BY film.film\_id;

Revised —

SELECT f.title, COUNT(fa.actor\_id) AS ’# of Actors in Film’

FROM film f INNER JOIN film\_actor fa ON f.film\_id = fa.film\_id

GROUP BY f.film\_id;

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

SELECT title, count(film.film\_id) as "Copies"

FROM film

INNER JOIN inventory ON

film.film\_id = inventory.film\_id

WHERE title = "Hunchback Impossible"

GROUP BY film.film\_id;

Revised —

SELECT title, COUNT(f.film\_id) AS ‘# of Copies’

FROM film f

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

WHERE film.title = ‘Hunchback Impossible’

GROUP BY film.film\_id;

#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 c.first\_name AS "First Name", c.last\_name AS "Last Name" , SUM(p.amount) AS "Total Paid"

FROM customer c

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

GROUP BY c.customer\_id

ORDER BY c.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.

SELECT title

FROM film

WHERE film.title = "K%" OR "Q%" AND language\_id IN

(SELECT language\_id

FROM language

WHERE language.name = "English"

)

;

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

SELECT CONCAT(first\_name, last\_name) AS ‘Actor\_Name’

FROM actor

WHERE actor\_id IN

(select actor\_id

FROM film\_actor

WHERE film\_id IN

(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 CONCAT(first\_name, last\_name) AS ‘Customer\_Name’, email

From customer

WHERE address\_id IN

(select address\_id

from address

WHERE city\_id IN

(select city\_id

from city

WHERE country\_id IN

(select country\_id

from country

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

FROM film

WHERE film\_id IN

(select film\_id

from film\_category

WHERE category\_id IN

(select category\_id

from category

WHERE name = "family"

));

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

SELECT f.title, count(r.rental\_id) AS Rented

FROM film f join inventory i on f.film\_id = i.film\_id join rental r on i.inventory\_id = r.inventory\_id

GROUP BY title

ORDER BY Rented DESC;

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

SELECT s.store\_id, SUM(p.amount) AS Revenue

FROM store s join staff st on s.manager\_staff\_id = st.staff\_id join payment p on st.staff\_id = p.staff\_id

GROUP BY store\_id;

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

SELECT s.store\_id, c.city AS city, co.country as country

FROM country co join city c on co.country\_id = c.country\_id join address a on c.city\_id = a.city\_id join store s on a.address\_id = s.address\_id

GROUP BY store\_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 SUM(p.amount) as Gross\_Revenue, c.name as category

FROM category c join film\_category fc on

c.category\_id = fc.category\_id join film f on

fc.film\_id = f.film\_id join inventory i on

f.film\_id = i.film\_id join rental r on

i.inventory\_id = r.inventory\_id join

GROUP BY category

ORDER BY SUM(p.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 Top\_Grossing AS

SELECT TOP 5 SUM(p.amount) as Gross\_Revenue, c.name as category

FROM category c join film\_category fc on

c.category\_id = fc.category\_id join film f on

fc.film\_id = f.film\_id join inventory i on

f.film\_id = i.film\_id join rental r on

i.inventory\_id = r.inventory\_id join

payment p on r.rental\_id = p.rental\_id

GROUP BY category

ORDER BY Gross\_revenue desc;

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

SELECT \* FROM Top\_Grossing;

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

DROP VIEW Top\_Grossing;