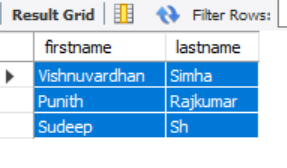
1a. Display the first and last names of all actors from the table actor.

Query: SELECT firstname, lastname FROM actors\_schema.actors;

Result:



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

Query: SELECT

firstname AS OriginalFname,

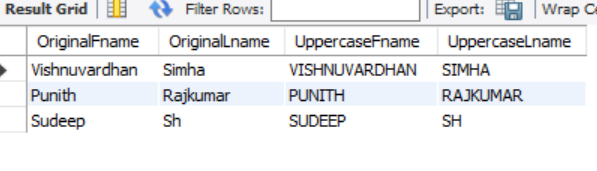
lastname AS OriginalLname,

UPPER(firstname) AS UppercaseFname,

UPPER(lastname) AS UppercaseLname

FROM actors\_schema.actors;

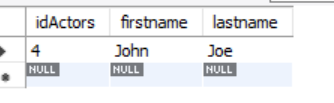
Result:



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?

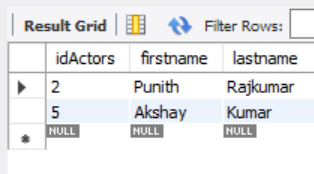
Query: SELECT \* FROM actors\_schema.actors where lastname="Joe" ;

Result:

  
  
2b. Find all actors whose last name contain the letters KUMAR:

Query: SELECT \* FROM actors\_schema.actors WHERE lastname LIKE '%kumar%';

Result:



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

Query: SELECT \* FROM actors\_schema.actors WHERE lastname LIKE '%ar%' order by firstname AND lastname;

2d. Using AR, display the country\_id and country columns of the following countries: india, america, and austrelia:

Query: select countryname, country\_id

from actors\_schema.country

left join actors\_schema.actors

on country.country\_id=actors.ac\_countryid

LIKE "%ar%";

3a. Add a middle\_name column to the table actor. Position it between first\_name and last\_name. Hint: you will need to specify the data type.

Query: ALTER TABLE actors\_schema.actors

ADD COLUMN middlename varchar(10) after firstname;  
  
3b. You realize that some of these actors have tremendously long last names. Change the data type of the middle\_name column to blobs.

Query: ALTER TABLE actors\_schema.actors

MODIFY COLUMN middlename BLOB(100);  
  
3c. Now delete the middle\_name column.

Query: ALTER TABLE actors\_schema.actors

DROP COLUMN middlename;  
  
4a. List the last names of actors, as well as how many actors have that last name.

Query: SELECT lastname FROM actors\_schema.actors;  
  
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

Query: SELECT min(lastname) FROM actors\_schema.actors where lastname="simha";  
  
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.

Query: 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. 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.)

Query: UPDATE actors\_schema.actors

SET first\_name =

CASE

WHEN first\_name = 'HARPO'

THEN 'GROUCHO'

ELSE 'MUCHO GROUCHO'

END

WHERE actor\_id = 5;  
  
5a. You cannot locate the schema of the address table. Which query would you use to re-create it?

Query: DESCRIBE sakila.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:

Query: SELECT s.first\_name, s.last\_name, a.address

FROM staff s LEFT 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.

Query: SELECT payment.staff\_id, staff.first\_name, staff.last\_name, payment.amount, payment.payment\_date

FROM staff INNER JOIN payment ON

staff.staff\_id = payment.staff\_id AND payment\_date LIKE '2005-08%';  
  
6c. List each film and the number of actors who are listed for that film. Use tables film\_actor and film. Use inner join.

Query: SELECT f.title AS 'Film Title', COUNT(fa.actor\_id) AS `Number of Actors`

FROM film\_actor fa

INNER JOIN film f

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

GROUP BY f.title;  
  
6d. How many copies of the film Hunchback Impossible exist in the inventory system?  
Query: SELECT title, (

SELECT COUNT(\*) FROM inventory

WHERE film.film\_id = inventory.film\_id

) AS 'Number of Copies'

FROM film

WHERE title = "Hunchback Impossible";

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.  
Query: SELECT title

FROM film WHERE title

LIKE 'K%' OR title LIKE 'Q%'

AND title IN

(

SELECT title

FROM film

WHERE language\_id = 1

);

7b. Use subqueries to display all actors who appear in the film Alone Trip.  
Query: SELECT first\_name, last\_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.  
Query: SELECT cus.first\_name, cus.last\_name, cus.email

FROM customer cus

JOIN address a

ON (cus.address\_id = a.address\_id)

JOIN city cty

ON (cty.city\_id = a.city\_id)

JOIN country

ON (country.country\_id = cty.country\_id)

WHERE country.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 famiy films.  
Query: SELECT title, description 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.  
Query: SELECT f.title, COUNT(rental\_id) AS 'Times Rented'

FROM rental r

JOIN inventory i

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

JOIN film f

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

GROUP BY f.title

ORDER BY `Times Rented` DESC;

7f. Write a query to display how much business, in dollars, each store brought in.  
Query: SELECT s.store\_id, SUM(amount) AS 'Revenue'

FROM payment p

JOIN rental r

ON (p.rental\_id = r.rental\_id)

JOIN inventory i

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

JOIN store s

ON (s.store\_id = i.store\_id)

GROUP BY s.store\_id;

7g. Write a query to display for each store its store ID, city, and country.  
Query: SELECT s.store\_id, cty.city, country.country

FROM store s

JOIN address a

ON (s.address\_id = a.address\_id)

JOIN city cty

ON (cty.city\_id = a.city\_id)

JOIN country

ON (country.country\_id = cty.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.)  
Query: SELECT c.name AS 'Genre', SUM(p.amount) AS 'Gross'

FROM category c

JOIN film\_category fc

ON (c.category\_id=fc.category\_id)

JOIN inventory i

ON (fc.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 c.name ORDER BY Gross 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.

Query: CREATE VIEW genre\_revenue AS

SELECT c.name AS 'Genre', SUM(p.amount) AS 'Gross'

FROM category c

JOIN film\_category fc

ON (c.category\_id=fc.category\_id)

JOIN inventory i

ON (fc.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 c.name ORDER BY Gross LIMIT 5;

8b. How would you display the view that you created in 8a?  
Query: SELECT \* FROM genre\_revenue;

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

Query: DROP VIEW genre\_revenue;