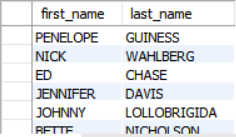
Ahmed Mohammed

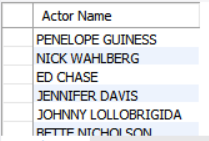
1a. Display the first and last names of all actors from the table actor.

**SELECT first\_name,last\_name FROM actor**



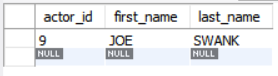
1b. Display the first and last name of each actor in a single column in upper case letters. Name the column Actor Name.

**SELECT CONCAT(first\_name, ' ', last\_name) as `Actor Name` FROM actor**



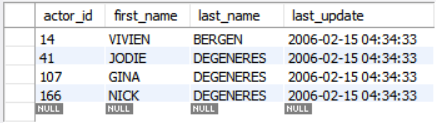
2a. You need to find the ID number, first name, and last name of an actor, of whom you know only the first name, "Joe." What is one query would you use to obtain this information?

**SELECT actor\_id,first\_name,last\_name FROM actor WHERE first\_name = 'Joe'**



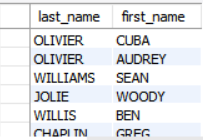
2b. Find all actors whose last name contain the letters GEN:

**SELECT \* FROM actor WHERE last\_name LIKE '%GEN%';**



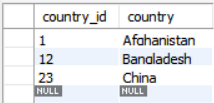
2c. Find all actors whose last names contain the letters LI. This time, order the rows by last name and first name, in that order:

**SELECT last\_name,first\_name FROM actor WHERE last\_name LIKE '%LI%';**



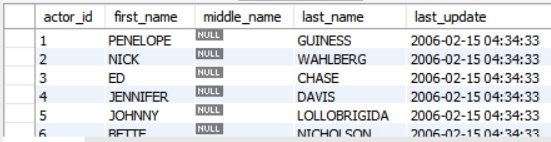
2d. Using IN, display the country\_id and country columns of the following countries: Afghanistan, Bangladesh, and China:

**SELECT country\_id,country FROM country WHERE country IN ('Afghanistan','Bangladesh','China');**



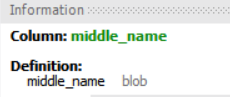
3a. Add a middle\_name column to the table actor. Position it between first\_name and last\_name. Hint: you will need to specify the data type.

**ALTER TABLE `sakila`.`actor` ADD COLUMN `middle\_name` VARCHAR(45) NULL AFTER `first\_name`;**



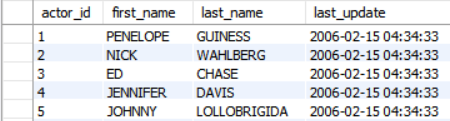
3b. You realize that some of these actors have tremendously long last names. Change the data type of the middle\_name column to blobs.

**ALTER TABLE `sakila`.`actor` MODIFY middle\_name Blob;**



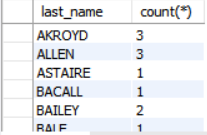
3c. Now delete the middle\_name column.

**ALTER TABLE `sakila`.`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(\*) FROM actor GROUP BY last\_name**



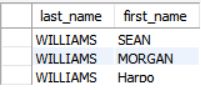
4b. List last names of actors and the number of actors who have that last name, but only for names that are shared by at least two actors

**SELECT last\_name, count(\*) FROM actor GROUP BY last\_name HAVING COUNT(\*)>=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 = 'WILLIAMS';**

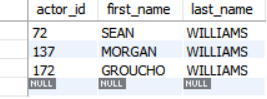


4d. Perhaps we were too hasty in changing GROUCHO to HARPO. It turns out that GROUCHO was the correct name after all! In a single query, if the first name of the actor is currently HARPO, change it to GROUCHO. Otherwise, change the first name to MUCHO GROUCHO, as that is exactly what the actor will be 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**

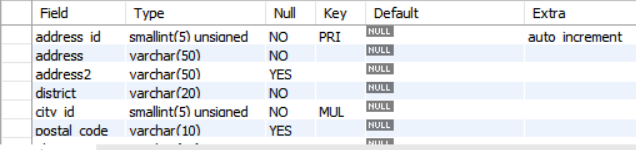
**SET first\_name = CASE WHEN first\_name = 'HARPO' THEN 'GROUCHO'ELSE 'MUCHO GROUCHO' END**

**WHERE actor\_id = 172;**



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

**DESCRIBE sakila.address**



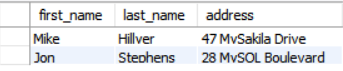
6a. Use JOIN to display the first and last names, as well as the address, of each staff member. Use the tables staff and address:

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

**FROM staff**

**INNER JOIN address ON**

**staff.staff\_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.

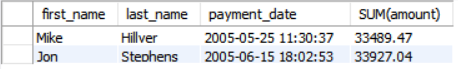
**SELECT staff.first\_name, staff.last\_name, payment.payment\_date, SUM(amount)**

**FROM staff**

**JOIN payment**

**ON staff.staff\_id = payment.staff\_id**

**GROUP BY staff.staff\_id, first\_name, last\_name**



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

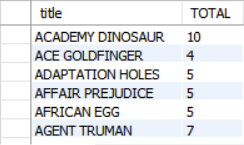
**SELECT film.title, COUNT(film\_actor.actor\_id)**

**FROM film**

**INNER JOIN film\_actor**

**ON film.film\_id = film\_actor.film\_id**

**GROUP BY film.title;**



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

**SELECT title, COUNT(inventory\_id)**

**FROM film**

**INNER JOIN inventory**

**ON film.film\_id = inventory.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 customer.last\_name, customer.first\_name, SUM(payment.amount)**

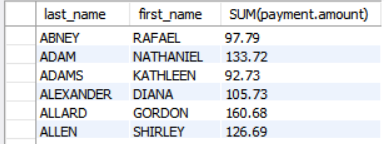
**FROM customer**

**INNER JOIN payment**

**ON customer.customer\_id = payment.customer\_id**

**GROUP BY payment.customer\_id**

**ORDER BY customer.last\_name;**

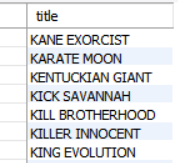


7a. The music of Queen and Kris Kristofferson have seen an unlikely resurgence. As an unintended consequence, films starting with the letters K and Q have also soared in popularity. Use subqueries to display the titles of movies starting with the letters K and Q whose language is English.

**SELECT title FROM film**

**WHERE (title LIKE 'K%' OR title LIKE 'Q%')**

**AND language\_id in (SELECT language\_id FROM language WHERE name = "English" );**



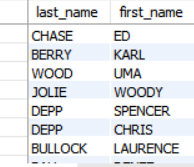
7b. Use subqueries to display all actors who appear in the film Alone Trip.

**SELECT first\_name, last\_name**

**FROM actor WHERE actor\_id in**

**(SELECT actor\_id FROM film\_actor WHERE film\_id 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.

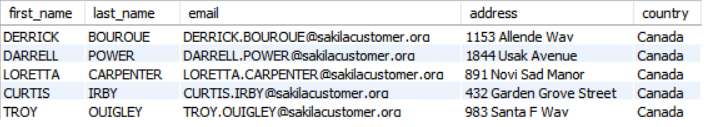
**SELECT customer.first\_name, customer.last\_name, customer.email, address.address, country.country FROM customer**

**LEFT JOIN address ON customer.address\_id = address.address\_id**

**LEFT JOIN city ON address.city\_id = city.city\_id**

**LEFT JOIN country ON city.country\_id = country.country\_id**

**WHERE country.country = 'Canada';**

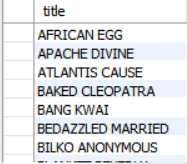


7d. Sales have been lagging among young families, and you wish to target all family movies for a promotion. Identify all movies categorized as famiy films.

**SELECT title**

**FROM film\_list**

**WHERE category = 'Family';**



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

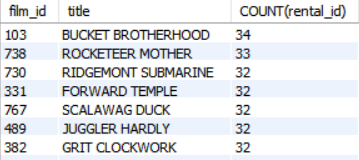
**SELECT film.film\_id, film.title, COUNT(rental\_id) FROM film**

**JOIN inventory ON film.film\_id = inventory.film\_id**

**JOIN rental ON rental.inventory\_id = inventory.inventory\_id**

**GROUP BY film.film\_id, film.title**

**ORDER BY COUNT(rental\_id) DESC;**



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

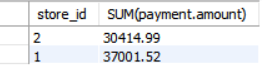
**SELECT store.store\_id, SUM(payment.amount) FROM store**

**JOIN customer ON customer.store\_id = store.store\_id**

**JOIN payment ON payment.customer\_id = customer.customer\_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.

**SELECT store.store\_id, country.country, city.city FROM store**

**JOIN address ON store.address\_id = address.address\_id**

**JOIN city ON city.city\_id = address.city\_id**

**JOIN country ON city.country\_id = country.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 category.name, SUM(payment.amount) FROM category**

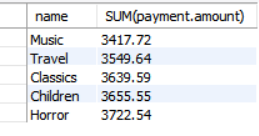
**JOIN film\_category ON (category.category\_id=film\_category.category\_id)**

**JOIN inventory ON (film\_category.film\_id=inventory.film\_id)**

**JOIN rental ON (inventory.inventory\_id=rental.inventory\_id)**

**JOIN payment ON (rental.rental\_id=payment.rental\_id)**

**GROUP BY category.name ORDER BY SUM(payment.amount) 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.

**SELECT category.name, SUM(payment.amount)**

**FROM category**

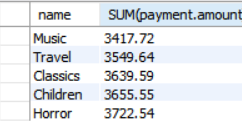
**JOIN film\_category ON (category.category\_id=film\_category.category\_id)**

**JOIN inventory ON (film\_category.film\_id=inventory.film\_id)**

**JOIN rental ON (inventory.inventory\_id=rental.inventory\_id)**

**JOIN payment ON (rental.rental\_id=payment.rental\_id)**

**GROUP BY category.name ORDER BY SUM(payment.amount) LIMIT 5;**



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

**SELECT \* FROM top\_five**

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

**DROP VIEW top\_five;**