|  |  |
| --- | --- |
| 1 | sort the customer table according to first\_name  Query: select \*from sakila.customer order by first\_name asc |
|  | Count the records in customer table  Query: select count(\*) from sakila.country; |
|  | Display the first and last names of all actors from the table actor.  Query: select first\_name, last\_name from sakila.actor; |
|  | Display the first and last name of each actor in a single column in upper case letters. Name the column Actor Name.  Query: select upper(concat(first\_name, ' ', last\_name)) as `Actor Name` from sakila.actor; |
| 2 |  |
|  | 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 actor\_id, first\_name, last\_name from sakila.actor where first\_name = "Joe"; |
|  | Find all actors whose last name contain the letters GEN:  Query: select actor\_id, first\_name, last\_name from sakila.actor where last\_name like '%GEN%'; |
|  | Find all actors whose last names contain the letters LI. This time, order the rows by last name and first name, in that order:  Query: select actor\_id, last\_name, first\_name from sakila.actor where last\_name like '%LI%'; |
|  | Using IN, display the country\_id and country columns of the following countries: Afghanistan, Bangladesh, and China:  Query: select country\_id, country from sakila.country where country in ('Afghanistan', 'Bangladesh', 'China'); |
|  |  |
| 3 | 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 sakila.actor add column middle\_name varchar(25) after first\_name; |
|  | 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 sakila.actor modify column middle\_name BLOB; |
|  | Now delete the middle\_name column.  Query: alter table actor drop column middle\_name; |
|  |  |
| 4 | List the last names of actors, as well as how many actors have that last name.  Query: select last\_name, count(\*) as 'Number of Actors' from sakila.actor group by last\_name; |
|  | 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 last\_name, count(\*) as 'Number of Actors' from sakila.actor group by last\_name having count(\*) >=2; |
|  | 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 sakila.actor set first\_name = 'HARPO' where First\_name = "Groucho" and last\_name = "Williams"; |
|  | 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 sakila.actor set first\_name = 'GROUCHO' where actor\_id = 172; |
|  |  |
| 5 | You cannot locate the schema of the address table. Which query would you use to re-create it?  Query: DESCRIBE sakila.address; |
| 6 | 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 first\_name, last\_name, address FROM sakila.staff s JOIN sakila.address a ON s.address\_id = a.address\_id; |
|  | 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%'; |
|  | 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; |
|  | How many copies of the film Hunchback Impossible exist in the inventory system?  Query: SELECT title, (SELECT COUNT(\*) FROM sakila.inventory WHERE film.film\_id = inventory.film\_id) AS 'Number of Copies' FROM sakila.film WHERE title = "Hunchback Impossible"; |
|  | Using the tables payment and customer and the JOIN command, list the total paid by each customer. List the customers alphabetically by last name:  Query: SELECT c.first\_name, c.last\_name, sum(p.amount) AS `Total Paid` FROM sakila.customer c JOIN sakila.payment p ON c.customer\_id= p.customer\_id GROUP BY c.last\_name; |
|  |  |
| 7 | 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 sakila.film WHERE title LIKE 'K%' OR title LIKE 'Q%' AND title IN (SELECT title FROM sakila.film WHERE language\_id = 1); |
|  | Use subqueries to display all actors who appear in the film Alone Trip.  Query: SELECT first\_name, last\_name FROM sakila.actor WHERE actor\_id IN (Select actor\_id FROM sakila.film\_actor WHERE film\_id IN (SELECT film\_id FROM sakila.film WHERE title = 'Alone Trip')); |
|  | 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 sakila.customer cus JOIN sakila.address a ON (cus.address\_id = a.address\_id) JOIN sakila.city cty ON (cty.city\_id = a.city\_id) JOIN sakila.country ON (country.country\_id = cty.country\_id) WHERE country.country= 'Canada'; |
|  | 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 sakila.film WHERE film\_id IN (SELECT film\_id FROM sakila.film\_category WHERE category\_id IN (SELECT category\_id FROM sakila.category WHERE name = "Family")); |
|  | Display the most frequently rented movies in descending order.  Query: SELECT f.title, COUNT(rental\_id) AS 'Times Rented' FROM sakila.rental r JOIN sakila.inventory i ON (r.inventory\_id = i.inventory\_id) JOIN sakila.film f ON (i.film\_id = f.film\_id) GROUP BY f.title ORDER BY `Times Rented` DESC; |
|  | Write a query to display how much business, in dollars, each store brought in.  **Important note regarding Sakila table structure from**[**MySQL Documentation**](https://downloads.mysql.com/docs/sakila-en.pdf)**:**   * ***5.1.6 The customer Table***   + *The customer table contains a list of all customers.*   + *The customer table is referred to in the payment and rental tables and refers to the address and store tables using foreign keys.*     - *store\_id: A foreign key identifying the customer's “home store.” Customers are not limited to renting only from this store, but this is the store they generally shop at.*   Query: SELECT s.store\_id, SUM(amount) AS 'Revenue' FROM sakila.payment p JOIN sakila.rental r ON (p.rental\_id = r.rental\_id) JOIN sakila.inventory i ON (i.inventory\_id = r.inventory\_id) JOIN sakila.store s ON (s.store\_id = i.store\_id) GROUP BY s.store\_id; |
|  | Write a query to display for each store its store ID, city, and country.  Query: SELECT s.store\_id, cty.city, country.country FROM sakila.store s JOIN sakila.address a ON (s.address\_id = a.address\_id) JOIN sakila.city cty ON (cty.city\_id = a.city\_id) JOIN sakila.country ON (country.country\_id = cty.country\_id); |
|  | 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 sakila.category c JOIN sakila.film\_category fc ON (c.category\_id=fc.category\_id) JOIN sakila.inventory i ON (fc.film\_id=i.film\_id) JOIN sakila.rental r ON (i.inventory\_id=r.inventory\_id) JOIN sakila.payment p ON (r.rental\_id=p.rental\_id) GROUP BY c.name ORDER BY Gross LIMIT 5; |
|  |  |
|  |  |
| 8 | 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: USE sakila;  create view top\_5\_genre\_revenue as  SELECT c.name as 'Film', sum(p.amount) as 'Gross Revenue' from category as c join film\_category as fc on fc.category\_id = c.category\_id join inventory as i on i.film\_id = fc.film\_id join rental as r on r.inventory\_id = i.inventory\_id join payment as p on p.rental\_id = r.rental\_id group by c.name order by sum(p.amount) desc limit 5; |
|  | How would you display the view that you created in 8a?  Query: SELECT \* FROM top\_5\_genre\_revenue; |
|  | You find that you no longer need the view top\_five\_genres. Write a query to delete it.  Query: drop view top\_5\_genre\_revenue; |