\* 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 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";**

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

**select actor\_id, first\_name, 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 \***

**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. 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 actor**

**add middle\_name varchar(45) 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 actor**

**modify column middle\_name blob;**

\* 3c. Now delete the `middle\_name` column.

**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)**

**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(last\_name)**

**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='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.)

**IF((SELECT first\_name, last\_name where first\_name='HARPO' and last\_name='williams' from actor),**

**UPDATE actor SET first\_name='GROUCHO',**

**UPDATE actor SET first\_name='MUCHO GROUCHO');**

**\*NOTE: Logic is simple, but I’ve struggled with the syntax of this one, I couldn’t get it to work.**

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

**CREATE TABLE address(**

**address\_id integer(5) AUTO\_INCREMENT NOT NULL,**

**address VARCHAR(50),**

**address2 VARCHAR(50),**

**district varchar(20),**

**citiy\_id integer(5),**

**postal\_code varchar(10),**

**phone varchar(20),**

**location varchar(30),**

**last\_update timestamp,**

**PRIMARY KEY (address\_id)**

**);**

\* 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 JOIN address ON staff.address\_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, sum(amount) as amount**

**FROM staff**

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

**where payment.payment\_date between '2005-08-01' and '2005-08-31'**

**group by staff.first\_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 count(film.title)**

**from film**

**join inventory on film.film\_id = inventory.film\_id**

**where film.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, sum(payment.amount)**

**from customer**

**join payment on customer.customer\_id=payment.customer\_id**

**group 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 film.title, l.name**

**from film**

**join**

**(select name, language\_id**

**from language**

**where name='English'**

**) as l**

**on film.language\_id=l.language\_id**

**where film.title like 'K%' or film.title like 'Q%';**

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

**select a.first\_name, a.last\_name, f.title**

**from actor a**

**join film\_actor fa on a.actor\_id=fa.actor\_id**

**join film f on f.film\_id=fa.film\_id**

**where f.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 cust.email, cont.country**

**from customer cust**

**join address addr on cust.address\_id=addr.address\_id**

**join city c on c.city\_id=addr.city\_id**

**join country cont on cont.country\_id=c.country\_id**

**where cont.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 f.title, c.name**

**from film f**

**join film\_category fc on f.film\_id=fc.film\_id**

**join category c on c.category\_id=fc.category\_id**

**where c.name='family'**

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

**select f.title, count(rent.rental\_id) as rented**

**from film f**

**join inventory inv on f.film\_id=inv.film\_id**

**join rental rent on rent.inventory\_id=inv.inventory\_id**

**group by f.title**

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

**select stre.store\_id, count(pmt.amount) as amt**

**from store stre**

**join staff st on stre.store\_id=st.store\_id**

**join payment pmt on st.staff\_id=pmt.staff\_id**

**group by stre.store\_id;**

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

**select stre.store\_id, c.city, cntr.country**

**from store stre**

**join address addr on stre.address\_id=addr.address\_id**

**join city c on c.city\_id=addr.city\_id**

**join country cntr on cntr.country\_id=c.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.)

\* 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.

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

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