**Josh Holloway**

**mySQL Install**: <https://youtu.be/0cNOD1KMJ40>

# **Assignment-1: MySQL Countries**

**Queries**

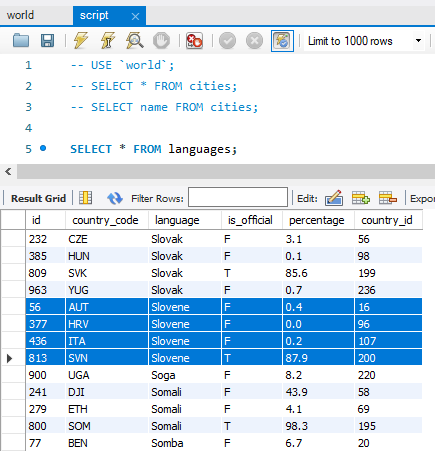
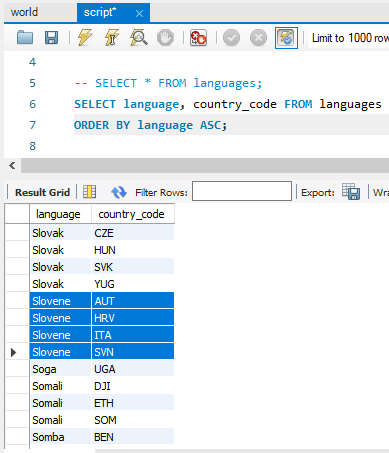
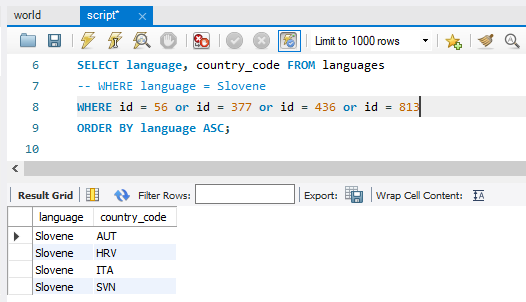
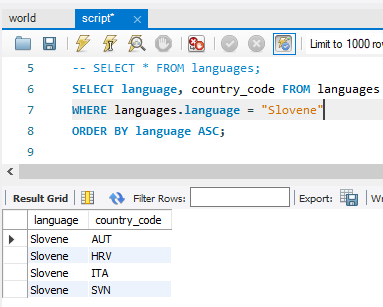
1. What query would you run to get all the **countries** that **speak Slovene**?

* Your query should return:
  + Name of the country
  + Language
  + Language percentage
* Your query should arrange the result by language percentage in descending order. (1)

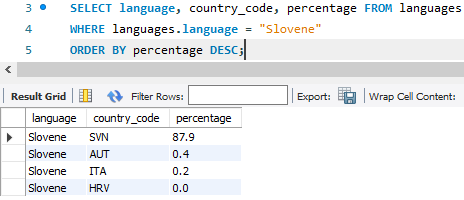
SELECT language, country\_code, percentage FROM languages

WHERE languages.language = "Slovene"

ORDER BY percentage DESC;

Steps taken to arrive at solution



2. What query would you run to display:

* **The total number of cities for each country**?
* Your query should return the **name of the country** and the **total number of cities**.
* Your query should arrange the result by the **number of cities** in **descending order**. (3)

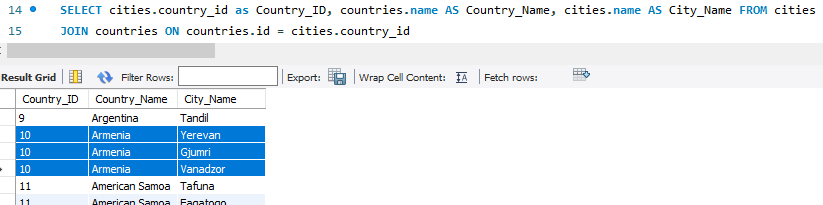
SELECT cities.country\_id as Country\_ID, countries.name AS Country\_Name, COUNT(\*) AS Number\_of\_Cities FROM cities

JOIN countries ON countries.id = cities.country\_id

GROUP BY countries.name

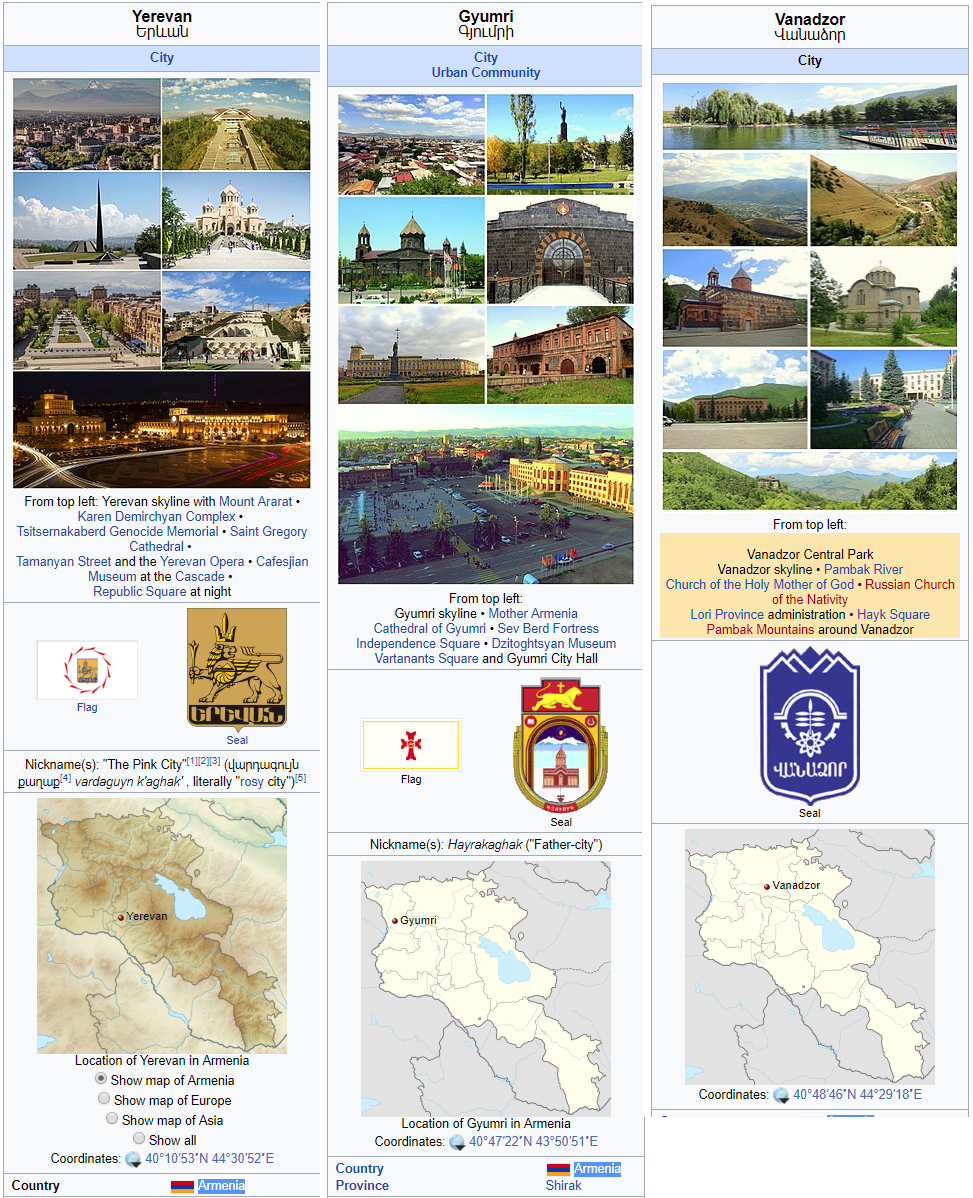
ORDER BY Number\_of\_Cities DESC;





- - - - - - - - - - - - - - - - - - - - - - - - - -





Cities used as a sanity check. [They are the 3 cities listed for Armenia]

3. What query would you run to get

* **all the cities**
* **in Mexico**
* **with a population of greater than 500,000**?
* Your query should arrange the result by population in descending order.

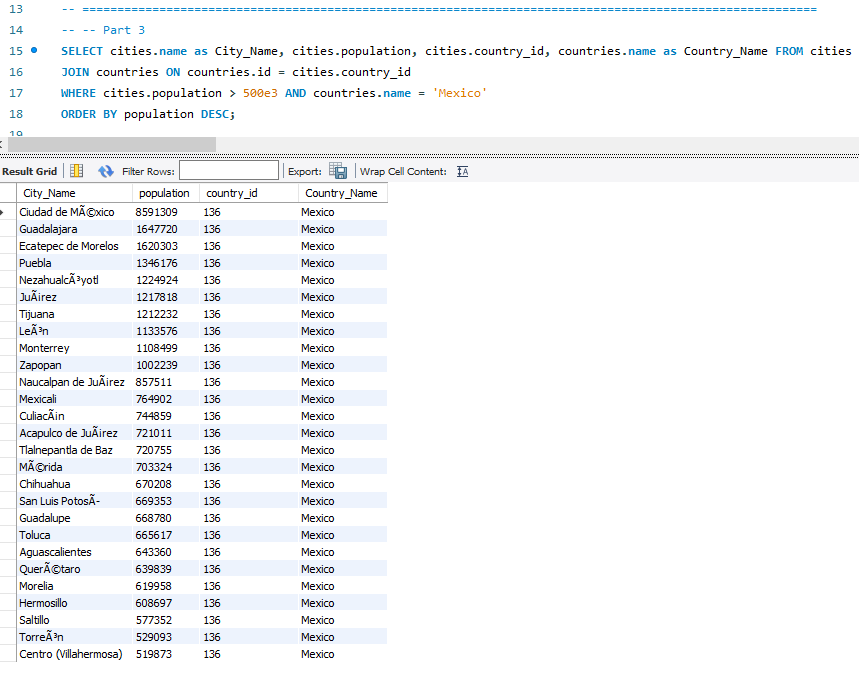
SELECT **cities**.name as City\_Name, **cities**.population, **cities**.country\_id, countries.name as Country\_Name FROM **cities**

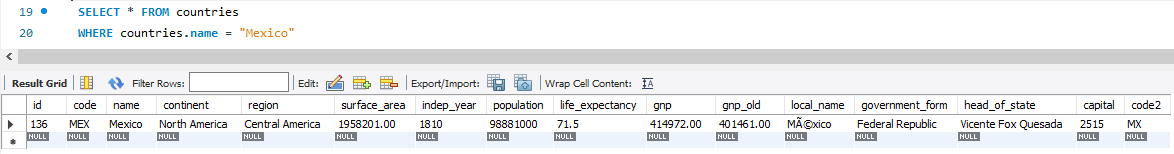
JOIN **countries** ON **countries**.id = **cities**.country\_id

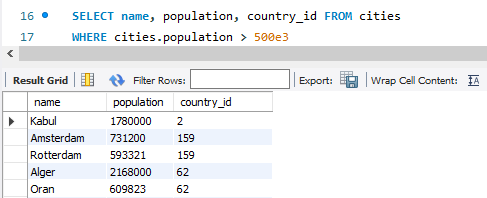
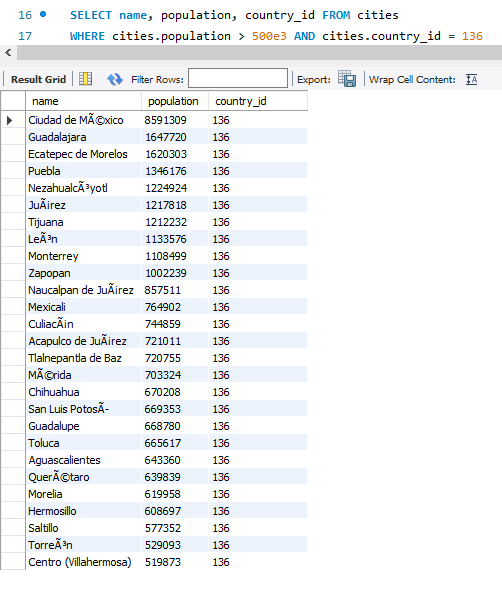
WHERE **cities**.population > 500e3 AND **countries**.name = 'Mexico'

ORDER BY population DESC;







Steps taken to work toward final solution.

4. What query would you run to get **all languages in each country with a percentage greater than 89%**?

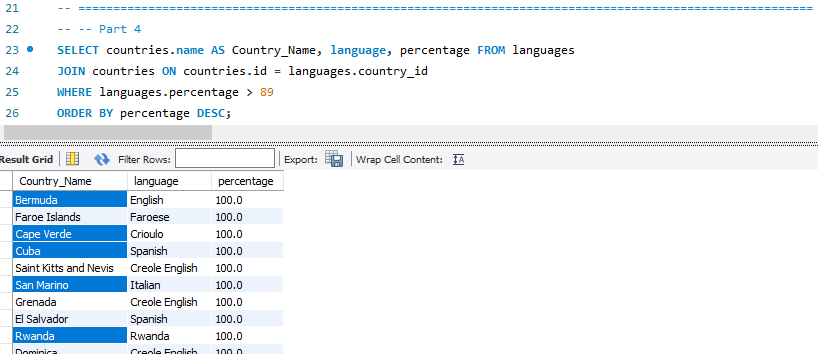
* Your query should arrange the result by percentage in descending order. (1)

SELECT countries.name AS Country\_Name, language, percentage FROM languages

JOIN countries ON countries.id = languages.country\_id

WHERE languages.percentage > 89

ORDER BY percentage DESC;

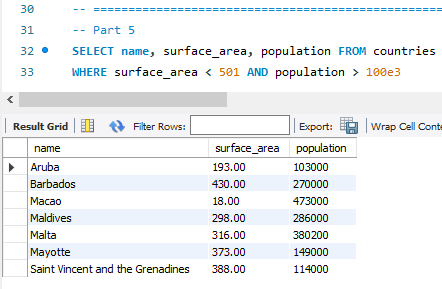


5. What query would you run to get:

* All the countries
* With Surface Area below 501
* And population greater than 100,000?

SELECT name, surface\_area, population FROM countries

WHERE surface\_area < 501 AND population > 100e3



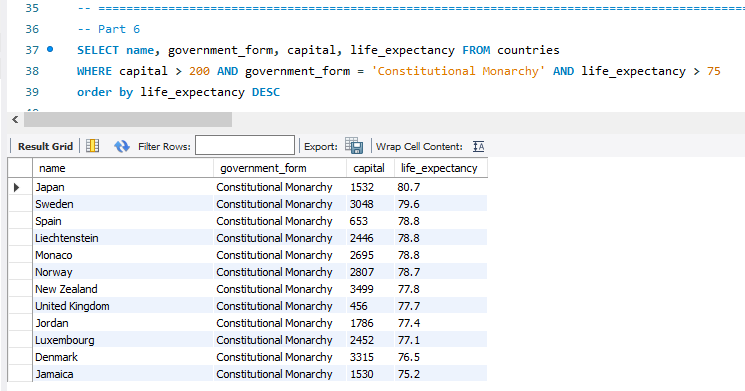
6. What query would you run to get:

* Countries
* With only Constitutional Monarchy
* With a capital greater than 200
* And a life expectancy greater than 75 years?

SELECT name, government\_form, capital, life\_expectancy FROM countries

WHERE capital > 200 AND government\_form = 'Constitutional Monarchy' AND life\_expectancy > 75

order by life\_expectancy DESC



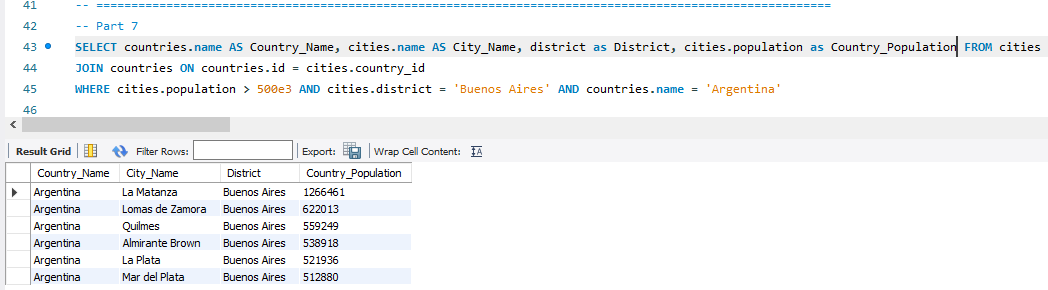
7. What query would you run to get:

* All the cities of Argentina
* Inside the Buenos Aires district
* And have the population greater than 500, 000?
* The query should return the Country Name, City Name, District and Population.

SELECT countries.name AS Country\_Name, cities.name AS City\_Name, district as District, cities.population as Country\_Population FROM cities

JOIN countries ON countries.id = cities.country\_id

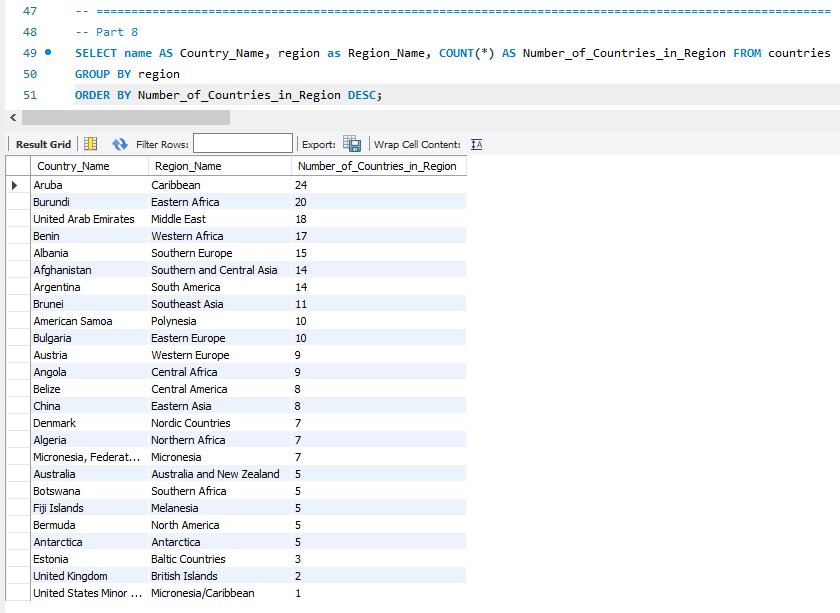
WHERE cities.population > 500e3 AND cities.district = 'Buenos Aires' AND countries.name = 'Argentina'

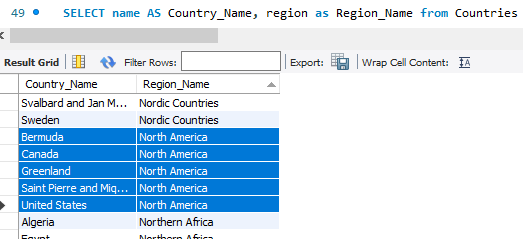


8. What query would you run to:

* Summarize the number of countries in each region?
* The query should display the name of the region and the number of countries.
* Also, the query should arrange the result by the number of countries in descending order. (2)







* **Showing that there are 5-countries in North-America region**
* **A few other regions were tested also**

# Assignment-2: ***Sakila***

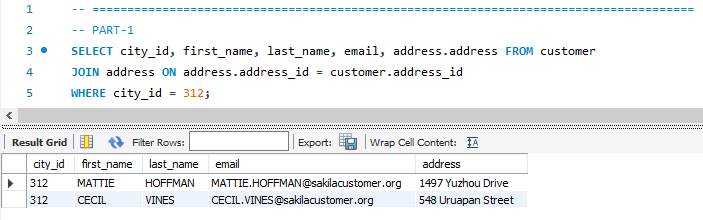
1. What query :

* All the customers
* Inside city\_id = 312
* Your query should return customer
  + first name
  + last name
  + email
  + address.

SELECT city\_id, first\_name, last\_name, email, address.address FROM customer

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

WHERE city\_id = 312;



2. What query would you run to get

* All comedy films
* Your query should return:
  + - film title
    - description
    - release year
    - rating
    - special features
    - genre (category)

SELECT film.film\_id, title AS Film\_Title, description, release\_year, rating, special\_features, name AS Genre FROM film\_category

JOIN film ON film.film\_id = film\_category.film\_id

JOIN category ON category.category\_id = film\_category.category\_id



3. What query would you run to get

* + All the films joined by actor\_id=5
* Your query should return
  + Actor id
  + Actor name
  + Film title
  + Description
  + Release year.

SELECT film\_actor.actor\_id, first\_name, last\_name, film.film\_id, title, film.description, film.release\_year FROM film\_actor

JOIN film ON film.film\_id = film\_actor.film\_id

JOIN actor ON actor.actor\_id = film\_actor.actor\_id

WHERE actor.actor\_id = 5



4. What query would you run to get:

* + All the customers in store\_id = 1
  + and inside these cities (1, 42, 312 and 459)
* Your query should return
  + first name
  + last name
  + email
  + address

SELECT \* FROM customer

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

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

WHERE store\_id = 1 and city.city\_id = 1 or city.city\_id = 42 or city.city\_id = 312 or city.city\_id = 459



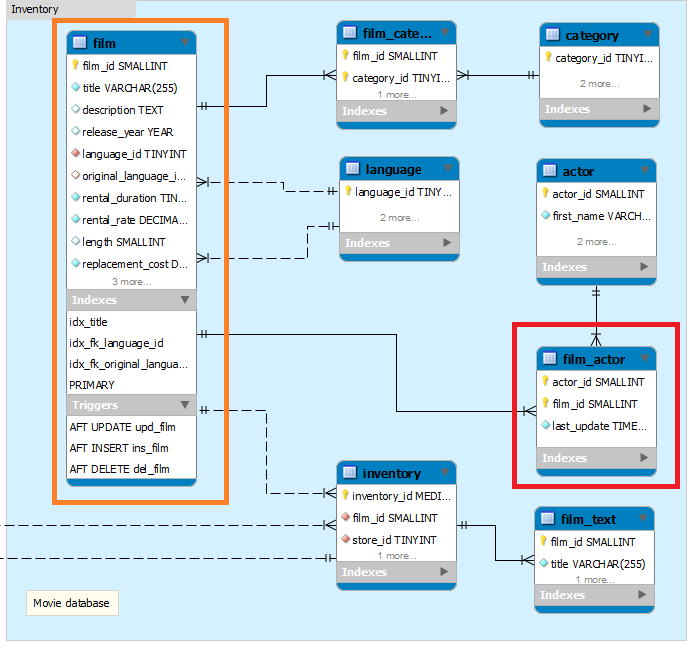
5. What query would you run to get

* + **all the films**
  + **with a "rating = G"**
  + and **"special feature = behind the scenes"**,
  + **joined by actor\_id = 15**
* Your query should return the film title
  + Description
  + Release Year
  + Rating
  + Special Feature.
* Hint: You may use LIKE function in getting the 'behind the scenes' part.

SELECT \* FROM film\_actor

JOIN film ON film.film\_id = film\_actor.film\_id

WHERE film\_actor.actor\_id LIKE 15 AND film.rating LIKE 'G' AND film.special\_features-- LIKE 'behind the scenes'



6. What query would you run to get

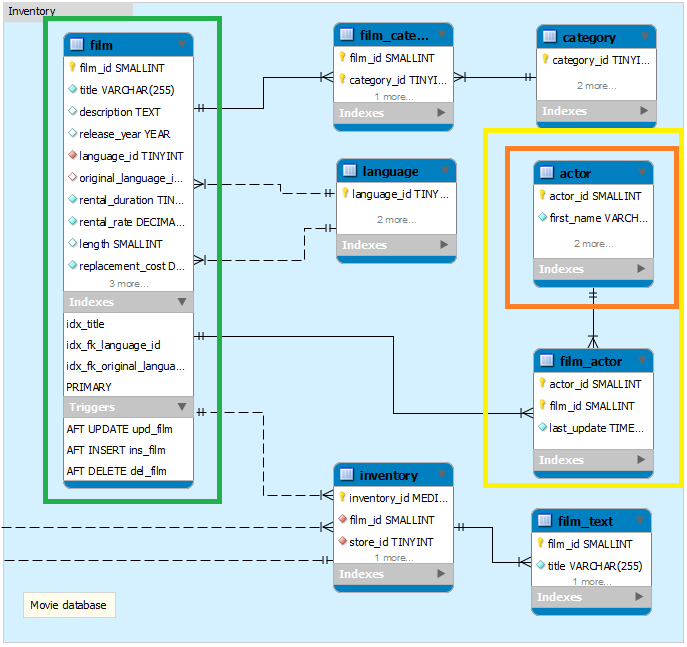
* + all the actors
  + that joined in the film\_id = 369
* Your query should return the
  + film\_id
  + title
  + actor\_id
  + actor\_name

SELECT film\_actor.film\_id, film.title, actor.actor\_id, actor.first\_name, actor.last\_name FROM film

JOIN film\_actor ON film\_actor.film\_id = film.film\_id

JOIN actor ON film\_actor.actor\_id = actor.actor\_id

WHERE film.film\_id = 369



7. What query would you run to get

* + all drama films
  + with a rental rate of 2.99
* Your query should return
  + film title
  + description
  + release year
  + rating
  + special features
  + genre

SELECT film.film\_id, title AS Film\_Title, description, release\_year, rating, special\_features, name AS Genre, rental\_rate FROM film\_category

JOIN film ON film.film\_id = film\_category.film\_id

JOIN category ON category.category\_id = film\_category.category\_id

WHERE rental\_rate = 2.99 AND name LIKE 'Drama'

8. What query would you run to get

* + all the action films
  + which are joined by SANDRA KILMER
* Your query should return
  + film title
  + description
  + release year
  + rating
  + special features
  + genre (category)
  + actor's first name
  + last name

SELECT actor.actor\_id, first\_name, last\_name,

film.film\_id, title AS Film\_Title, description, release\_year,

rating, special\_features, name AS Genre

FROM film\_category

-- Join 'category'-Table into 'film'-Table

JOIN film ON film.film\_id = film\_category.film\_id

JOIN category ON category.category\_id = film\_category.category\_id

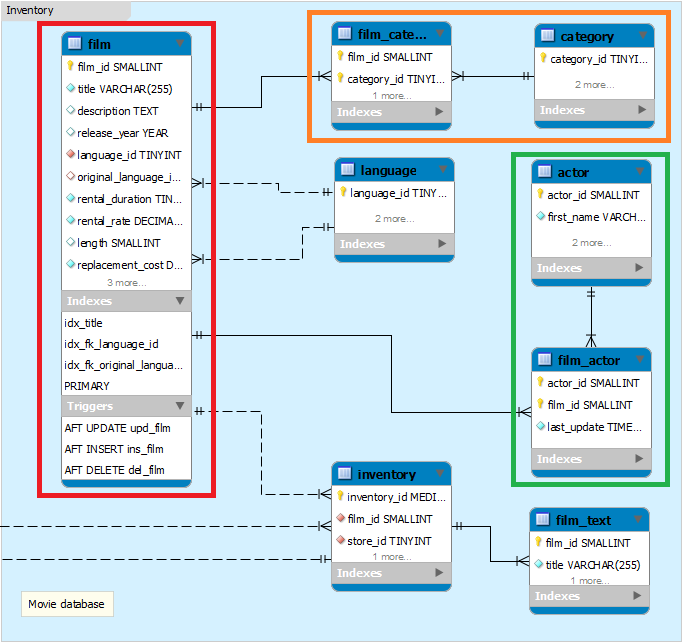
-- Join 'actor'-Table into 'film'-Table

JOIN film\_actor ON film\_actor.film\_id = film.film\_id

JOIN actor ON film\_actor.actor\_id = actor.actor\_id

-- Condition

WHERE first\_name LIKE 'Sandra' AND last\_name LIKE 'KILMER' AND name LIKE 'Action'

****

# Assignment-3: ***Friends***

To run this program, do the following:

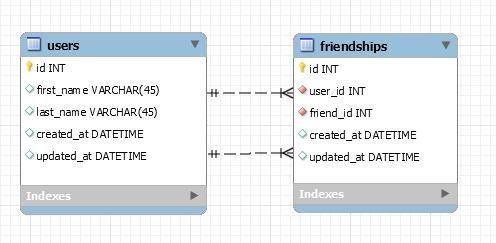
* Step 1: In new script, run the following:

CREATE DATABASE IF NOT exists friends;

* Step 2: Refresh the Schemas and double click the friends Schema
* Step 3: Load friends.sql

**Problem Statement:**

* Using the below ERD as reference, write a SQL query that returns a list of users along with their friends' names.



Your results should look like below:

|  |  |  |  |
| --- | --- | --- | --- |
| first\_name | last\_name | friend\_first\_name | friend\_last\_name |
| Amy | Giver | Eli | Byers |
| Amy | Giver | Big | Bird |
| Amy | Giver | Kermit | The Frog |
| Eli | Byers | Kermit | The Frog |
| Eli | Byers | Marky | Mark |
| Marky | Mark | Big | Bird |

Your actual query will look something similar to this:

SELECT \* FROM users

LEFT JOIN friendships ON friendships.user\_id=users.id

LEFT JOIN users as user2 ON friendships.friend\_id=users.id

Take note that we're joining the ***users*** table again but we're specifying the second ***users*** table **as** user2.  You can then reference the second ***users***by calling user2 (e.g. user2.id, user2.first\_name, etc).

1. Return all users who are friends with Kermit, make sure their names are displayed in results.
2. Return the count of all friendships

**Step 1**: Look at the raw data for the friendship relationship between the users.

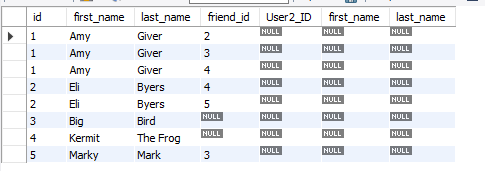
SELECT users.id, users.first\_name, users.last\_name, friend\_id, user2.id as User2\_ID, user2.first\_name, user2.last\_name,

COUNT(\*) AS Number\_of\_Friends FROM users

LEFT JOIN friendships ON friendships.user\_id=users.id

LEFT JOIN users as user2 ON friendships.friend\_id=users.id

-- WHERE friendships.friend\_id != NULL

GROUP BY users.id  
Result of displaying joining the two tables.

**Step 2**: Determine the desired result:

|  |  |
| --- | --- |
| **Person** | **Number of Friends** |
| Amy | 3 |
| Eli | 2 |
| Big | 0 |
| Kermit | 0 |
| Marky | 1 |

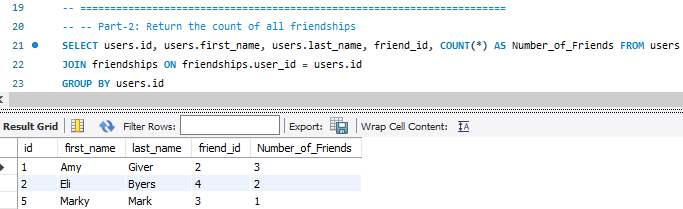
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **first\_name** | **last\_name** | **friend\_first\_name** | **friend\_last\_name** | **number\_of\_friends** |
| Amy | Giver | Eli | Byers | 3 |
| Amy | Giver | Big | Bird | 3 |
| Amy | Giver | Kermit | The Frog | 3 |
| Eli | Byers | Kermit | The Frog | 2 |
| Eli | Byers | Marky | Mark | 2 |
| Marky | Mark | Big | Bird | 1 |

**Step 3**: Attempt to produce the desired result:

SELECT users.id, users.first\_name, users.last\_name, friend\_id, COUNT(\*) AS Number\_of\_Friends FROM users

JOIN friendships ON friendships.user\_id = users.id

GROUP BY users.id



1. Find out who has the most friends and return the count of their friends.
2. Create a new user and make them friends with Eli Byers, Kermit The Frog, and Marky Mark
3. Return the friends of Eli in alphabetical order
4. Remove Marky Mark from Eli’s friends.
5. Return all friendships, displaying just the first and last name of both friends