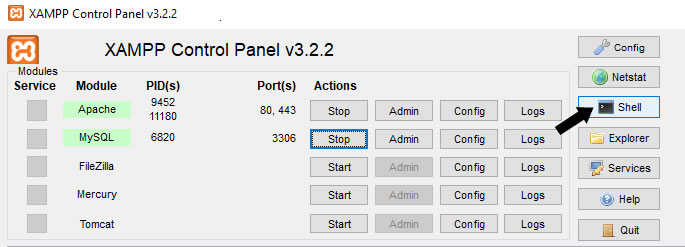
Rune Brito N.

EP:

Se accedió al programa xaamp, se activo el Mysql y a su vez se ejecutó “El terminal” para dar inicio al ejercicio.



Luego a través de los siguientes comando se importó la base de datos Shakila.sql y a su vez se los datos que esta contiene.

Dentro del terminal o consola, se introdujo la secuencia que indica que queremos conectarnos a MySQL con el usuario rootpor defecto que y sin contraseña .

**# mysql -u root –p**

**MariaDB [(none)]>**

Del archivo que nos hemos descargado de la base de datos Sakila, vemos que hay dos archivos, “sakila-schema.sql” corresponde a la estructura de la base de datos y “sakila-data.sql” que corresponden a los datos. Estos se compiaron en el disco C:/ del ordenador.

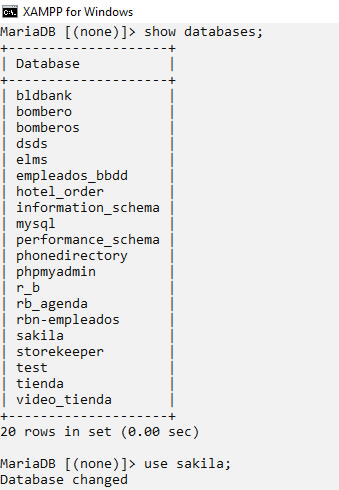
Lo primero será importar la estructura, para lo cual teniendo el archivo “sakila-schema.sql” descargado en C:/

**MariaDB [(none)]>SOURCE C:/sakila-schema.sql;**

Luego, lo mismo para los datos poniendo el nombre de fichero correspondiente en la misma ubicación de C:/

**MariaDB [(none)]>SOURCE D:/sakila-data.sql;**

Una vez importado los dos archivos, se ha creado la base de datos shakila con toda suestructura y datos en su interior

**1. Creacion de tablas y campos:**

Se creo una tabla con sus campos y contenidos en dentro de la base de datos Sakila que lleva por nombre “Artistas\_CD” y que lleva como campos para ser rellenados a futuro: nombre, apellidos, agrupación, email, teléfono y país.

CREATE TABLE IF NO EXIST `artista\_cd` ( `nombre` varchar(16) NOT NULL,

`Apellidos` varchar(30) NOT NULL,

`agrupacion` varchar(30) NOT NULL,

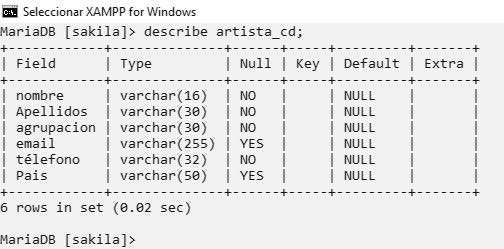
`email` varchar(255) DEFAULT NULL,

`télefono` varchar(32) NOT NULL,

`Pais` varchar(50) DEFAULT NULL )

ENGINE=InnoDB DEFAULT CHARSET=UTT-8

Luego dentro de la línea de comando se escribió la sentencia, la tabla y su contenido

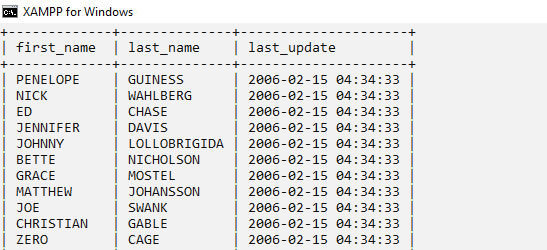


**2. CONSULTAS EN LA BASE DE DATOS SAKILA**

**2.1 Consulta Básicas**

Se realizó una consulta básica en la tabla “actor” por “nombre”, “apellidos” y “fecha”

SELECT first\_name,last\_name, last\_update FROM actor;

****

Con esta sentencia se genera una consulta que muestra los clientes que estan están en la tabla rental.

SELECT rental.rental\_id,customer.first\_name,

customer.last\_name,rental.rental\_date,rental.return\_date

FROM rental

INNER JOIN customer

ON rental.customer\_id=customer.customer\_id

Con esta sentencia se muestra las películas alquiladas con una mayor frecuencia en orden descendente.

SELECT COUNT(r.rental\_id), f.title as "Film"

FROM film f

INNER JOIN inventory i ON f.film\_id = i.film\_id

INNER JOIN rental r ON i.inventory\_id = r.inventory\_id

GROUP BY 2

ORDER BY 2 DESC;

Con esta sentencia se puede consultar los apellidos de los actores y la cantidad de actores que tienen ese apellido, pero solo para nombres compartidos por al menos dos actores.

SELECT last\_name, COUNT(\*)

FROM actor

GROUP BY 1

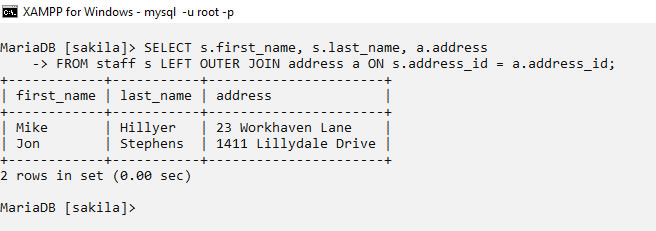
HAVING COUNT(\*) >= 2

**2.2 Consultas con left Joint**

Se utilizó Joint para la consulta de los nombres y apellidos, así como la dirección de cada miembro del personal y nombre del personal del staff.

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

FROM staff s LEFT OUTER JOIN address a ON s.address\_id = a.address\_id;

****

**2.3 CONSULTAS ELABORADAS CON INNER JOINT**

Se utilizo un par de consultas “Joint”, para ello se tomó como ejemplo de la vida real dentro de Sakila y por ende de una tienda de DVD, la cuales pueden producir una lista diaria de alquileres vencidos para que los clientes puedan ser contactados y pedirles que devuelvan sus DVD vencidos.

Para una consulta de este tipo, se buscan películas en la tabla de alquiler con una fecha de retorno NULA y donde la fecha de alquiler sea más antigua que la duración del alquiler especificada en la tabla de películas. Si es así, la película está vencida y debemos producir el nombre de la película junto con el nombre del cliente y el número de teléfono.

SELECT CONCAT (customer.last\_name,', ',customer.first\_name)AScustomer,

address.phone, film.title

FROM rental INNERJOIN customer

ON rental.customer\_id=customer.customer\_id

INNER JOIN addressONcustomer.address\_id=address.address\_id

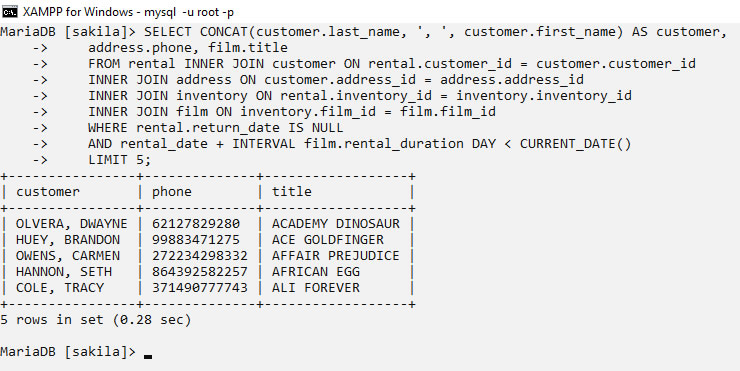
INNER JOIN inventoryONrental.inventory\_id=inventory.inventory\_id

INNER JOIN film ONinventory.film\_id=film.film\_id

WHERE rental.return\_dateI SNULL

AND rental\_date + INTERVALfilm.rental\_duration DAY < CURRENT\_DATE ()

LIMIT5;

****

**DESCENCADENADORES ( TRIGGERS)**

**Borrar**

CREATE TRIGGER `sakila`.`actor\_BEFORE\_DELETE` BEFORE DELETE ON `actor` FOR EACH ROW

BEGIN

If OLD.actor\_id between 1 and 20 then

SIGNAL SQL STATE '45000' SET MESSAGE\_TEXT = 'No puedes borrar';

else insert into actor\_log(first\_name, last\_name) values (OLD.first\_name, OLD.last\_name);

end if;

END

**Insertar una oferta de 7.50$ cuando tenga un precio de 10$**

CREATE DEFINER=`root`@`localhost` TRIGGER `sakila`.`payment\_BEFORE\_INSERT` BEFORE INSERT ON `payment` FOR EACH ROW

BEGIN

if NEW.amount>=10 then

set NEW.amount=7.50;

end if;

END

**Creación de Vistas**

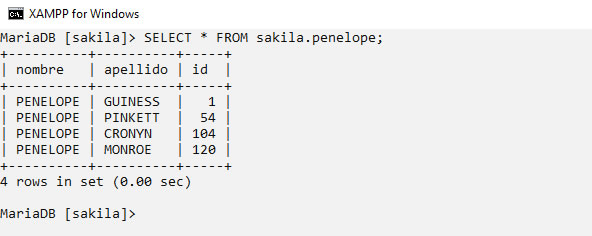
Con este código o sentencia se obtuvo una vista con los datos de los films cuyos nombres de actrices empiezan por Penélope.

CREATE VIEW penelope AS

SELECT first\_name as nombre, last\_name as apellido, actor\_id as id

FROM actor

WHERE first\_name = 'PENELOPE';



Creando una vista para ver los el top de cinco géneros que generan mayores ingresos a la tienda.

CREATE VIEW top\_5 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

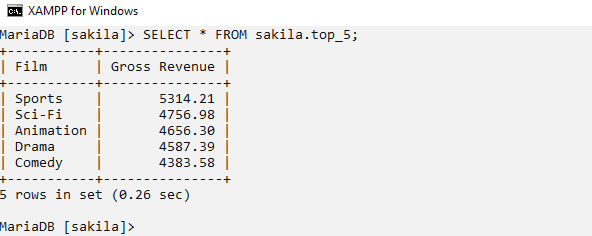
join rental as r on r.inventory\_id = i.inventory\_id

join payment as p on p.rental\_id = r.rental\_id

group by c.name

order by sum(p.amount) desc

limit 5;



Función

Procedimiento