# IFCD 0210. DESARROLLO DE APLICACIONES CON TECNOLOGÍAS WEB.

MF 0492\_3. SERVIDOR

UF 1845_3. ACCESO A DATOS EN APLICACIONES WEB DEL ENTONO SERVIDOR.
•••••

- Examen Práctico -

**Angel David Morales Expósito** 

# ACTIVIDAD 2. Realizar cinco consultas SQL de libre elección.

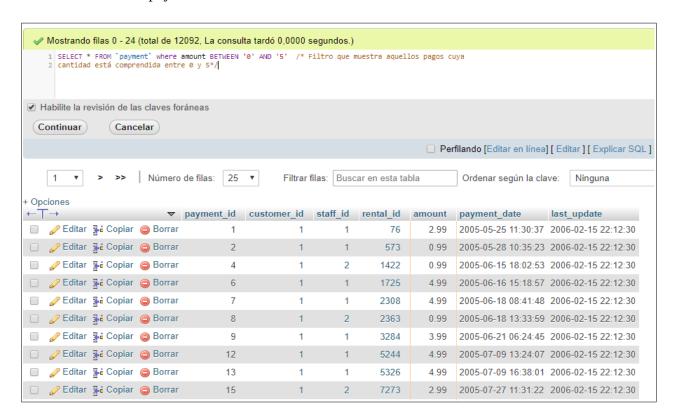
\*Consulta que muestra aquellos actores en los que su apellido comienza por la letra A.

SELECT \* FROM 'actor' where last\_name like 'A%'



\*Consulta que muestra aquellos pagos cuya cantidad está comprendida entre 0 y 5.

SELECT \* FROM 'payment' where amount BETWEEN '0' AND '5'



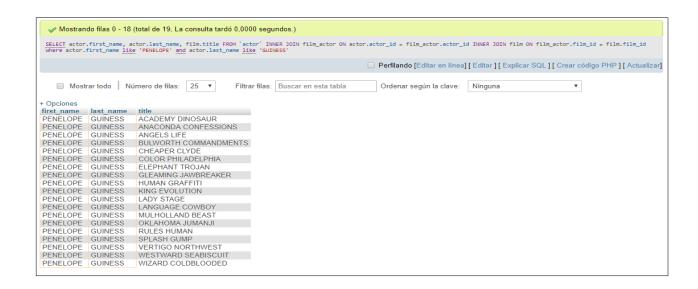
#### \*Ordena todos los actores por orden alfabético según su nombre.

#### SELECT \* FROM 'actor' ORDER BY first\_name

✓ Mostrando filas 0 - 24 (total de 200, La consulta tardó 0,0000 segundos.) [first_name: ADAM CARMEN]												
SE	LECT * FROM	`actor` ORDE	ER BY first_	_name								
	☐ Perfilando [Editar en línea] [ Editar ] [ Explicar SQL											
	1 🔻	> >>	☐ Mostr	ar todo	Número de filas:	25 ▼ Filtrar	filas: Buscar en esta ta	abla Ordenar según la clave:				
	pciones		$\forall$	actor_id	first_name 🔺 1	last_name	last_update					
	🥒 Edita	r 💤 Copiar	Borrar	71	ADAM	GRANT	2006-02-15 04:34:33					
	<i>⊘</i> Edita	r 👫 Copiar	Borrar	132	ADAM	HOPPER	2006-02-15 04:34:33					
	🥜 Edita	r 👫 Copiar	Borrar	165	AL	GARLAND	2006-02-15 04:34:33					
	🥜 Edita	r 🛂 Copiar	Borrar	173	ALAN	DREYFUSS	2006-02-15 04:34:33					
	🥜 Edita	r 🛂 Copiar	Borrar	146	ALBERT	JOHANSSON	2006-02-15 04:34:33					
	🥜 Edita	r 🛂 Copiar	Borrar	125	ALBERT	NOLTE	2006-02-15 04:34:33					
	∂ Edita	r 🏰 Copiar	Borrar	29	ALEC	WAYNE	2006-02-15 04:34:33					
	🥜 Edita	r 🛂 Copiar	Borrar	65	ANGELA	HUDSON	2006-02-15 04:34:33					
	∂ Edita	r 🏰 Copiar	Borrar	144	ANGELA	WITHERSPOON	2006-02-15 04:34:33					
	🥜 Edita	r 🛂 Copiar	Borrar	76	ANGELINA	ASTAIRE	2006-02-15 04:34:33					
	∂ Edita	r 🛂 Copiar	Borrar	49	ANNE	CRONYN	2006-02-15 04:34:33					
	<i>⊘</i> Edita	r <b>≩≟</b> Copiar	Borrar	190	AUDREY	BAILEY	2006-02-15 04:34:33					
	<i>⊘</i> Edita	r 🛂 Copiar	Borrar	34	AUDREY	OLIVIER	2006-02-15 04:34:33					
	Consola ta	r 🛂 Copiar	Borrar	196	BELA	WALKEN	2006-02-15 04:34:33					

#### \*Consulta en la que se muestran todas las películas realizadas por la actriz PENELOPE GUINESS.

SELECT actor.first\_name, actor.last\_name, film.title FROM `actor`
INNER JOIN film\_actor
ON actor.actor\_id = film\_actor.actor\_id
INNER JOIN film
ON film\_actor.film\_id = film.film\_id
where actor.first\_name like 'PENELOPE' and actor.last\_name like 'GUINESS'



\*Consulta en la que se muestran todas las rentas realizadas el día 2005-05-25, ordenadas por orden cronológico ascendente.

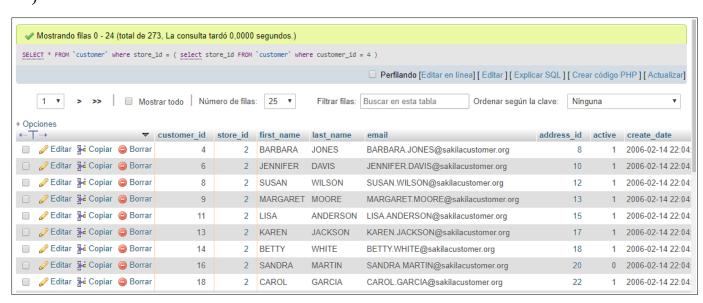
SELECT \* FROM `rental` where rental\_date BETWEEN '2005-05-25 00:00:00' AND '2005-05-25 23:59:59' ORDER by rental\_date ASC

1	/lostrando	filas 0 - 13	6 (total de 1	I37. La cons	ulta tardó 0,0000 segu	ndos.) [rental da	ate: 2005-05-25 (	00:00:40 2005-05-2	5 23:59:03	1
					'2005-05-25 00:00:00'					•
	Permanuo [cuitai en linea] [ cuitai ] [ Explicar SQL ] [ crear courgo Prin-]									
	Mostra	rtodo N	úmero de fi	las: 500	▼ Filtrar filas: B	uscar en esta ta	bla Ord	lenar según la clave:	Ninguna	▼
pci	ones									
T	<b>→</b>		~	rental_id	rental_date 🔺 1	inventory_id	customer_id	return_date	staff_id	last_update
)		<b>3</b> -i Copiar	Borrar	9	2005-05-25 00:00:40	2580	126	2005-05-28 00:22:40	1	2006-02-15 21:30:53
)	<i>⊘</i> Editar	<b>≩</b> di Copiar	Borrar	10	2005-05-25 00:02:21	1824	399	2005-05-31 22:44:21	2	2006-02-15 21:30:53
)		<b>3</b> -i Copiar	Borrar	11	2005-05-25 00:09:02	4443	142	2005-06-02 20:56:02	2	2006-02-15 21:30:53
	Ø Editar	<b>≩</b> i Copiar	Borrar	12	2005-05-25 00:19:27	1584	261	2005-05-30 05:44:27	2	2006-02-15 21:30:53
	<i></i> €ditar	<b>3</b> -i Copiar	Borrar	13	2005-05-25 00:22:55	2294	334	2005-05-30 04:28:55	1	2006-02-15 21:30:53
		<b>≩</b> i Copiar	Borrar	14	2005-05-25 00:31:15	2701	446	2005-05-26 02:56:15	1	2006-02-15 21:30:53
Ī	<i></i> €ditar	<b>3</b> -i Copiar	Borrar	15	2005-05-25 00:39:22	3049	319	2005-06-03 03:30:22	1	2006-02-15 21:30:53
		<b>3</b> di Copiar	Borrar	16	2005-05-25 00:43:11	389	316	2005-05-26 04:42:11	2	2006-02-15 21:30:53
		<b>3</b> -i Copiar	Borrar	17	2005-05-25 01:06:36	830	575	2005-05-27 00:43:36	1	2006-02-15 21:30:53
	<i></i> € Editar	<b>3</b> € Copiar	Borrar	18	2005-05-25 01:10:47	3376	19	2005-05-31 06:35:47	2	2006-02-15 21:30:53
	Editar	<b>3</b> -i Copiar	Borrar	19	2005-05-25 01:17:24	1941	456	2005-05-31 06:00:24	1	2006-02-15 21:30:53
	<i></i> € Editar	<b>≩</b> di Copiar	Borrar	20	2005-05-25 01:48:41	3517	185	2005-05-27 02:20:41	2	2006-02-15 21:30:53
Ī		<b>≩</b> € Copiar	Borrar	21	2005-05-25 01:59:46	146	388	2005-05-26 01:01:46	2	2006-02-15 21:30:53
)		<b>≩</b> di Copiar	Borrar	22	2005-05-25 02:19:23	727	509	2005-05-26 04:52:23	2	2006-02-15 21:30:53

<u>ACTIVIDAD 3.</u> Realizar una subconsulta de libre eleccción en la que se genere al menos una vista SQL

\* Se muestran todos los registros que tengan un store\_id igual que el que tiene el customer\_id numero 4.

```
SELECT * FROM `customer`
where store_id =
(
  select store_id FROM `customer`
  where customer_id = 4
)
```



# ACTIVIDAD 4 Implementar al menos dos triggers

\* Trigger que guarda la información de una dirección en otra tabla cuando se actualice la información de dicha dirección.

```
CREATE TRIGGER 'update_address' BEFORE UPDATE ON 'address'
FOR EACH ROW BEGIN
      IF (OLD.address_id != new.address_id ) OR (OLD.address != new.address ) OR (OLD.district !
= new.district ) OR (OLD.city_id != new.city_id ) OR (OLD.postal_code != new.postal_code ) OR
(OLD.phone != new.phone )
  THEN
     INSERT INTO old_address (
       old_address_id,
       address.
       district,
       city_id,
       postal_code,
       phone,
       change_date)
     VALUES (
       OLD.address_id,
       OLD.address,
       OLD.district,
       OLD.city_id,
       OLD.postal_code,
       OLD.phone,
       now()
       );
END IF;
END
```

\* Trigger que cuando se borra un registro de rental, comprueba que la película se ha entregado a tiempo, y de no ser así, se incluyen los datos del cliente en otra tabla llamada deuda.

```
CREATE TRIGGER 'add deuda' BEFORE DELETE ON 'rental'
FOR EACH ROW BEGIN
      IF ( old.return_date < NOW() )</pre>
      THEN
    INSERT INTO deuda (
      rental_id,
      return date,
      customer_id,
      first name,
      last_name)
     VALUES (
       old.rental id,
       old.return date,
       old.customer_id,
       (SELECT first_name FROM `customer`
       where customer_id = old.customer_id),
       (SELECT last name FROM 'customer'
       where customer_id = old.customer_id)
       );
END IF;
END
```

## ACTIVIDAD 5 IMPLEMENTAR AL MENOS UNA FUNCIÓN

\*Función en la que se introduce el id de un usuario y devuelve la cantidad de dinero que ha gastado en pagos.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `customer_payment_amount`(`usuario` SMALLINT) RETURNS decimal(10,0)
    NO SQL
BEGIN
declare total decimal;

SET total = (SELECT SUM(amount)
FROM payment
WHERE customer_id = usuario);

return total;
END$$
DELIMITER;
```

## ACTIVIDAD 6 IMPLEMENTAR AL MENOS UN PROCEDIMIENTO

\*Procedimiento que muestra todas las películas realizadas por la actriz Penélope Guinness ( al igual que la consulta número 4).

DELIMITER \$\$
CREATE DEFINER=`root`@`localhost` PROCEDURE `pe\_pe`()
NO SQL
SELECT actor.first\_name, actor.last\_name, film.title FROM `actor`
INNER JOIN film\_actor
ON actor.actor\_id = film\_actor.actor\_id
INNER JOIN film
ON film\_actor.film\_id = film.film\_id
where actor.first\_name like 'PENELOPE' and actor.last\_name like 'GUINESS'\$\$
DELIMITER;