

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%'

✓ Mostrando filas 0 - 6 (total de 7, La consulta tardó 0,0000 segundos.)

1 SELECT * FROM `actor` where last_name like 'A%' /* Filtro que muestra aquellos actores en los que su apellido comienza por la letra A */

☒ Habilite la revisión de las claves foráneas

Continuar Cancelar

☐ Perfilando [Editar en línea] [Editar] [Explicar SQL]

☐ Mostrar todo | Número de filas: 25 Filtar filas: Buscar en esta tabla Ordenar según la clave: Ninguna

+ Opciones

		actor_id	first_name	last_name	last_update
<input type="checkbox"/>	Editar Copiar Borrar	58	CHRISTIAN	AKROYD	2006-02-15 04:34:33
<input type="checkbox"/>	Editar Copiar Borrar	92	KIRSTEN	AKROYD	2006-02-15 04:34:33
<input type="checkbox"/>	Editar Copiar Borrar	182	DEBBIE	AKROYD	2006-02-15 04:34:33
<input type="checkbox"/>	Editar Copiar Borrar	118	CUBA	ALLEN	2006-02-15 04:34:33
<input type="checkbox"/>	Editar Copiar Borrar	145	KIM	ALLEN	2006-02-15 04:34:33
<input type="checkbox"/>	Editar Copiar Borrar	194	MERYL	ALLEN	2006-02-15 04:34:33
<input type="checkbox"/>	Editar Copiar Borrar	76	ANGELINA	ASTAIRE	2006-02-15 04:34:33

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

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

✓ Mostrando filas 0 - 24 (total de 12092, La consulta tardó 0,0000 segundos.)

1 SELECT * FROM `payment` where amount BETWEEN '0' AND '5' /* Filtro que muestra aquellos pagos cuya
2 cantidad está comprendida entre 0 y 5 */

☒ Habilite la revisión de las claves foráneas

Continuar Cancelar

☐ Perfilando [Editar en línea] [Editar] [Explicar SQL]

1 > >> | Número de filas: 25 Filtar filas: Buscar en esta tabla Ordenar según la clave: Ninguna

+ Opciones

		payment_id	customer_id	staff_id	rental_id	amount	payment_date	last_update
<input type="checkbox"/>	Editar Copiar Borrar	1	1	1	76	2.99	2005-05-25 11:30:37	2006-02-15 22:12:30
<input type="checkbox"/>	Editar Copiar Borrar	2	1	1	573	0.99	2005-05-28 10:35:23	2006-02-15 22:12:30
<input type="checkbox"/>	Editar Copiar Borrar	4	1	2	1422	0.99	2005-06-15 18:02:53	2006-02-15 22:12:30
<input type="checkbox"/>	Editar Copiar Borrar	6	1	1	1725	4.99	2005-06-16 15:18:57	2006-02-15 22:12:30
<input type="checkbox"/>	Editar Copiar Borrar	7	1	1	2308	4.99	2005-06-18 08:41:48	2006-02-15 22:12:30
<input type="checkbox"/>	Editar Copiar Borrar	8	1	2	2363	0.99	2005-06-18 13:33:59	2006-02-15 22:12:30
<input type="checkbox"/>	Editar Copiar Borrar	9	1	1	3284	3.99	2005-06-21 06:24:45	2006-02-15 22:12:30
<input type="checkbox"/>	Editar Copiar Borrar	12	1	1	5244	4.99	2005-07-09 13:24:07	2006-02-15 22:12:30
<input type="checkbox"/>	Editar Copiar Borrar	13	1	1	5326	4.99	2005-07-09 16:38:01	2006-02-15 22:12:30
<input type="checkbox"/>	Editar Copiar Borrar	15	1	2	7273	2.99	2005-07-27 11:31:22	2006-02-15 22:12:30

***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...]

```
SELECT * FROM `actor` ORDER BY first_name
```

☐ Perfilando [Editar en línea] [Editar] [Explicar SQL]

1 > >> | ☐ Mostrar todo | Número de filas: 25 | Filtrar filas: Ordenar según la clave:

+ Opciones

	actor_id	first_name	last_name	last_update
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	71	ADAM	GRANT	2006-02-15 04:34:33
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	132	ADAM	HOPPER	2006-02-15 04:34:33
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	165	AL	GARLAND	2006-02-15 04:34:33
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	173	ALAN	DREYFUSS	2006-02-15 04:34:33
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	146	ALBERT	JOHANSSON	2006-02-15 04:34:33
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	125	ALBERT	NOLTE	2006-02-15 04:34:33
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	29	ALEC	WAYNE	2006-02-15 04:34:33
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	65	ANGELA	HUDSON	2006-02-15 04:34:33
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	144	ANGELA	WITHERSPOON	2006-02-15 04:34:33
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	76	ANGELINA	ASTAIRE	2006-02-15 04:34:33
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	49	ANNE	CRONYN	2006-02-15 04:34:33
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	190	AUDREY	BAILEY	2006-02-15 04:34:33
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	34	AUDREY	OLIVIER	2006-02-15 04:34:33
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	196	BELA	WALKEN	2006-02-15 04:34:33

Consola

***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'
```

Mostrando filas 0 - 18 (total de 19, La consulta tardó 0,0000 segundos.)

```
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'
```

☐ Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

☐ Mostrar todo | Número de filas: 25 | Filtrar filas: Ordenar según la clave: Ninguna

+ Opciones

first_name	last_name	title
PENELOPE	GUINESS	ACADEMY DINOSAUR
PENELOPE	GUINESS	ANACONDA CONFESSIONS
PENELOPE	GUINESS	ANGELS LIFE
PENELOPE	GUINESS	BULWORTH COMMANDMENTS
PENELOPE	GUINESS	CHEAPER CLYDE
PENELOPE	GUINESS	COLOR PHILADELPHIA
PENELOPE	GUINESS	ELEPHANT TROJAN
PENELOPE	GUINESS	GLEAMING JAWBREAKER
PENELOPE	GUINESS	HUMAN GRAFFITI
PENELOPE	GUINESS	KING EVOLUTION
PENELOPE	GUINESS	LADY STAGE
PENELOPE	GUINESS	LANGUAGE COWBOY
PENELOPE	GUINESS	MULHOLLAND BEAST
PENELOPE	GUINESS	OKLAHOMA JUMANJI
PENELOPE	GUINESS	RULES HUMAN
PENELOPE	GUINESS	SPLASH GUMP
PENELOPE	GUINESS	VERTIGO NORTHWEST
PENELOPE	GUINESS	WESTWARD SEABISCUIT
PENELOPE	GUINESS	WIZARD COLDBLOODED

***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

Mostrando filas 0 - 136 (total de 137, La consulta tardó 0,0000 segundos.) [rental_date: 2005-05-25 00:00:40... - 2005-05-25 23:59:03...]

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

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo | Número de filas: 500 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: Ninguna

	rental_id	rental_date	inventory_id	customer_id	return_date	staff_id	last_update
Editar Copiar Borrar	9	2005-05-25 00:00:40	2580	126	2005-05-28 00:22:40	1	2006-02-15 21:30:53
Editar Copiar Borrar	10	2005-05-25 00:02:21	1824	399	2005-05-31 22:44:21	2	2006-02-15 21:30:53
Editar Copiar Borrar	11	2005-05-25 00:09:02	4443	142	2005-06-02 20:56:02	2	2006-02-15 21:30:53
Editar Copiar Borrar	12	2005-05-25 00:19:27	1584	261	2005-05-30 05:44:27	2	2006-02-15 21:30:53
Editar Copiar Borrar	13	2005-05-25 00:22:55	2294	334	2005-05-30 04:28:55	1	2006-02-15 21:30:53
Editar Copiar Borrar	14	2005-05-25 00:31:15	2701	446	2005-05-26 02:56:15	1	2006-02-15 21:30:53
Editar Copiar Borrar	15	2005-05-25 00:39:22	3049	319	2005-06-03 03:30:22	1	2006-02-15 21:30:53
Editar Copiar Borrar	16	2005-05-25 00:43:11	389	316	2005-05-26 04:42:11	2	2006-02-15 21:30:53
Editar Copiar Borrar	17	2005-05-25 01:06:36	830	575	2005-05-27 00:43:36	1	2006-02-15 21:30:53
Editar 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 Copiar Borrar	19	2005-05-25 01:17:24	1941	456	2005-05-31 06:00:24	1	2006-02-15 21:30:53
Editar Copiar Borrar	20	2005-05-25 01:48:41	3517	185	2005-05-27 02:20:41	2	2006-02-15 21:30:53
Editar Copiar Borrar	21	2005-05-25 01:59:46	146	388	2005-05-26 01:01:46	2	2006-02-15 21:30:53
Editar Copiar Borrar	22	2005-05-25 02:19:23	727	509	2005-05-26 04:52:23	2	2006-02-15 21:30:53

ACTIVIDAD 3.

Realizar una subconsulta de libre elección 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
)

Mostrando filas 0 - 24 (total de 273, La consulta tardó 0,0000 segundos.)

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

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

1 > >> | Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: Ninguna

	customer_id	store_id	first_name	last_name	email	address_id	active	create_date
Editar Copiar Borrar	4	2	BARBARA	JONES	BARBARA.JONES@sakilacustomer.org	8	1	2006-02-14 22:04:
Editar Copiar Borrar	6	2	JENNIFER	DAVIS	JENNIFER.DAVIS@sakilacustomer.org	10	1	2006-02-14 22:04:
Editar Copiar Borrar	8	2	SUSAN	WILSON	SUSAN.WILSON@sakilacustomer.org	12	1	2006-02-14 22:04:
Editar Copiar Borrar	9	2	MARGARET	MOORE	MARGARET.MOORE@sakilacustomer.org	13	1	2006-02-14 22:04:
Editar Copiar Borrar	11	2	LISA	ANDERSON	LISA.ANDERSON@sakilacustomer.org	15	1	2006-02-14 22:04:
Editar Copiar Borrar	13	2	KAREN	JACKSON	KAREN.JACKSON@sakilacustomer.org	17	1	2006-02-14 22:04:
Editar Copiar Borrar	14	2	BETTY	WHITE	BETTY.WHITE@sakilacustomer.org	18	1	2006-02-14 22:04:
Editar Copiar Borrar	16	2	SANDRA	MARTIN	SANDRA.MARTIN@sakilacustomer.org	20	0	2006-02-14 22:04:
Editar Copiar Borrar	18	2	CAROL	GARCIA	CAROL.GARCIA@sakilacustomer.org	22	1	2006-02-14 22:04:

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() )  
    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 $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `pe_pe`()
    NO SQL
SELECT actor.first_name, actor.last_name, film.title FROM `actor`
INNER JOIN film_actor
ON actor.actor_id = film_actor.actor_id
INNER JOIN film
ON film_actor.film_id = film.film_id
where actor.first_name like 'PENELOPE' and actor.last_name like 'GUINNESS'$$
DELIMITER ;
```