|  |  |  |
| --- | --- | --- |
| Universidad Don Bosco - Wikipedia, la enciclopedia libre | CARNET | NOMBRES |
| AA191128 | Marco Antonio Arias Artiga |
| PA150267 | Anthony William Pérez Abrego |
| MA192074 | Gabriel Steven Molina Arias |
| ENLACE GITHUB | |
| https://github.com/maaa010620/Proyecto\_Catedra\_DSS.git | |

DICCIONARIO DE DATOS

BD: centro\_peliculas

TABLAS:

alquileres (tabla) (table que define los registros de alquileres efectuados)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |
| codPelicula | int(11) | No |  | Código de cada pelicula existente |
| codUsuario | int(11) | No |  | Codigo de cada usuario existente |
| precioAlquiler | decimal(10,2) | Yes | *NULL* | Precio por alquiler de cada pelicula |
| fechaInicio | date | No |  | Fecha de inicio cuando se alquile la pelicula |
| fechaFin | date | No |  | Fecha de finalizacion del alquiler de la pelicula |
| fechaRetorno | date | Yes | *NULL* | Fecha en que se devolvio la pelicula |
| multa | decimal(10,2) | Yes | 0.00 | Precio monetario de la multa aplicada en caso de retraso |
| unidades | int(11) | No |  | Unidades disponibles |

**Indexes (constraints) (restricciones aplicables a la tabla)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Keyname** | **Type** | **Unique** | **Packed** | **Column** | **Cardinality** | **Collation** | **Null** | **Comment** |
| codPelicula | BTREE | No | No | codPelicula | 0 | A | No | Llave foranea relacionada con la tabla de declaracion de la pelicula |
| codUsuario | BTREE | No | No | codUsuario | 0 | A | No | Llave foranea relacionada con la tabla de declaracion del usuario |

cambios\_precio\_alquiler (Query)

Table comments: *VIEW*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |
| título | varchar(80) | No |  | Vista creada con el fin de consultar por medio del titulo, los cambios en los precios de alquiler de cada pelicula, junto con la fecha en que se realize el cambio. |
| Precio Alquiler Viejo | decimal(10,2) | No |  |
| Precio Alquiler Nuevo | decimal(10,2) | No |  |
| Fecha y Hora de Actualización | datetime | No |  |

cambios\_precio\_compra (Query)

Table comments: *VIEW*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |
| título | varchar(80) | No |  | Vista creada con el fin de consultar por medio del titulo, los cambios en los precios de compra de cada pelicula, junto con la fecha en que se realize el cambio. |
| Precio Compra Viejo | decimal(10,2) | No |  |
| Precio Compra Nuevo | decimal(10,2) | No |  |
| Fecha y Hora de Actualización | datetime | No |  |

compras (tabla) (table que define los registros de compras efectuadas)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |
| codPelicula | int(11) | No |  | Código de cada pelicula existente |
| codUsuario | int(11) | No |  | Código de cada usuario existente |
| precioCompra | decimal(10,2) | Yes | *NULL* | Precio de compra de cada pelicula |
| fechaCompra | date | No |  | Fecha en que se realiza la compra de cada pelicula |
| unidades | int(11) | No |  | Cantidad de unidades que esta comprando |

**Indexes (constraints) (restricciones aplicables a la tabla)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Keyname** | **Type** | **Unique** | **Packed** | **Column** | **Cardinality** | **Collation** | **Null** | **Comment** |
| codPelicula | BTREE | No | No | codPelicula | 0 | A | No | Llave foranea relacionada con la tabla de declaracion de la pelicula |
| codUsuario | BTREE | No | No | codUsuario | 0 | A | No | Llave foranea relacionada con la tabla de declaracion del usuario |

log\_precios\_alquiler(tabla) (table que define los registros de los precios de alquiler efectuados)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |
| codPelicula | int(11) | No |  | Código de cada pelicula existente |
| precioAlquilerViejo | decimal(10,2) | No |  | Precio asignado antiguamente al alquiler de dicha pelicula |
| precioAlquilerNuevo | decimal(10,2) | No |  | Nuevo precio asignado al alquiler de dicha pelicula |
| fechaCambio | datetime | No |  | Fecha en que se efectuaron los cambios |

**Indexes (constraints) (restricciones aplicables a la tabla)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Keyname** | **Type** | **Unique** | **Packed** | **Column** | **Cardinality** | **Collation** | **Null** | **Comment** |
| codPelicula | BTREE | No | No | codPelicula | 1 | A | No | Llave foranea relacionada con la tabla de declaracion de la pelicula |

log\_precios\_compra(tabla) (table que define los registros de precios de compra efectuados)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |
| codPelicula | int(11) | No |  | Código de cada pelicula existente |
| precioCompraViejo | decimal(10,2) | No |  | Precio asignado antiguamente de compra de dicha pelicula |
| precioCompraNuevo | decimal(10,2) | No |  | Nuevo precio asignado a la compra de dicha pelicula |
| fechaCambio | datetime | No |  | Fecha en que se efectuaron los cambios |

**Indexes (constraints) (restricciones aplicables a la tabla)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Keyname** | **Type** | **Unique** | **Packed** | **Column** | **Cardinality** | **Collation** | **Null** | **Comment** |
| codPelicula | BTREE | No | No | codPelicula | 1 | A | No | Llave foranea relacionada con la tabla de declaracion de la pelicula |

peliculas(tabla) (table que define los registros de las peliculas registradas)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |
| codPelicula *(Primary)* | int(11) | No |  | Definicion unica del codigo asignado a una pelicula en especifico |
| título | varchar(80) | No |  | Titulo asignado a una pelicula en especifico |
| descripción | text | No |  | Descripcion asignada a una pelicula en especifico |
| imagenURL | varchar(80) | No |  | url de imagen de muestra asignado a una pelicula en especifico |
| precioAlquiler | decimal(10,2) | No |  | Primer precio de alquiler asignado a una pelicula en especifico |
| precioCompra | decimal(10,2) | No |  | Primer precio de compra asignado a una pelicula en especifico |
| stock | int(11) | No |  | Candtidad de unidades disponibles por peliculas |
| disponibilidad | int(11) | No |  | Disponibilidad actual para poder rentar la pelicula |
| likes | int(11) | No |  | Votacion sobre la popularidad de cada pelicula |

**Indexes (constraints) (restricciones aplicables a la tabla)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Keyname** | **Type** | **Unique** | **Packed** | **Column** | **Cardinality** | **Collation** | **Null** | **Comment** |
| PRIMARY | BTREE | Yes | No | codPelicula | 2 | A | No | Defincion de llave primaria sobre el campo donde se asigna el codigo de la pelicula |

peliculas\_alquiladas\_actualmente (Query)

Table comments: *VIEW*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |
| Título | varchar(80) | No |  | Vista creada con el fin de consultar por medio del titulo o el nombre de un usuario, las peliculas alquiladas actualmente, las fechas de inicio y finalizacion para el aquiler, la cantidad de peliculas que alquilo y el precio total. |
| Nombre | varchar(80) | No |  |
| Apellidos | varchar(80) | No |  |
| Fecha Inicio | date | No |  |
| Fecha Fin | date | No |  |
| Unidades | int(11) | No |  |
| Precio | decimal(10,2) | Yes | *NULL* |
| Total | decimal(20,2) | Yes | *NULL* |

peliculas\_alquiladas\_previamente (Query)

Table comments: *VIEW*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |
| Título | varchar(80) | No |  | Vista creada con el fin de consultar por medio del titulo o el nombre de un usuario, las peliculas alquiladas previamente, |
| Nombre | varchar(80) | No |  |
| Apellidos | varchar(80) | No |  |
| Fecha Inicio | date | No |  |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fecha Fin | date | No |  | las fechas de inicio y finalizacion del aquiler, la fecha de retorno, la cantidad de peliculas que alquilo y el precio total que en caso de que hubo el retraso la multa que pago. |
| Fecha Retorno | date | Yes | *NULL* |
| Unidades | int(11) | No |  |
| Precio | decimal(10,2) | Yes | *NULL* |
| Multa | decimal(10,2) | Yes | 0.00 |
| Total | decimal(21,2) | Yes | *NULL* |

peliculas\_disponibles (Query)

Table comments: *VIEW*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |
| Código | int(11) | No | 0 | Vista creada con la finalidad de poder consultar las peliculas que esten disponibles para poder ser alquiladas o compradas por los usuarios, pudiendo mostrar de las peliculas disponibles los datos como: su titulo, descripcion, imagen, precio de alquiuler o compra, unidades disponibles y la puntuacion de dicha pelicula. |
| Título | varchar(80) | No |  |
| Descripción | text | No |  |
| Carátula | varchar(80) | No |  |
| Precio Alquiler | decimal(10,2) | No |  |
| PrecioCompra | decimal(10,2) | No |  |
| Unidades Totales | int(11) | No |  |
| Unidades Disponibles | int(11) | No |  |
| Likes | int(11) | No |  |

peliculas\_no\_disponibles (Query)

Table comments: *VIEW*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |
| Código | int(11) | No | 0 | Vista creada con la finalidad de poder consultar las peliculas que no esten disponibles para poder ser alquiladas o compradas por los usuarios, pudiendo mostrar de las peliculas no disponibles los datos como: su titulo, descripcion, imagen, precio de alquiuler o compra, unidades disponibles y la puntuacion de dicha pelicula. |
| Título | varchar(80) | No |  |
| Descripción | text | No |  |
| Carátula | varchar(80) | No |  |
| Precio Alquiler | decimal(10,2) | No |  |
| PrecioCompra | decimal(10,2) | No |  |
| Unidades Totales | int(11) | No |  |
| Unidades Disponibles | int(11) | No |  |
| Likes | int(11) | No |  |

peliculas\_retornadas\_puntualmente (Query)

Table comments: *VIEW*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Título | varchar(80) | No |  | Vista creada con la finalidad de poder consultar las peliculas que fueron alquiladas y retornadas puntualmente por los usuarios, con detalles como: informacion del usuario que la retorna, las fechas desde que alquilo la pelicula hasta su devolucion y su fecha de retorno, las unidades que alquilo, precios totals sin multas aplicables. |
| Nombre | varchar(80) | No |  |
| Apellidos | varchar(80) | No |  |
| Fecha Inicio | date | No |  |
| Fecha Fin | date | No |  |
| Fecha Retorno | date | Yes | *NULL* |
| Unidades | int(11) | No |  |
| Precio | decimal(10,2) | Yes | *NULL* |
| Multa | decimal(10,2) | Yes | 0.00 |
| Total | decimal(21,2) | Yes | *NULL* |

peliculas\_retornadas\_tarde (Query)

Table comments: *VIEW*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |
| Título | varchar(80) | No |  | Vista creada con la finalidad de poder consultar las peliculas que fueron alquiladas y no fueron retornadas puntualmente por los usuarios, con detalles como: informacion del usuario que la retorna, las fechas desde que alquilo la pelicula hasta su devolucion y su fecha de retorno vencida, las unidades que alquilo, precios totals y las multas aplicables por ser retornada tarde. |
| Nombre | varchar(80) | No |  |
| Apellidos | varchar(80) | No |  |
| Fecha Inicio | date | No |  |
| Fecha Fin | date | No |  |
| Fecha Retorno | date | Yes | *NULL* |
| Unidades | int(11) | No |  |
| Precio | decimal(10,2) | Yes | *NULL* |
| Multa | decimal(10,2) | Yes | 0.00 |
| Total | decimal(21,2) | Yes | *NULL* |

peliculas\_todas (Query)

Table comments: *VIEW*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |
| Código | int(11) | No | 0 | Vista creada con la finalidad de poder consultar todas las peliculas actuales que posee a dispocision la empresa, con detalles como: su titulo, descripcion, imagen, precio de alquiuler o compra, unidades disponibles y la puntuacion de dicha pelicula. |
| Título | varchar(80) | No |  |
| Descripción | text | No |  |
| Carátula | varchar(80) | No |  |
| Precio Alquiler | decimal(10,2) | No |  |
| PrecioCompra | decimal(10,2) | No |  |
| Unidades Totales | int(11) | No |  |
| Unidades Disponibles | int(11) | No |  |
| Likes | int(11) | No |  |

peliculas\_vendidas (Query)

Table comments: *VIEW*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |
| Título | varchar(80) | No |  | Vista creada con la finalidad de poder consultar todas las peliculas actuales que ya no posee a dispocision la empresa, por haberlas vendido con detalles como: informacion del usuario al que se les vendios, su titulo, fecha de venta, precio de compra, unidades compradas y precios totales. |
| Nombre | varchar(80) | No |  |
| Apellidos | varchar(80) | No |  |
| Fecha | date | No |  |
| Unidades | int(11) | No |  |
| Precio | decimal(10,2) | Yes | *NULL* |
| Total | decimal(20,2) | Yes | *NULL* |

usuarios(tabla) (table que define los registros de usuarios registrados)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |
| codUsuario *(Primary)* | int(11) | No |  | Codigo unico de identificacion de cada usuario |
| nombre | varchar(80) | No |  | Nombre de cada usuario |
| apellido | varchar(80) | No |  | Apelllido del usuario |
| fechaNac | date | No |  | Fecha de Nacimiento |
| dirección | varchar(120) | No |  | Direccion actual del usuario registrado |
| teléfono | varchar(20) | No |  | Telefono de contacto con el usuario |
| adminStatus | tinyint(1) | No |  | Si el usuario es administrador o no |

**Indexes (constraints) (restricciones aplicables a la tabla)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Keyname** | **Type** | **Unique** | **Packed** | **Column** | **Cardinality** | **Collation** | **Null** | **Comment** |
| PRIMARY | BTREE | Yes | No | codUsuario | 2 | A | No | Defincion de llave primaria sobre el campo donde se asigna el codigo del usuario |

usuarios\_no\_administradores (Query)

Table comments: *VIEW*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |
| Código | int(11) | No | 0 | Vista creada con el fin de consultar todos aquellos usuarios que simplemente son usuarios con detalles como: nombre, apellido, fecha de Nacimiento, contacto y ubicacion. |
| Nombre | varchar(80) | No |  |
| Apellido | varchar(80) | No |  |
| Fecha de Nacimiento | date | No |  |
| Teléfono | varchar(20) | No |  |
| Dirección | varchar(120) | No |  |

usuarios\_administradores (Query)

Table comments: *VIEW*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Comments** |
| Código | int(11) | No | 0 | Vista creada con el fin de consultar todos aquellos usuarios que son administradores con detalles como: nombre, apellido, fecha de Nacimiento, contacto y ubicacion. |
| Nombre | varchar(80) | No |  |
| Apellido | varchar(80) | No |  |
| Fecha de Nacimiento | date | No |  |
| Teléfono | varchar(20) | No |  |
| Dirección | varchar(120) | No |  |

**CONSULTAS Y PRUEBAS DE MANTENIMIENTO DE ACUERDO A REQUERIMIENTOS:**

**INGRESO DE PELICULAS:**

INSERT INTO `peliculas` (`codPelicula`, `título`, `descripción`, `imagenURL`, `precioAlquiler`, `precioCompra`, `stock`, `disponibilidad`, `likes`) VALUES

(200201, 'Resident Evil', 'En un centro clandestino de investigación genética, se produce un virus mortal. Para contener la fuga, se cierra toda la instalación y se cree que mueren todos los empleados, pero en realidad se convierten en zombis.', 'caratula.img', '2.99', '5.50', 4, 0, 100),

(200701, 'Soy leyenda', 'Robert Neville, un brillante científico, es el único sobreviviente de una plaga creada por el hombre que transforma a los humanos en mutantes sedientos de sangre. Él vaga solitario por Nueva York, buscando a posibles sobrevivientes, y trabaja para hallar una cura para la plaga usando su propia sangre inmune. Neville sabe que las posibilidades son mínimas, y todos esperan que cometa un error para que él caiga en sus manos.', 'caratula.jpg', '2.99', '7.50', 2, 5, 230),

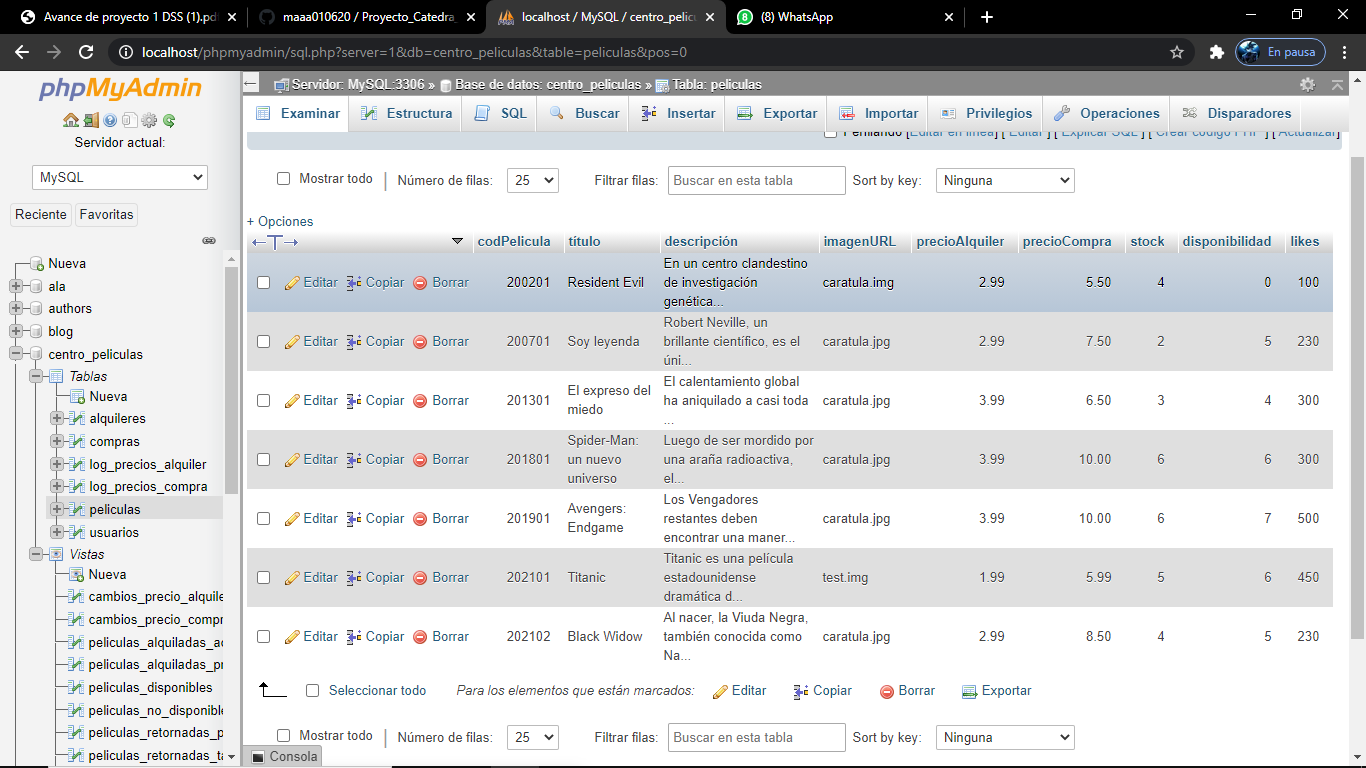
(201301, 'El expreso del miedo', 'El calentamiento global ha aniquilado a casi toda la humanidad. Los supervivientes viajan en un tren que atraviesa un mundo de hielo y nieve, un tren en el que hay dos clases sociales claramente diferenciadas, pero el descontento lleva a la revuelta.', 'caratula.jpg', '3.99', '6.50', 3, 4, 300),

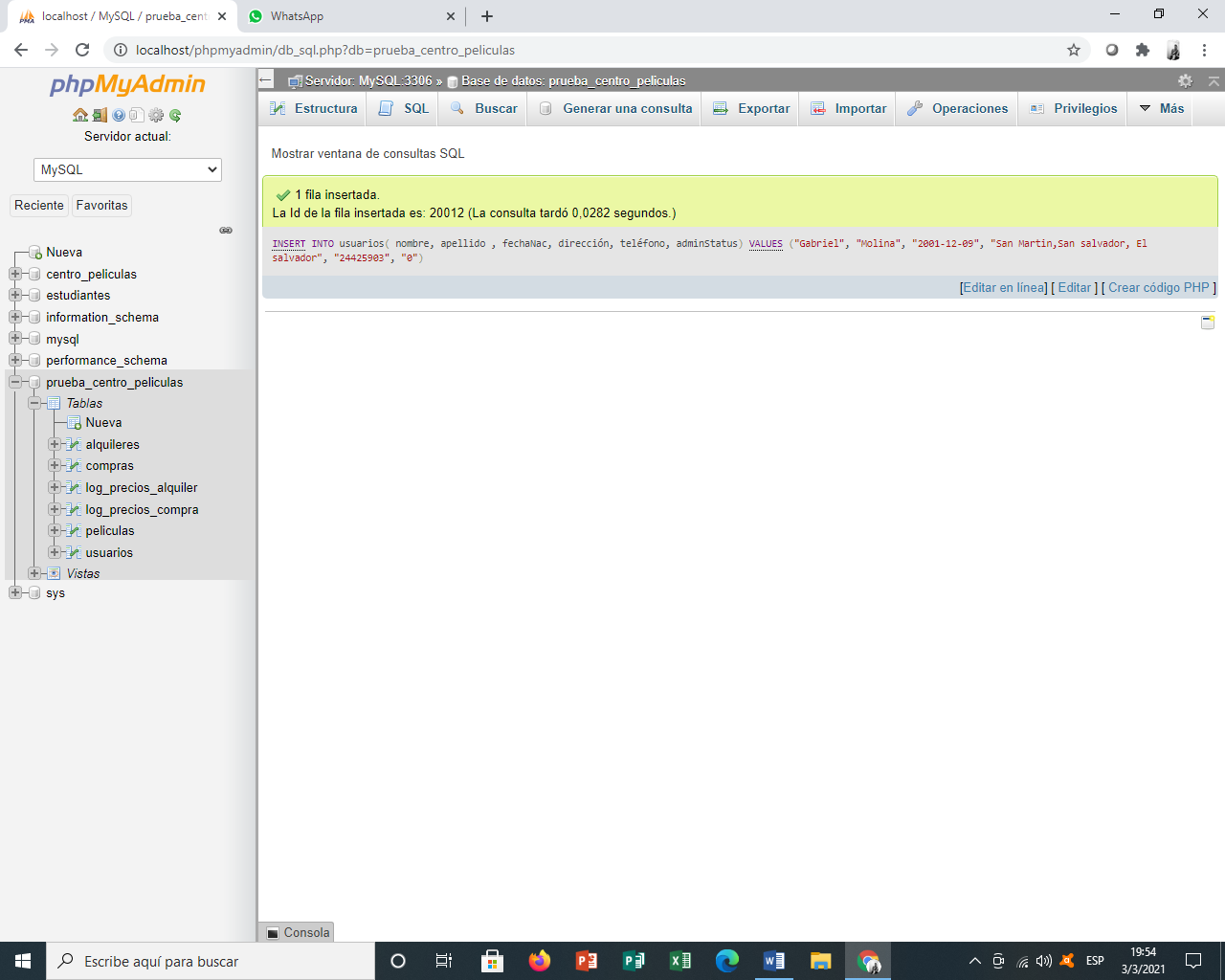
(201801, 'Spider-Man: un nuevo universo', 'Luego de ser mordido por una araña radioactiva, el joven Miles Morales desarrolla misteriosos poderes que lo transforman en el Hombre Araña. Ahora deberá usar sus nuevas habilidades ante el malvado Kingpin, un enorme demente que puede abrir portales hacia otros universos.', 'caratula.jpg', '3.99', '10.00', 6, 6, 300),

(201901, 'Avengers: Endgame', 'Los Vengadores restantes deben encontrar una manera de recuperar a sus aliados para un enfrentamiento épico con Thanos, el malvado que diezmó el planeta y el universo.', 'caratula.jpg', '3.99', '10.00', 6, 7, 500),

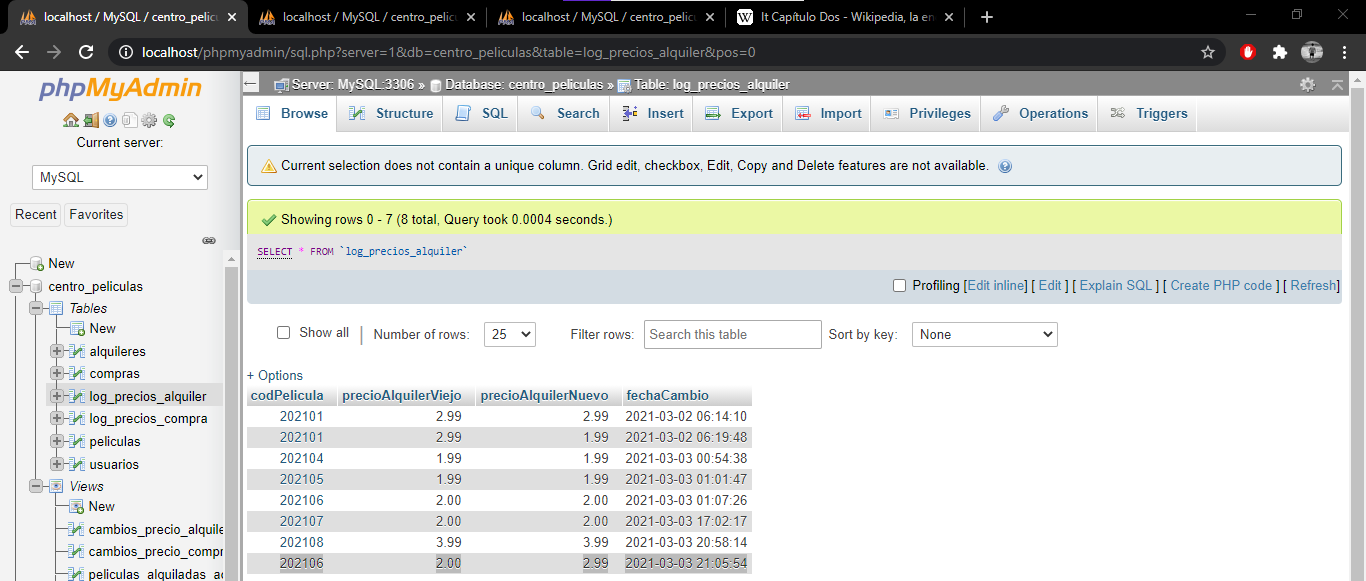
(202101, 'Titanic', 'Titanic es una película estadounidense dramática de catástrofe de 1997 dirigida y escrita por James Cameron y protagonizada por Leonardo DiCaprio, Kate Winslet, Billy Zane, Kathy Bates, Gloria Stuart y Bill Paxton. La trama, una epopeya romántica, relata la relación de Jack Dawson y Rose DeWitt Bukater, dos jóvenes que se conocen y se enamoran a bordo del transatlántico RMS Titanic en su viaje inaugural desde Southampton, Inglaterra, a Nueva York, EE. UU., en abril de 1912. Pertenecientes a diferentes clases sociales, intentan salir adelante pese a las adversidades que los separarían de forma definitiva, entre ellas el prometido de Rose, Caledon «Cal» Hockley (un adinerado del cual ella no está enamorada, pero su madre la ha obligado a permanecer con él para garantizar un futuro económico próspero)', 'test.img', '1.99', '5.99', 5, 6, 450),

(202102, 'Black Widow', 'Al nacer, la Viuda Negra, también conocida como Natasha Romanova, se entrega a la KGB para convertirse en su agente definitivo. Cuando la URSS se separa, el gobierno intenta matarla mientras la acción se traslada a la actual Nueva York.', 'caratula.jpg', '2.99', '8.50', 4, 5, 230);



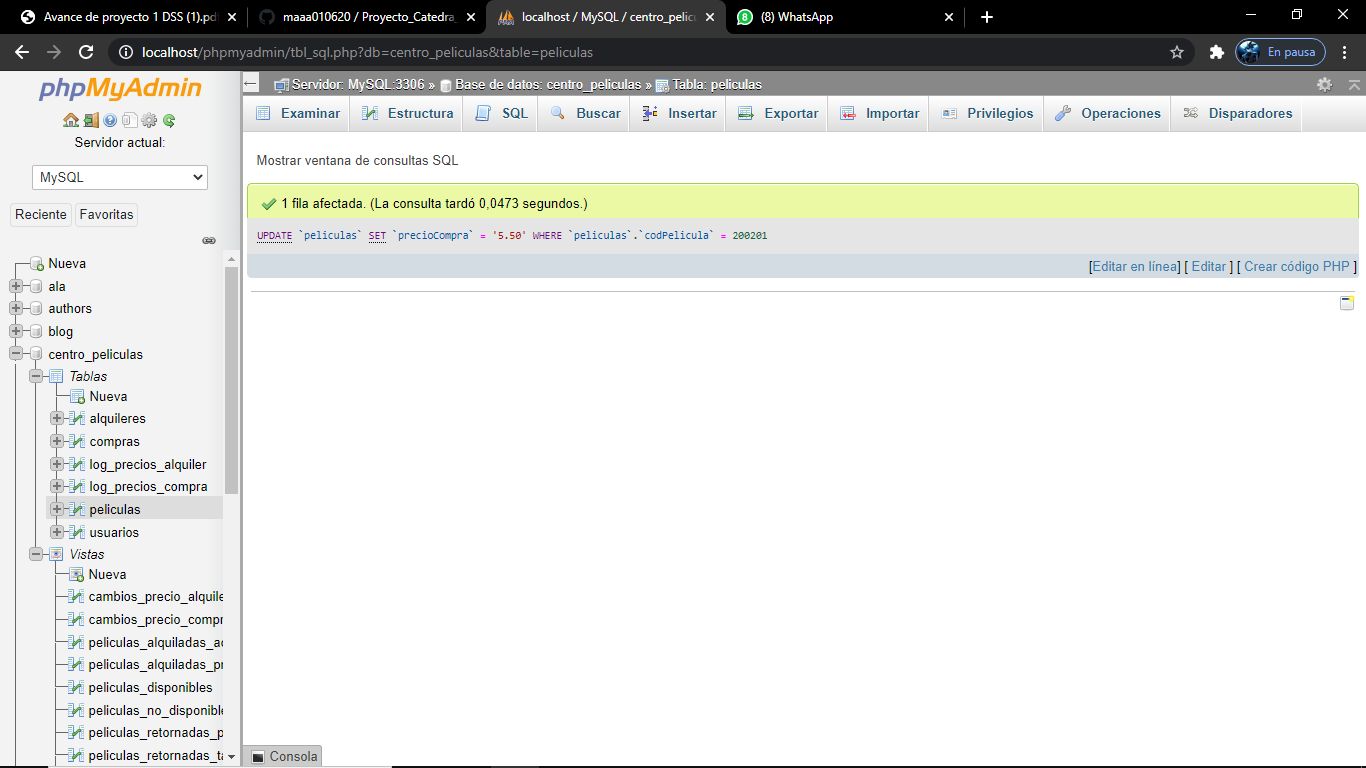
Usamos este comando para poder ingresar un nuevo usuario a la tabla usuario 

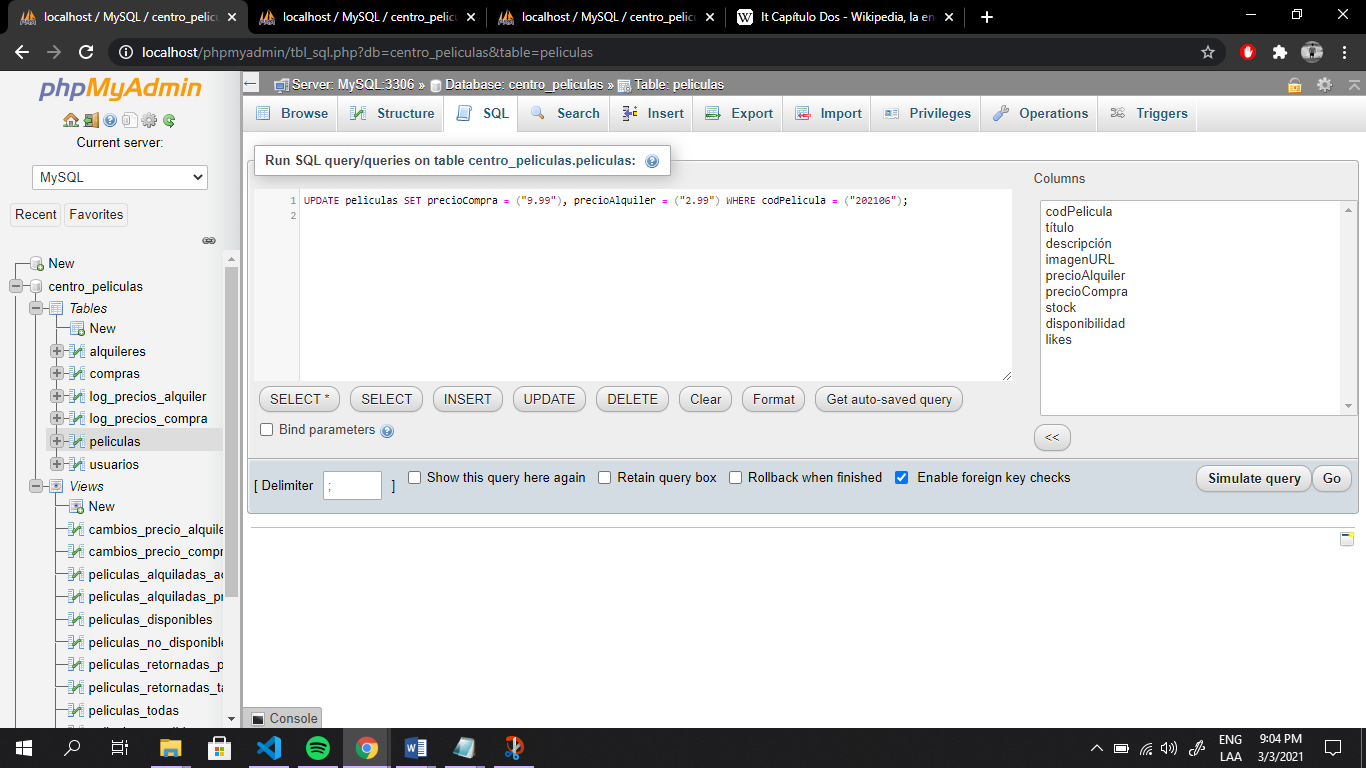
Por medio de un trigger, automáticamente también se hare un INSERT en las tablas de log para el precio de alquiler y el precio de venta cuando estos cambien y se guardara la hora en la que se hizo el cambio.



**MODIFICAR PELICULAS:**

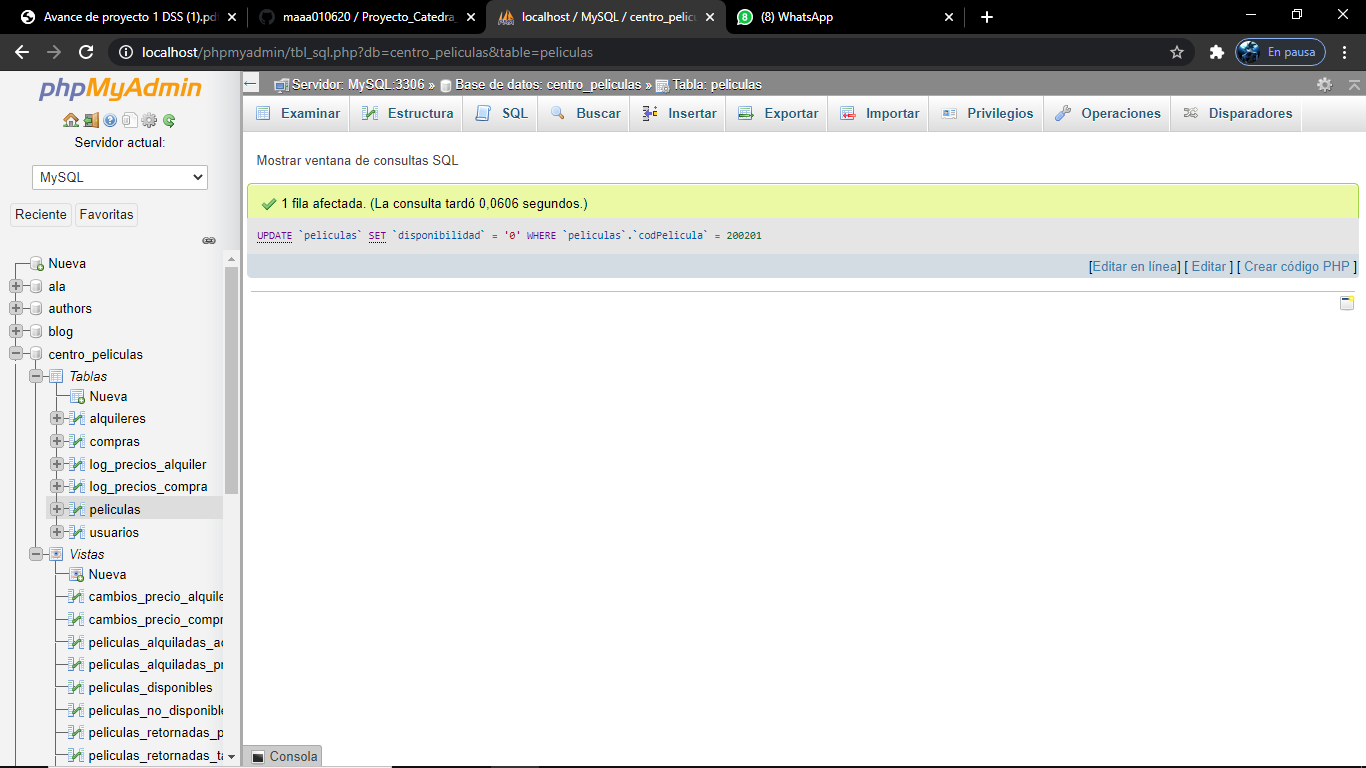
UPDATE `peliculas` SET `precioCompra` = '5.50' WHERE `peliculas`.`codPelicula` = 200201;





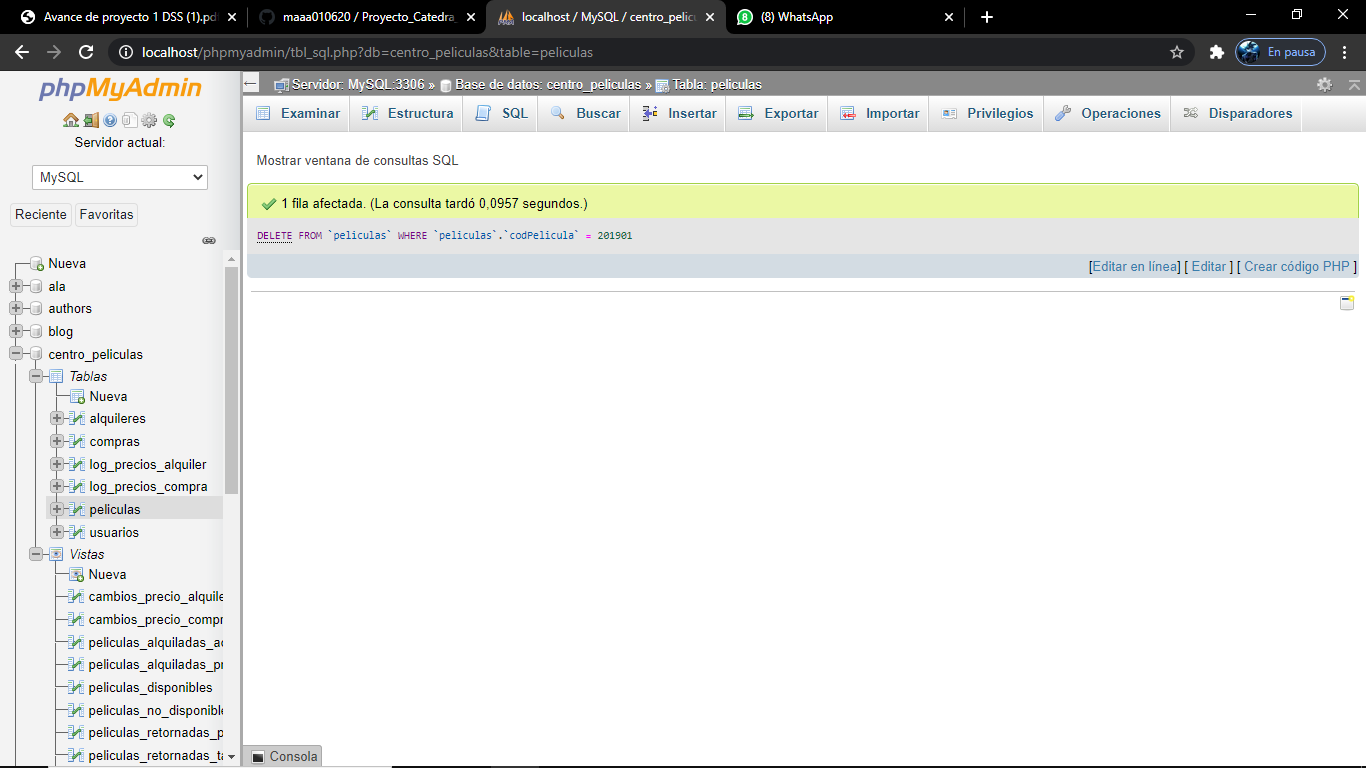
**REMOVER PELICULAS:**

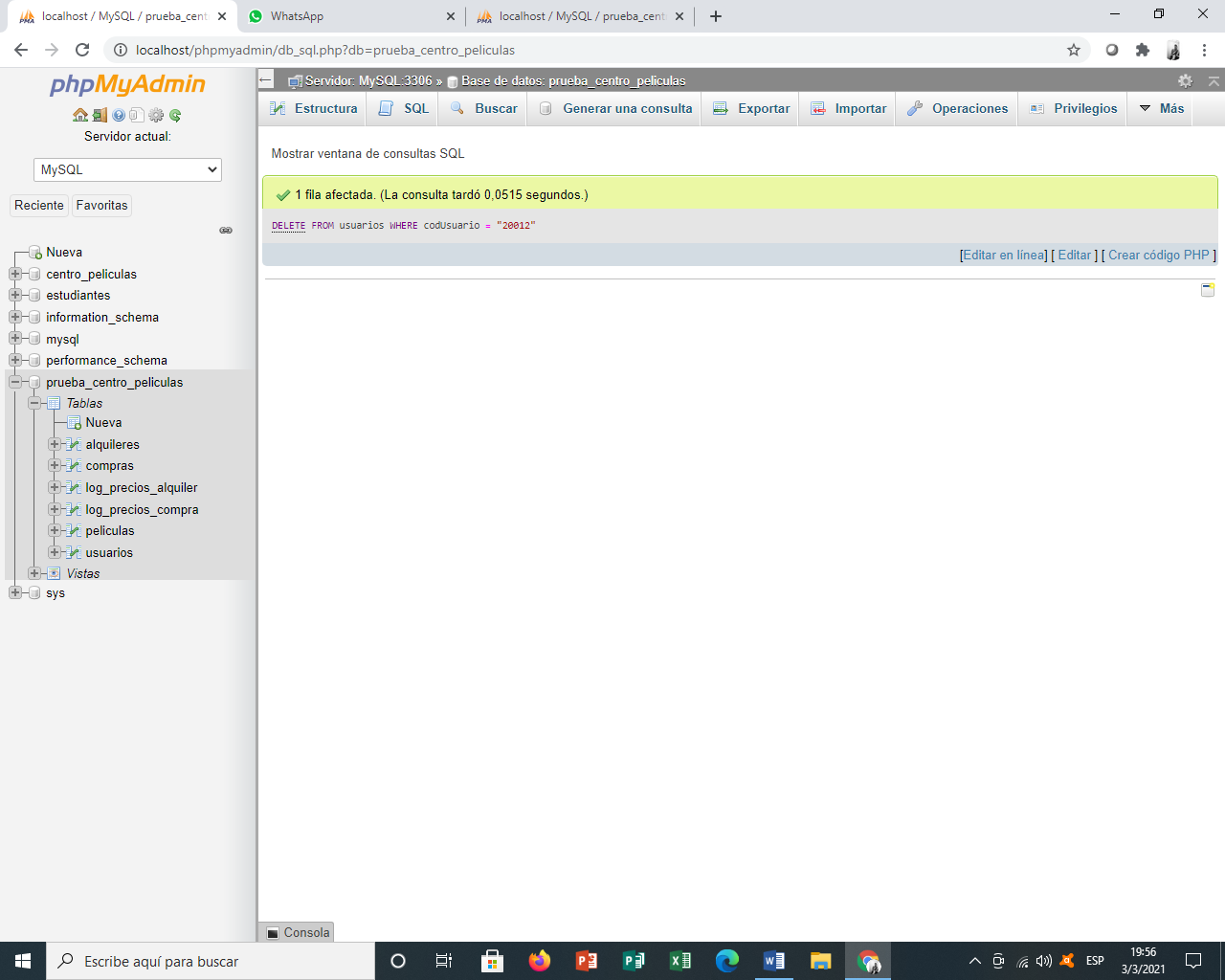
UPDATE `peliculas` SET `disponibilidad` = '0' WHERE `peliculas`.`codPelicula` = 200201;



**BORRAR PELICULAS:**

DELETE FROM `peliculas` WHERE `peliculas`.`codPelicula` = 201901;

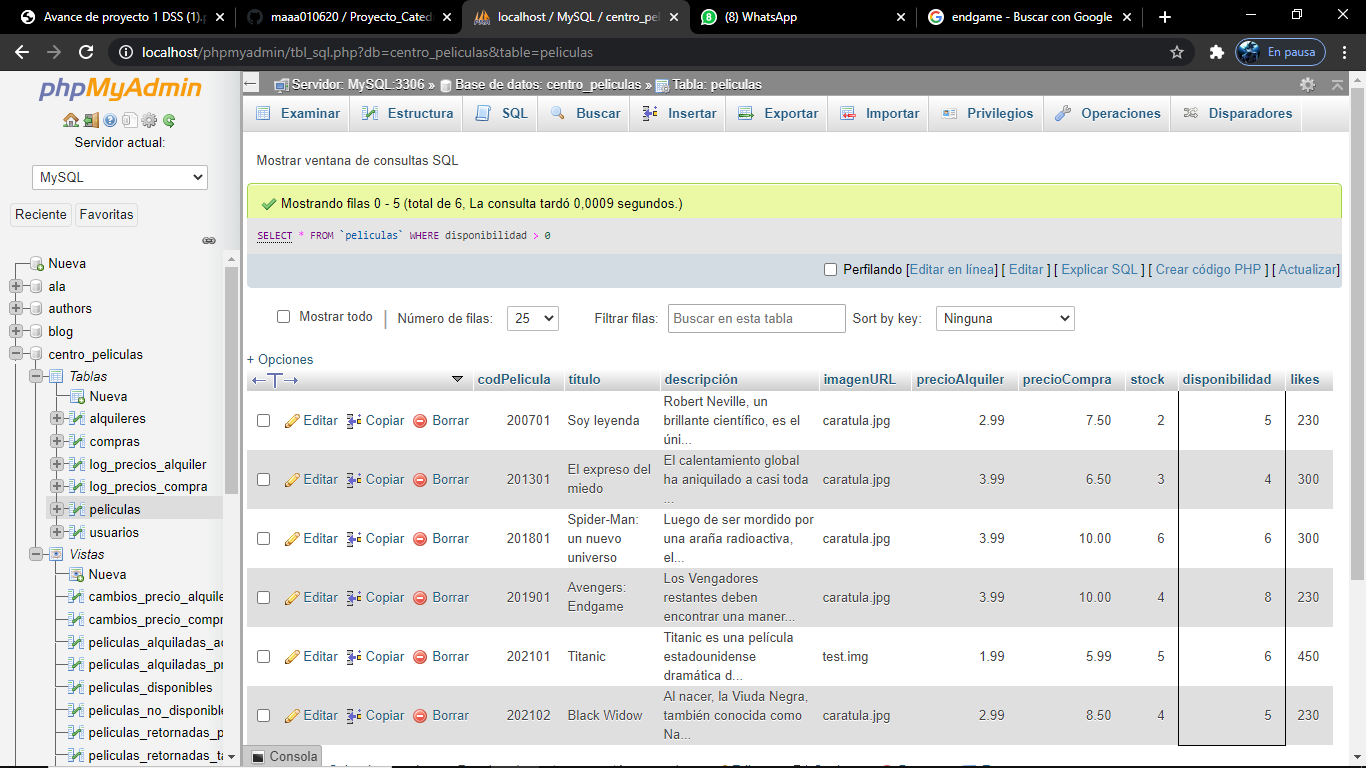
  
BORRAR USUARIOS:

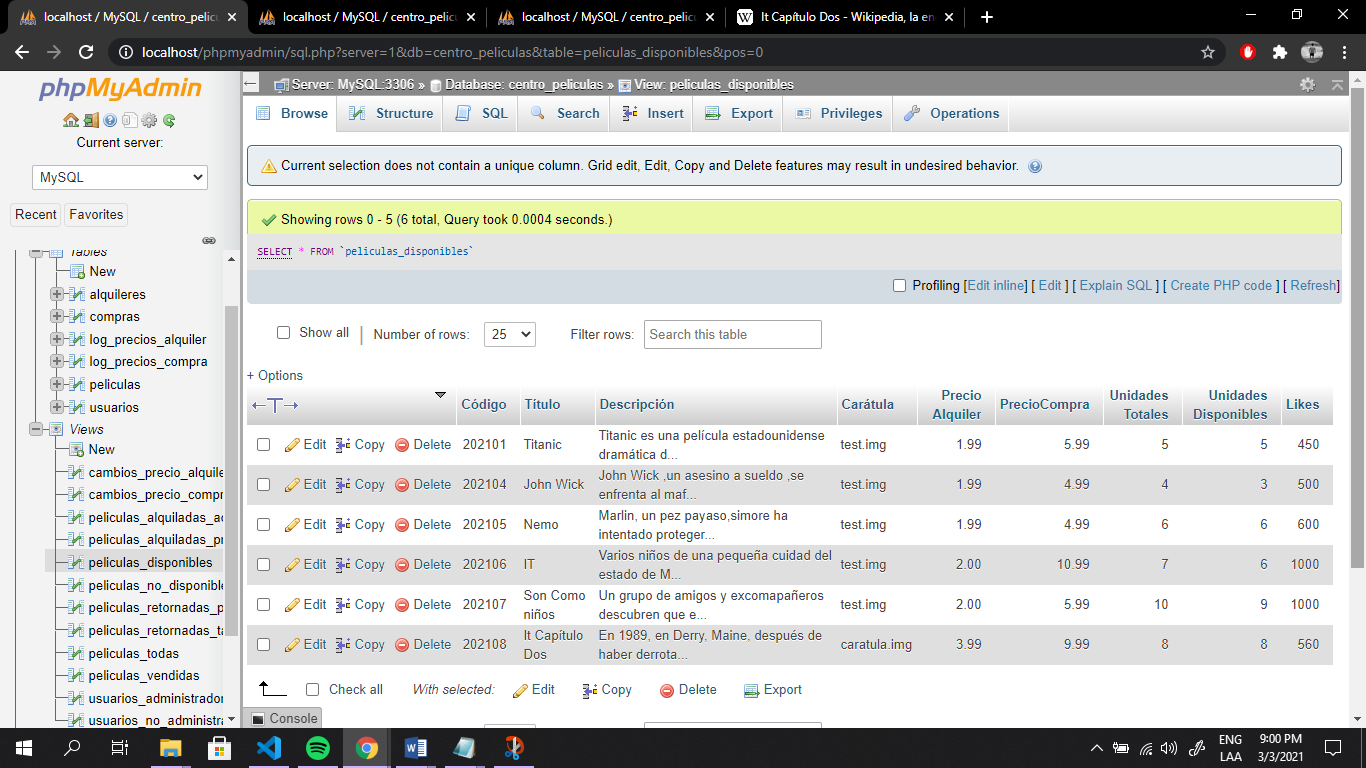


**VISUALIZAR PELICULAS, FILTRAR POR DISPONIBLES Y NO DISPONIBLES:**

**DISPONIBLES:**

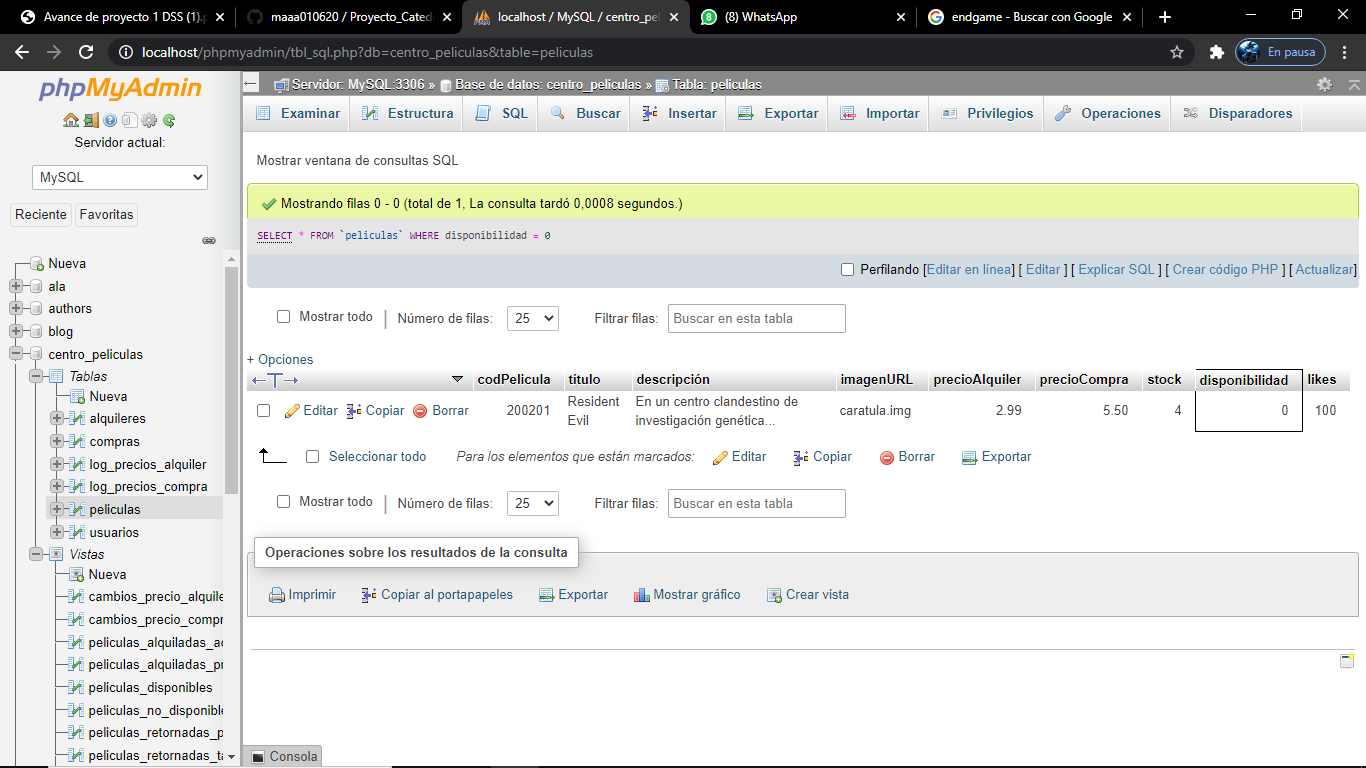
SELECT \* FROM `peliculas` WHERE disponibilidad > 0;

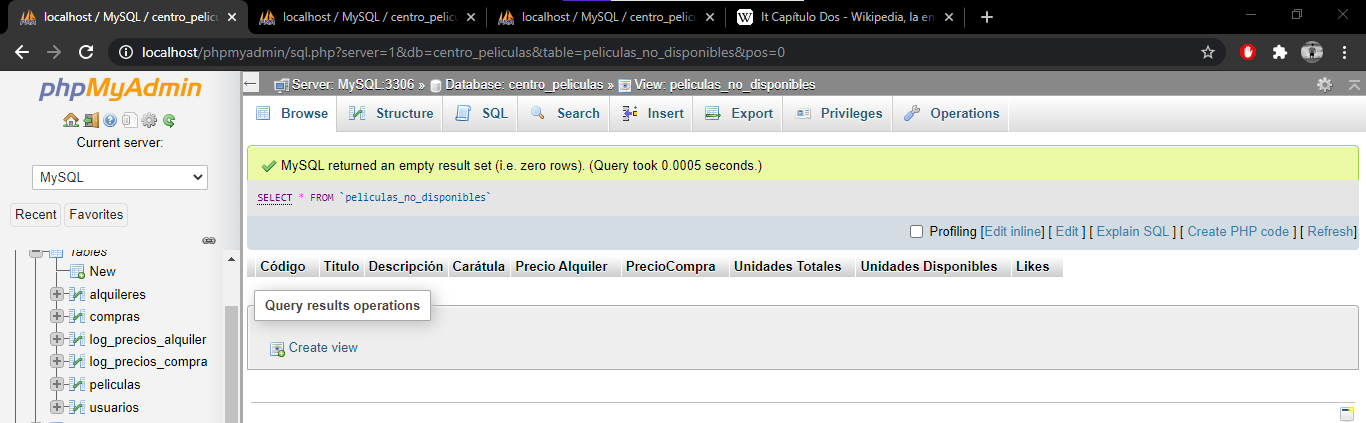




**NO DISPONIBLES:**

SELECT \* FROM `peliculas` WHERE disponibilidad = 0;

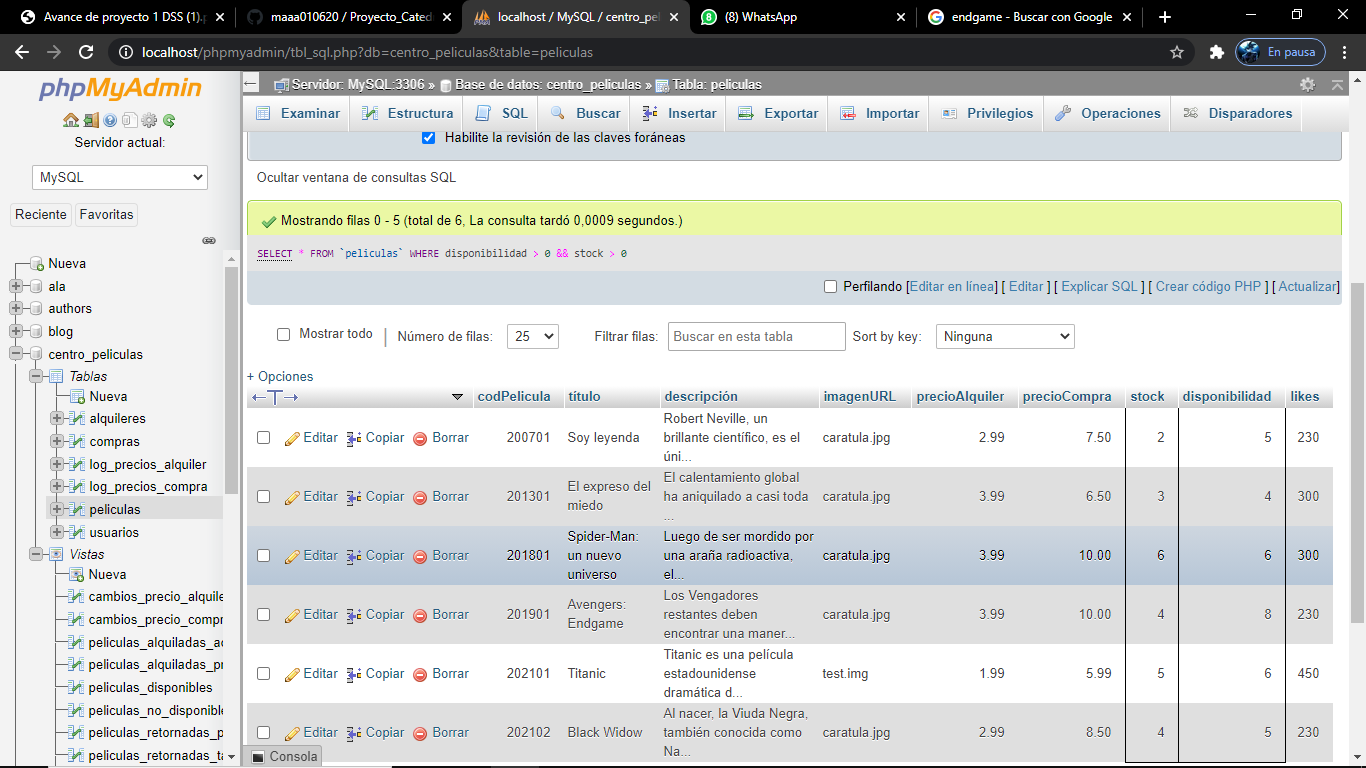




**PELICULAS DISPONIBLES PARA COMPRAR O ALQUILAR:**

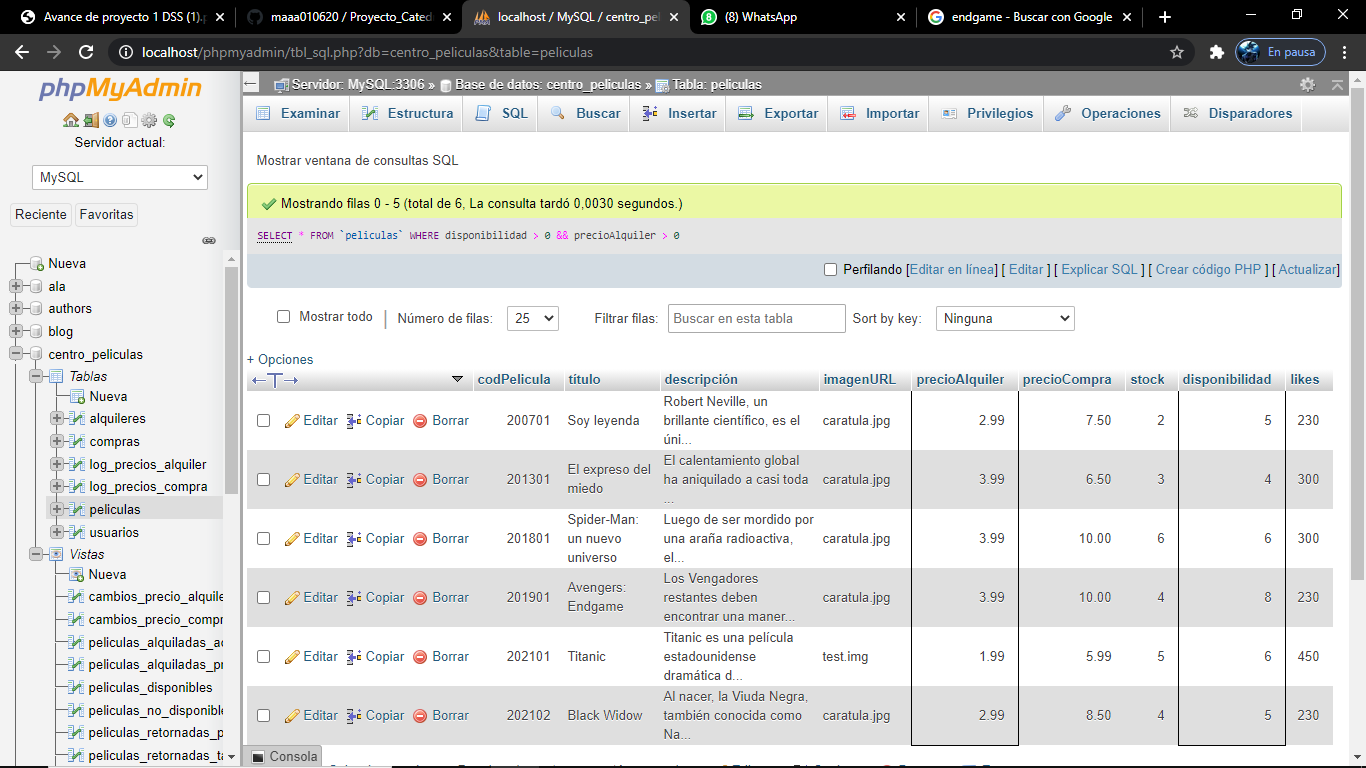
**COMPRAR:**

SELECT \* FROM `peliculas` WHERE disponibilidad > 0 && stock > 0

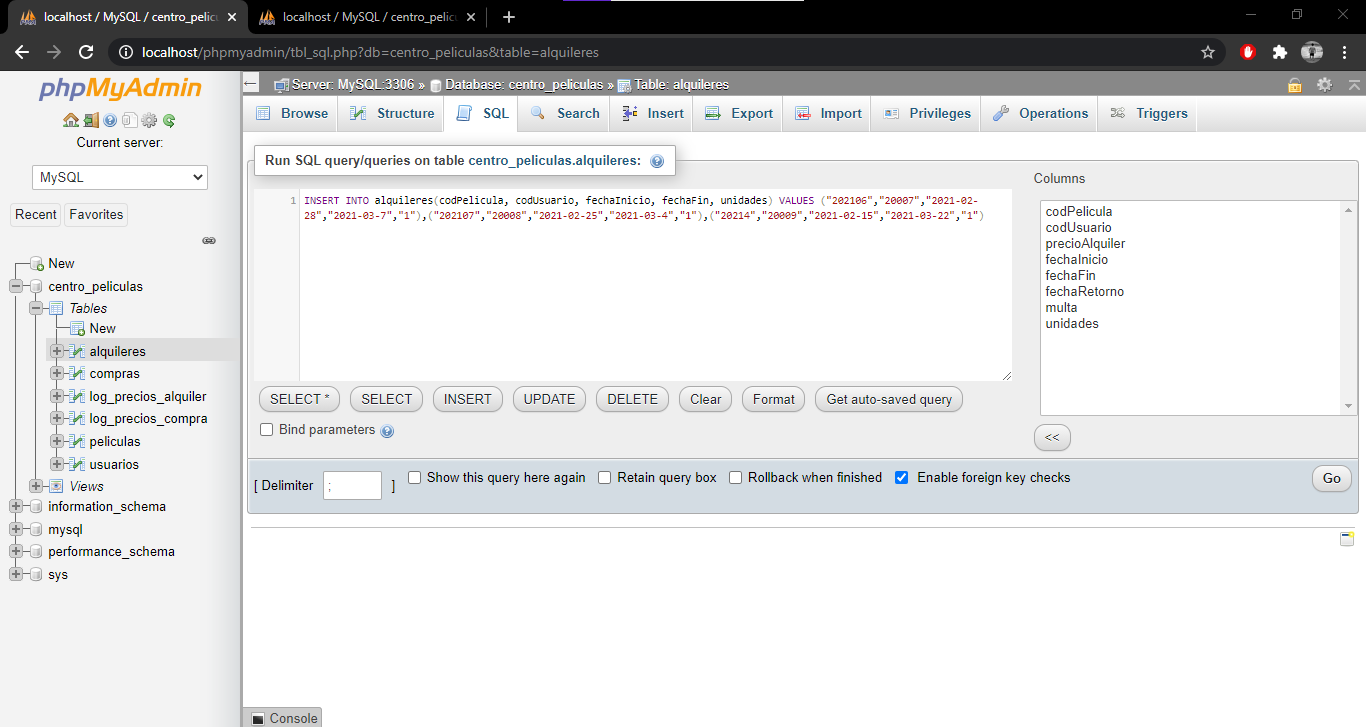


**ALQUILAR:**

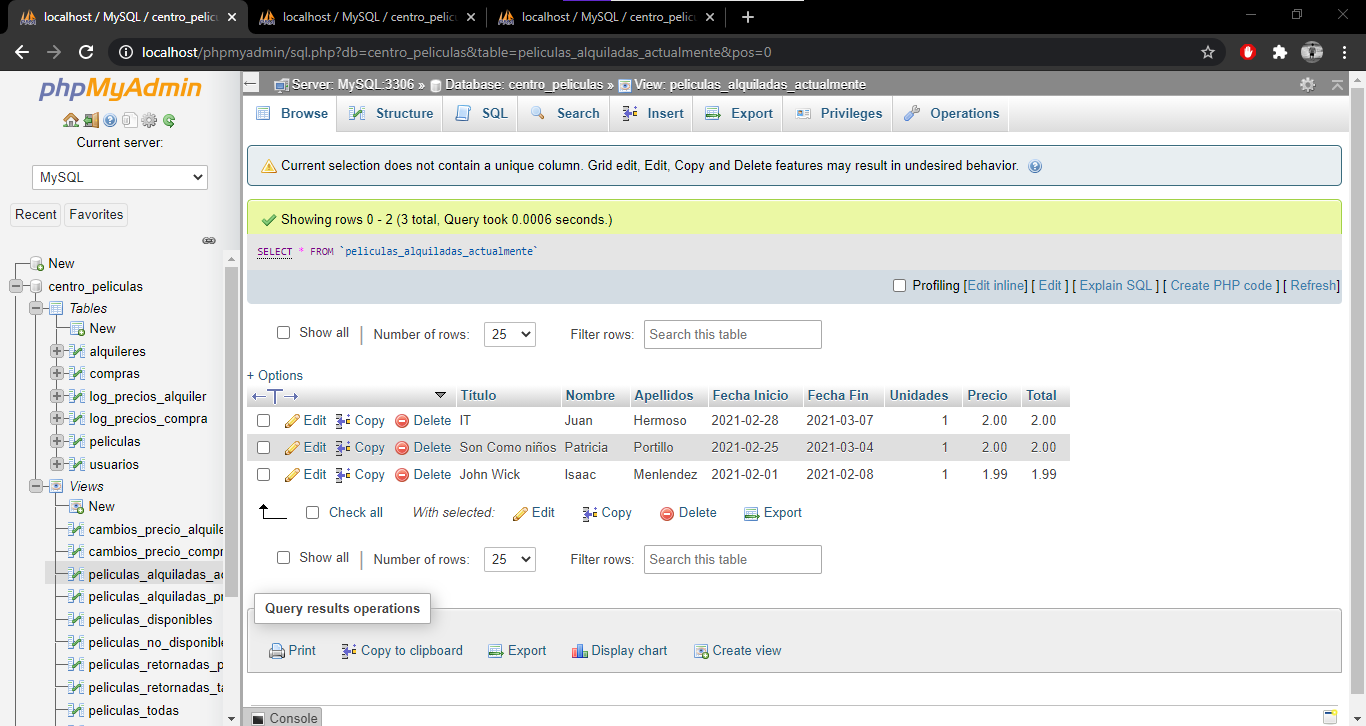
SELECT \* FROM `peliculas` WHERE disponibilidad > 0 && precioAlquiler > 0



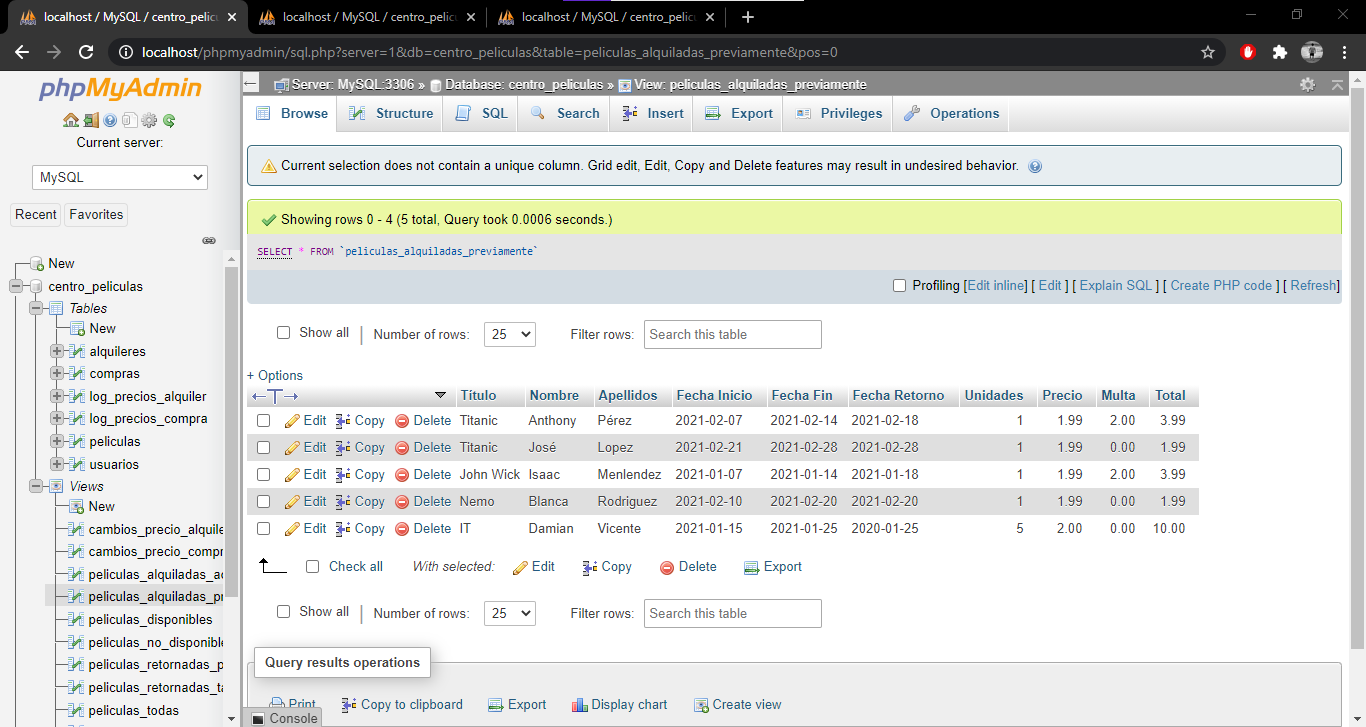
Con este comando podemos registrar el alquiler de una película.



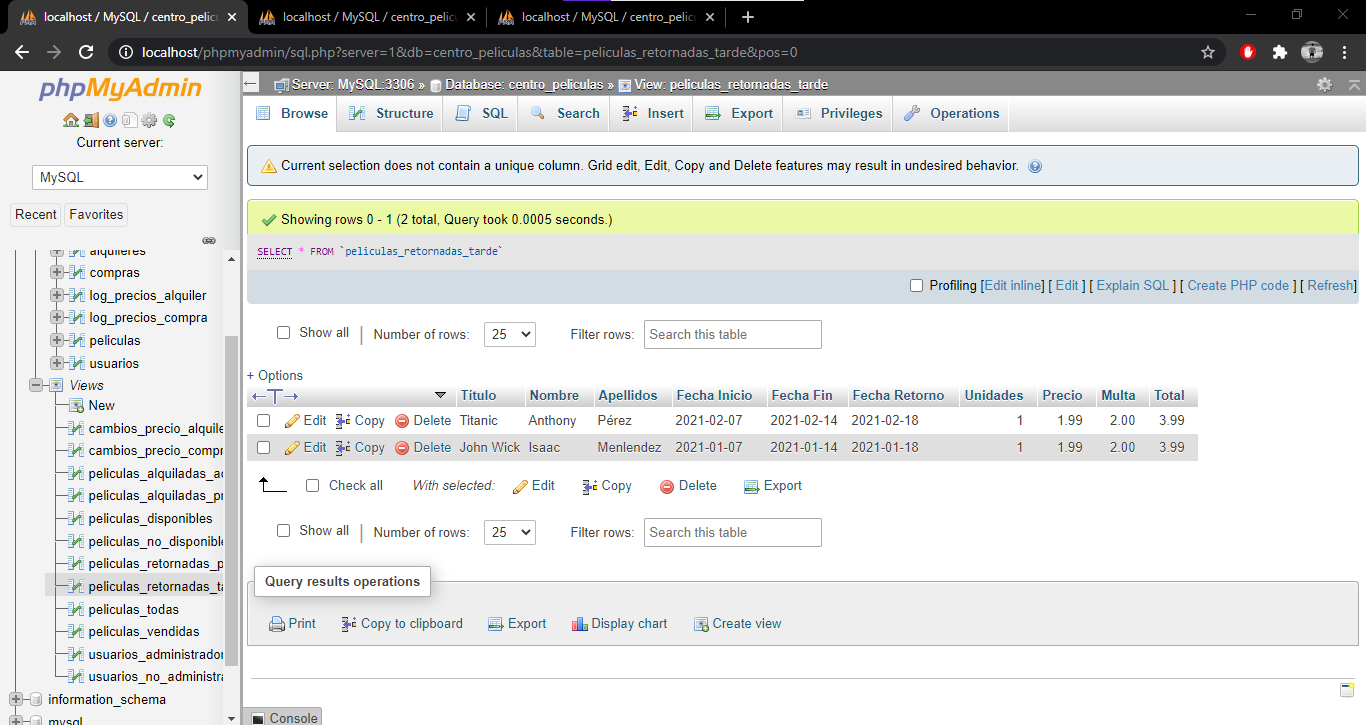
Con este comando podemos ver la vista de películas rentadas que no han sido regresadas



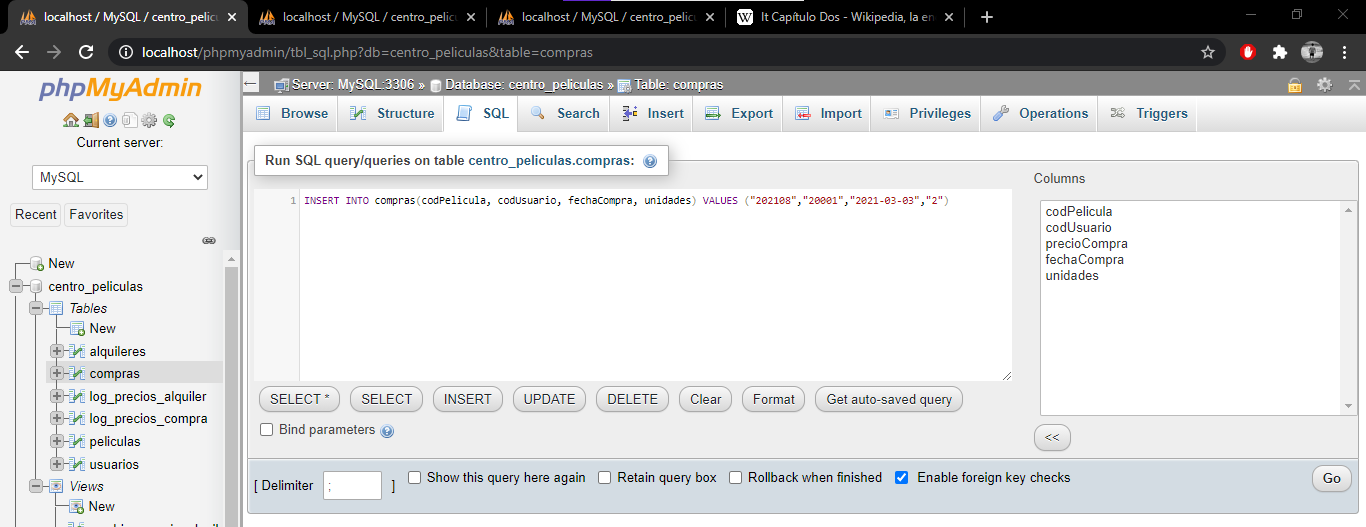
Con este comando podemos ver la vista de películas que se han rentado previamente y han sido regresadas



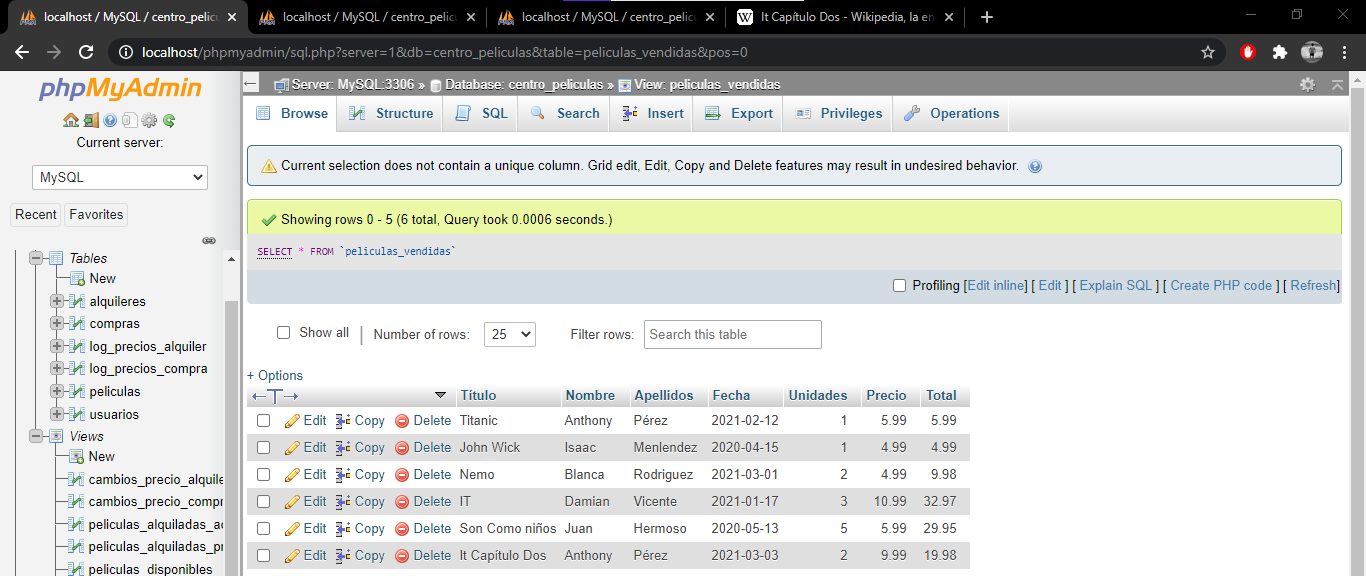
Con este comando podemos ver la vista de películas que se regresaron tarde y se aplicó un cargo de mora de $0.50 centavos por día tarde.



Similarmente, con el siguiente comando podemos hacer un INSERT para la realización de una compra de película.



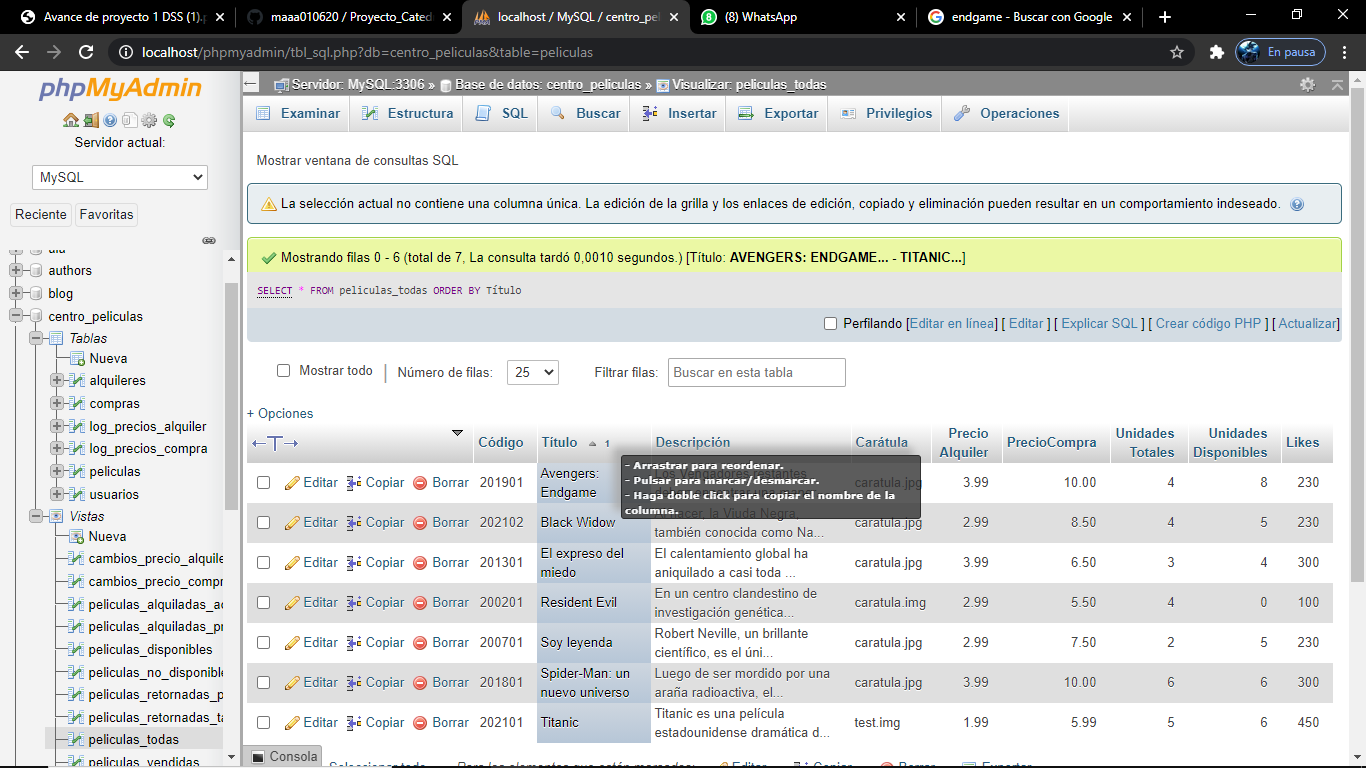
Y podemos ver las compras de película que se han realizado con el siguiente SELECT y vista



**LISTAR PELICULAS POR TITULO Y POPULARIDAD:**

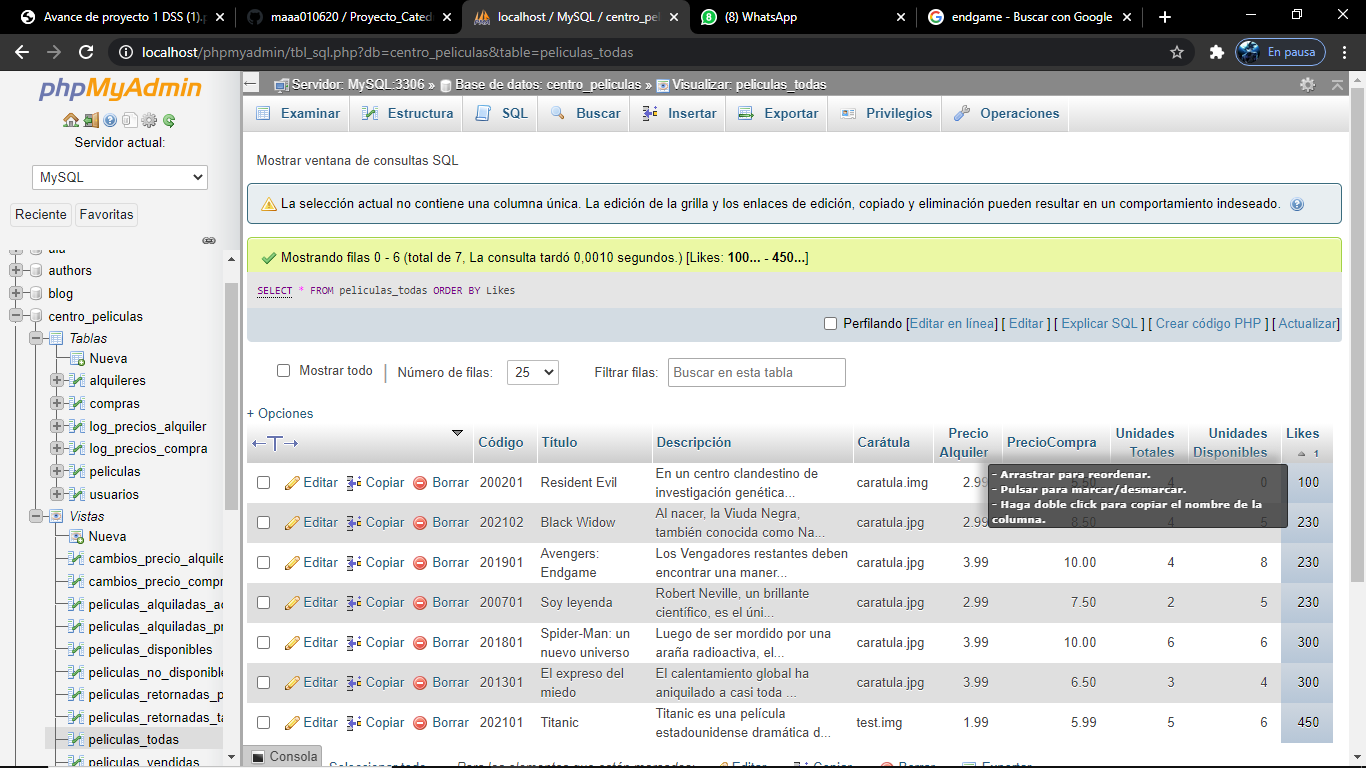
**TITULO:**

SELECT \* FROM peliculas\_todas ORDER BY Título



**POPULARIDAD:**

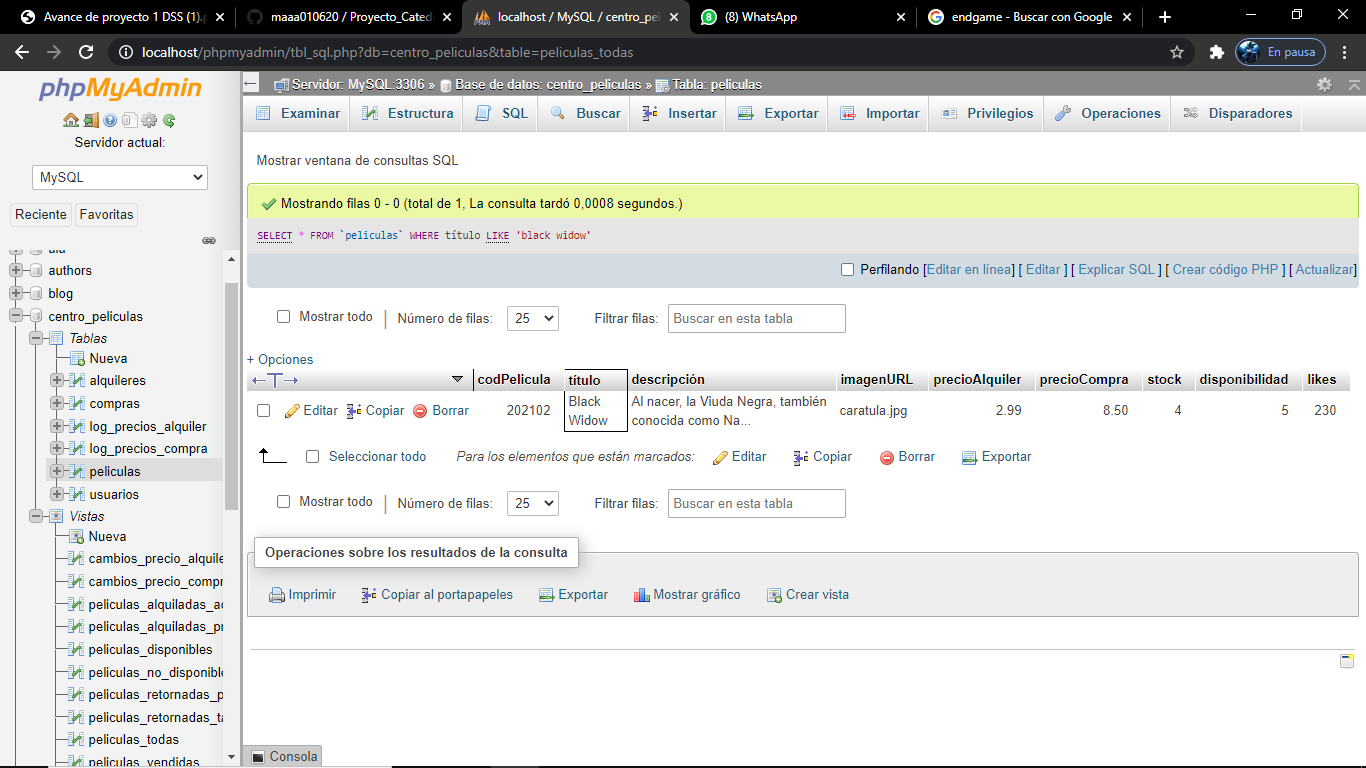
SELECT \* FROM peliculas\_todas ORDER BY Likes



**BUSCAR PELICULAS POR NOMBRE:**

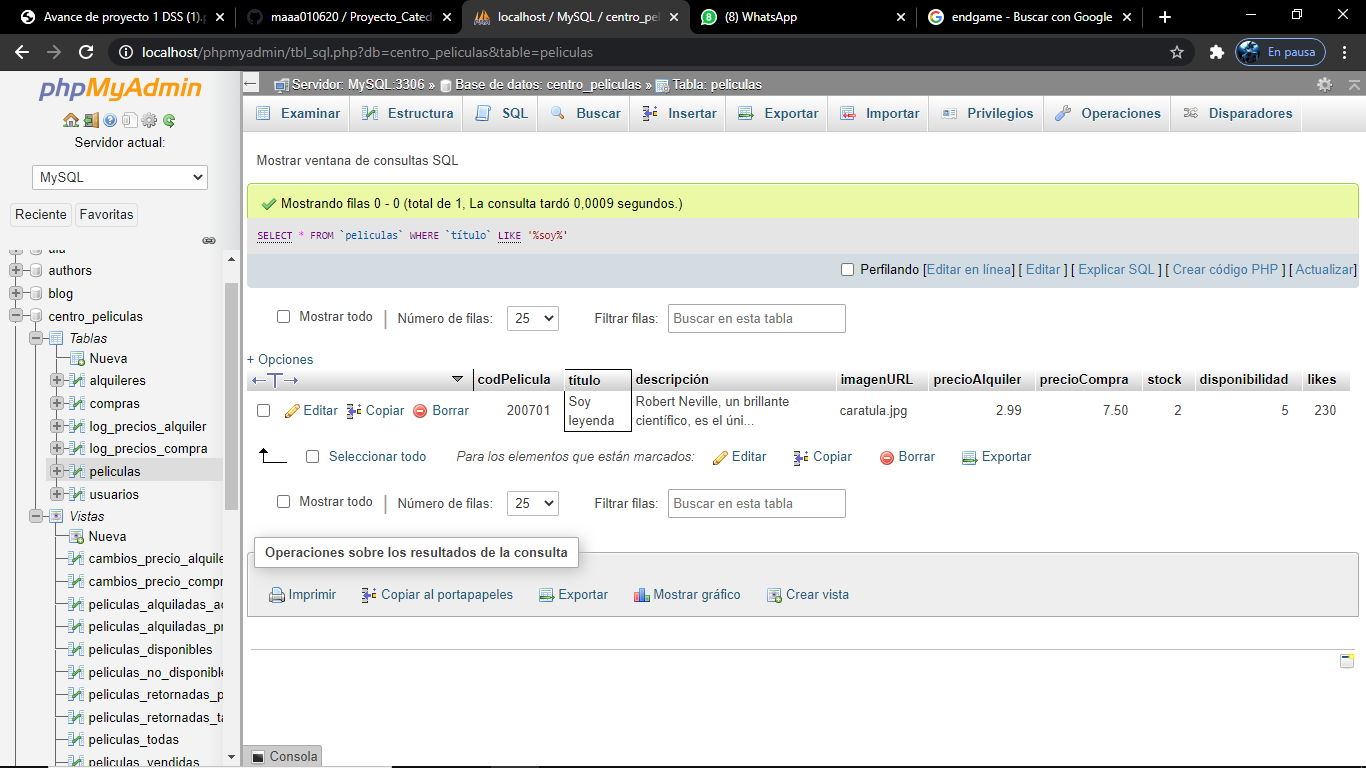
**PELICULA POR NOMBRE:**

SELECT \* FROM `peliculas` WHERE título LIKE 'black widow'

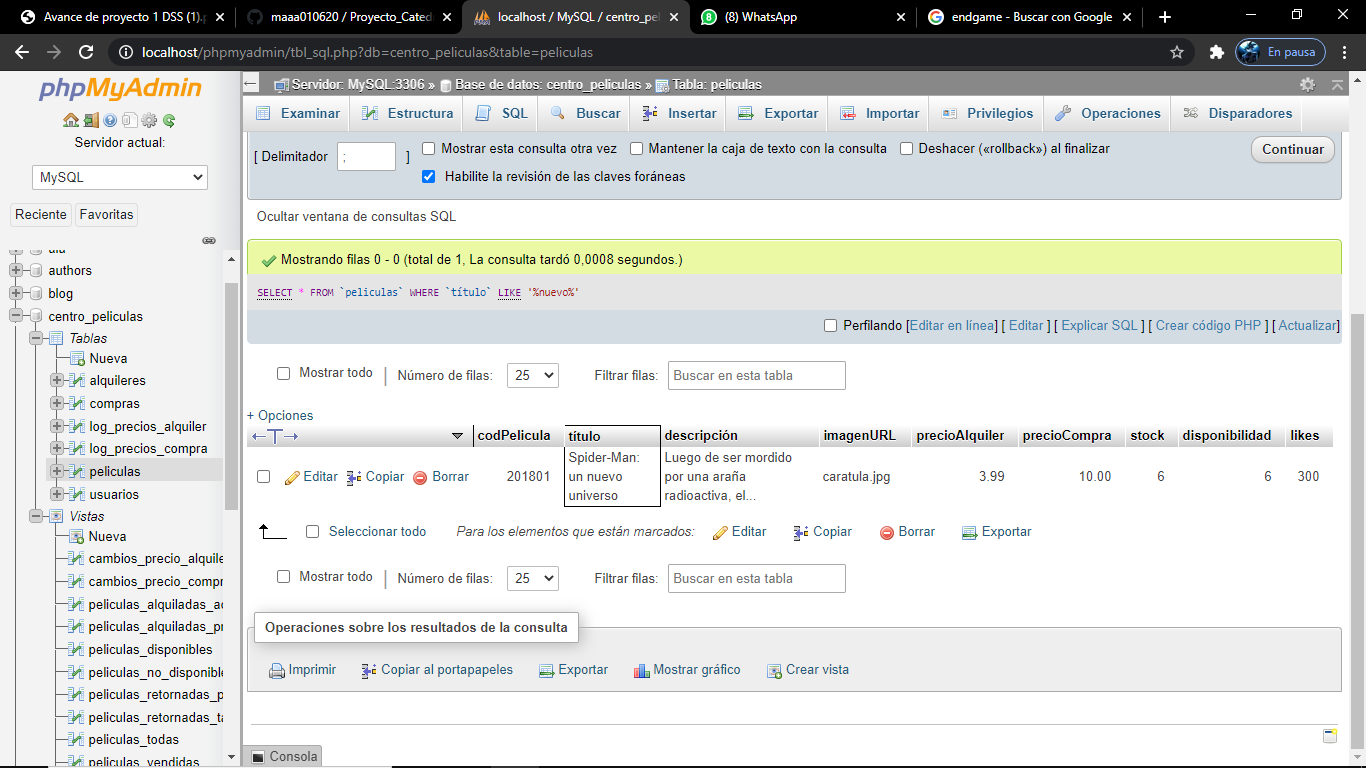


**PELICULA QUE COINCIDA CON:**

SELECT \* FROM `peliculas` WHERE `título` LIKE '%soy%'



SELECT \* FROM `peliculas` WHERE `título` LIKE '%nuevo%'

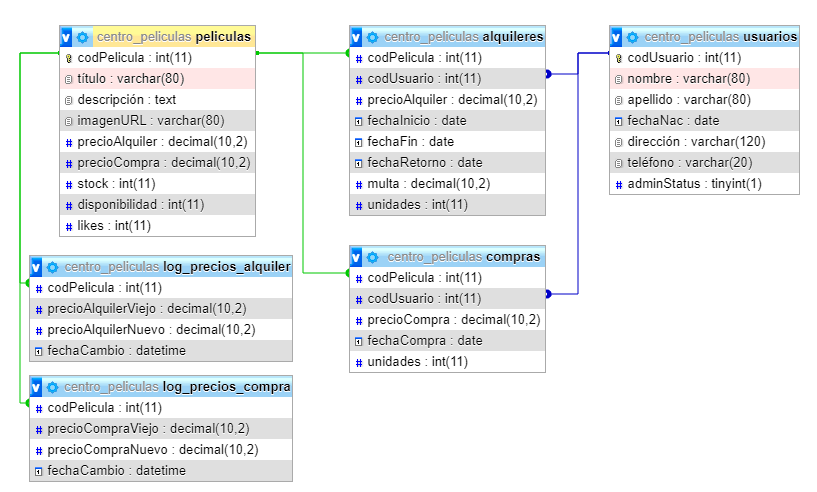


**CONSTRAINTS Y TRIGGERS IMPLEMENTADOS:**

|  |  |
| --- | --- |
| **Tabla Peliculas** | |
| *CREATE TRIGGER `cambio\_precio\_alquiler` AFTER UPDATE ON `peliculas`*  *FOR EACH ROW IF old.precioAlquiler <> new.precioAlquiler THEN*  *INSERT INTO log\_precios\_alquiler (codPelicula, precioAlquilerViejo, precioAlquilerNuevo, fechaCambio)*  *VALUES (new.codPelicula, old.precioAlquiler, new.precioAlquiler,NOW());*  *END IF* | Este trigger nos permite llenar la tabla log\_precios\_alquiler con el precio viejo, nuevo y fecha de cambio cuando se actualiza el precio de alquiler de una película. |
| *CREATE TRIGGER `cambio\_precio\_compra` AFTER UPDATE ON `peliculas`*  *FOR EACH ROW IF old.precioCompra <> new.precioCompra THEN*  *INSERT INTO log\_precios\_compra (codPelicula, precioCompraViejo, precioCompraNuevo, fechaCambio)*  *VALUES (new.codPelicula, old.precioCompra, new.precioCompra,NOW());*  *END IF* | Este trigger nos permite llenar la tabla log\_precios\_compra con el precio viejo, nuevo y fecha de cambio cuando se actualiza el precio de alquiler de una película. |
| *CREATE TRIGGER `check\_positivos\_peliculas` BEFORE INSERT ON `peliculas`*  *FOR EACH ROW IF new.precioAlquiler<0 OR new.precioCompra<0 OR new.stock<0 OR new.likes<0 OR new.disponibilidad<0*  *THEN SIGNAL SQLSTATE '12345'*  *SET MESSAGE\_TEXT*  *= "Valores negativos no son admitidos";*  *END IF* | Este trigger nos permite validar que los valores de entrada de los campos de película no sean negativos. Ya que mysql no permite el uso de CHECK, se necesita el uso de un trigger para que este verifique y no permita el insert si no son válidos los valores. |
| *CREATE TRIGGER `registrar\_precio\_original\_alquiler` AFTER INSERT ON `peliculas`*  *FOR EACH ROW INSERT INTO log\_precios\_alquiler (codPelicula, precioAlquilerViejo, precioAlquilerNuevo, fechaCambio)*  *VALUES (new.codPelicula, new.precioAlquiler, new.precioAlquiler, NOW())* | Este trigger nos permite registrar el precio original de alquiler de una película cuando se agrega como punto de partida en la tabla de log\_precios\_alquiler. También sirve para registrar en qué fecha se agregó la película. |
| *CREATE TRIGGER `registrar\_precio\_original\_compra` AFTER INSERT ON `peliculas`*  *FOR EACH ROW INSERT INTO log\_precios\_compra (codPelicula, precioCompraViejo, precioCompraNuevo, fechaCambio)*  *VALUES (new.codPelicula, new.precioCompra, new.precioCompra,NOW())* | Este trigger nos permite registrar el precio original de compra de una película cuando se agrega como punto de partida en la tabla de log\_precios\_compra. También sirve para registrar en qué fecha se agregó la película. |
| **Tabla Compras** | |
| *CREATE TRIGGER `verificar\_disponibilidad\_compra` BEFORE INSERT ON `compras`*  *FOR EACH ROW IF new.unidades>(SELECT disponibilidad FROM peliculas WHERE codPelicula = new.codPelicula)*  *THEN SIGNAL SQLSTATE '12345'*  *SET MESSAGE\_TEXT*  *= "NO EXISTE DISPONIBILIDAD";*  *ELSE*  *UPDATE peliculas SET stock = stock-new.unidades, disponibilidad = disponibilidad-new.unidades WHERE codPelicula = new.codPelicula;*  *END IF* | Este trigger nos permite ver si hay stock disponible para comprar la película antes de realizar un insert en la tabla de compras. Si no hay stock, no se permite el insert y si hay disponibilidad, se completa el insert y se actualiza el stock y disponibilidad en la tabla películas. |
| *CREATE TRIGGER `agregar\_precio\_actual\_compra` BEFORE INSERT ON `compras`*  *FOR EACH ROW SET new.precioCompra = (SELECT precioCompra FROM peliculas WHERE codPelicula= new.codPelicula)* | Este trigger nos permite referenciar la tabla de películas y agregar el precio actual de la compra cuando se realiza un insert en la tabla para que no se tenga que ingresar manualmente. |
| *CREATE TRIGGER `check\_positivos\_compras` BEFORE INSERT ON `compras`*  *FOR EACH ROW IF new.unidades<0 THEN SIGNAL SQLSTATE '12345' SET MESSAGE\_TEXT = "Valores negativos no son admitidos"; END IF* | Este trigger realiza la función de un CHECK ya que no se puede ingresar un valor negativo de unidades para una compra. |
| **Tabla Alquiler** | |
| *CREATE TRIGGER `verificar\_disponibilidad\_alquiler` BEFORE INSERT ON `alquileres`*  *FOR EACH ROW IF new.unidades>(SELECT disponibilidad FROM peliculas WHERE codPelicula = new.codPelicula)*  *THEN SIGNAL SQLSTATE '12345'*  *SET MESSAGE\_TEXT*  *= "NO EXISTE DISPONIBILIDAD";*  *ELSE*  *UPDATE peliculas SET disponibilidad = disponibilidad-new.unidades WHERE codPelicula = new.codPelicula;*  *END IF* | Este trigger nos permite ver si hay stock disponible para alquilar la película antes de realizar un insert en la tabla de alquileres. Si no hay stock, no se permite el insert y si hay disponibilidad, se completa el insert y se actualiza la disponibilidad en la tabla películas. |
| *CREATE TRIGGER `agregar\_precio\_actual\_alquiler` BEFORE INSERT ON `alquileres`*  *FOR EACH ROW SET new.precioAlquiler = (SELECT precioAlquiler FROM peliculas WHERE codPelicula= new.codPelicula)* | Este trigger nos permite referenciar la tabla de películas y agregar el precio actual de la compra cuando se realiza un insert en la tabla para que no se tenga que ingresar manualmente. |
| *CREATE TRIGGER `check\_positivos\_alquileres` BEFORE INSERT ON `alquileres`*  *FOR EACH ROW IF new.unidades<0 THEN SIGNAL SQLSTATE '12345'*  *SET MESSAGE\_TEXT*  *= "Valores negativos no son admitidos";*  *END IF* | Este trigger realiza la función de un CHECK ya que no se puede ingresar un valor negativo de unidades para un alquiler. |
| *CREATE TRIGGER `check\_fechafin\_mayor\_fechainicio` BEFORE INSERT ON `alquileres`*  *FOR EACH ROW IF new.fechaFin < new.fechaInicio THEN SIGNAL SQLSTATE '12345'*  *SET MESSAGE\_TEXT*  *= "Rango de fechas no es valido";*  *END IF* | Este trigger realiza la función de un CHECK ya que la fecha de fin del periodo de alquiler no puede ser menor (estar antes de) la fecha de inicio. Al ser ese el caso, no se permite el insert. |
| *CREATE TRIGGER `regreso\_pelicula` BEFORE UPDATE ON `alquileres`*  *FOR EACH ROW IF new.fechaRetorno > old.fechaFin THEN*  *SET new.multa = old.unidades\*0.50\*DATEDIFF( new.fechaRetorno,old.fechaFin);*  *UPDATE peliculas SET disponibilidad = disponibilidad+old.unidades WHERE codPelicula = new.codPelicula;*  *ELSE*  *UPDATE peliculas SET disponibilidad = disponibilidad+old.unidades WHERE codPelicula = new.codPelicula;*  *END IF* | Este trigger realiza la función de comparar la fecha de retorno con la fecha de fin de un alquiler. Si la fecha de retorno se pasa de la fecha de fin, se calcula una multa basada en la cantidad de días tarde que se realiza el retorno de la película. También este trigger actualiza el stock de la película en la tabla películas. |

**ESQUEMA BD.**

**centro\_peliculas**

****