PROYECTO

2ºTrimestre

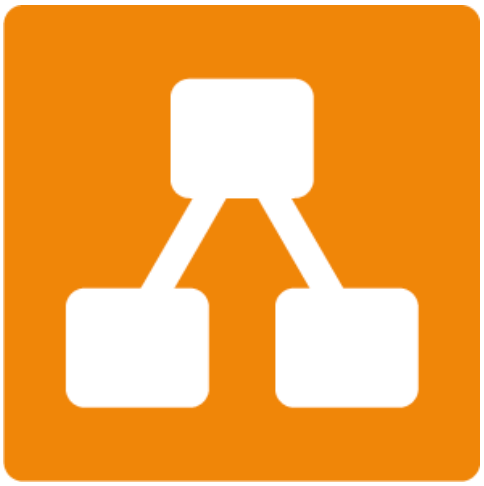
Base de Datos

Álvaro Castillero Saravia

1ºDAW

IES Alixar

24/25



1. Introducción

Este proyecto está basado en una plataforma que opera como un mercado global para la compra y ventas de productos de edición limitada, se enfoca en las zapatillas, ropa urbana, relojes, accesorios y artículos coleccionables. Este mercado se distingue por su modelo de mercado, donde los precios suben y bajan según la oferta y la demanda, parecido a cómo pasa en la bolsa de valores.

2. Modelo Entidad Relación (Draw.io)

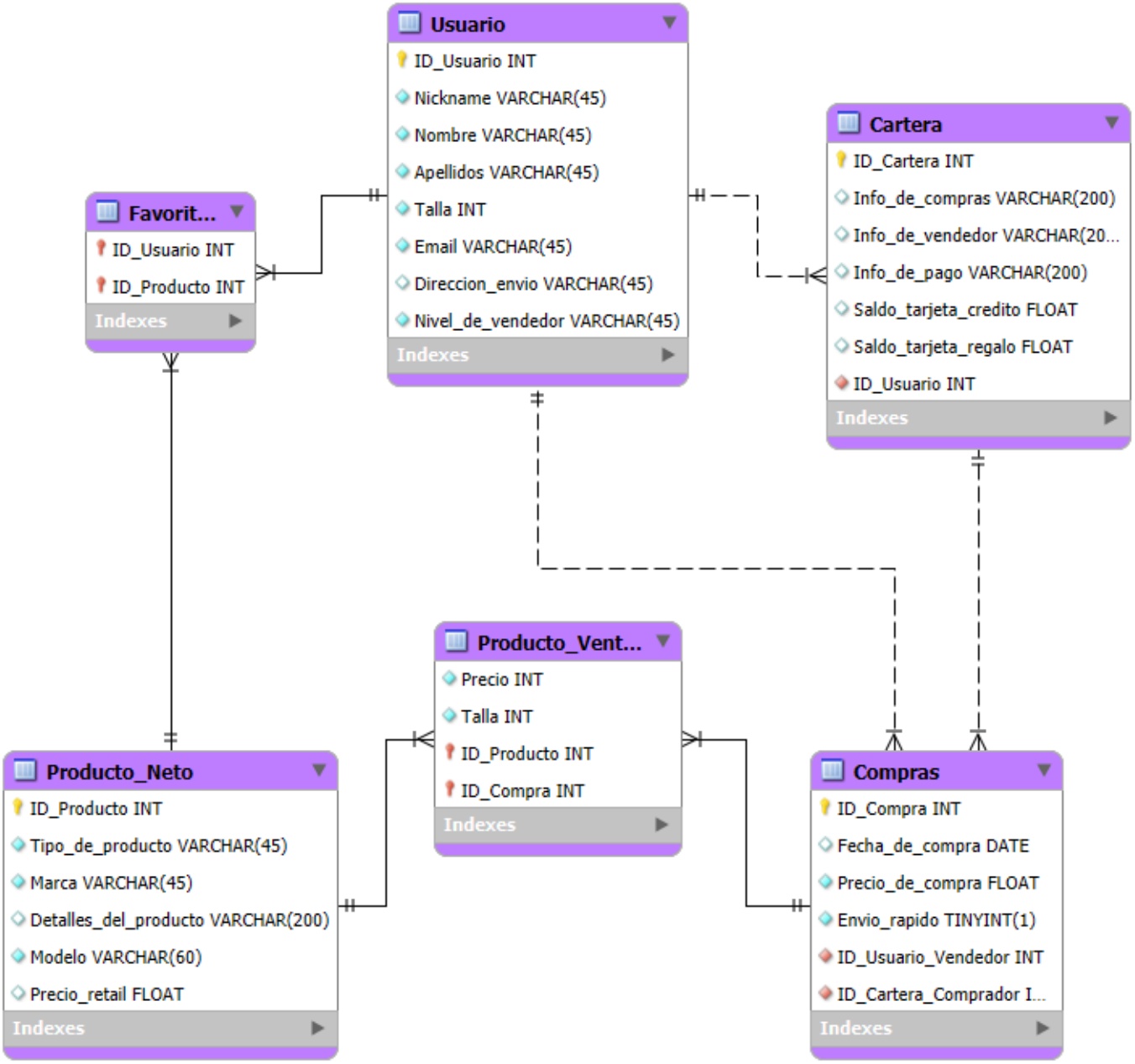
El siguiente modelo se distribuye de la siguiente forma:

- Un usuario puede realizar **compras**, esta entidad tiene almacenados los datos de la compra y de la venta de la misma.
- Un usuario usa la **cartera** que tiene saldo de tarjeta de crédito o/y de regalo, para comprar artículos que están en venta.
- Las compras se realizan sobre un **producto en venta**, es una entidad débil ya que sin las compras y el producto neto no existiría. En el producto venta se tiene la talla y el precio que se le pone al **producto neto** que está en venta. En el producto neto se almacenan las características del producto.
- Un usuario puede añadir a una lista de **favoritos** los productos netos. Es una N:M en la que se almacenan la ID del producto en favoritos y la ID del usuario que la ha añadido a su lista.

3. Modelo Relacional (MySQL Workbench)

Con el modelo delante, podemos aclarar varias cosas del modelo anterior, como la lista de favoritos, la cual tiene la FK de Usuario y de Producto Neto.

También podemos ver las compras, que tienen la ID del Usuario vendedor y la ID de la Cartera del comprador y el producto el venta, que tiene incluida la ID del producto neto que está puesto en venta y la ID de la compra, la cual almacena datos como el usuario que lo compra y el usuario que lo vende.



4. Esquema y carga masiva de datos (Mockaroo)

He vuelto a hacer el esquema y la carga masiva desde 0, ya que tenía que volver a hacer la carga masiva y habían cosas del esquema que tenía que cambiar, como algunas claves primarias, nombres confusos y más fallos que he arreglado. Para la carga masiva de datos he usado Mockaroo y para algunos valores he tenido que usar IA, como es el caso de los modelos de ropa, ya que en Mockaroo no habían los que quería. Las tablas Compras y Producto_Venta tienen 1000 registros, mientras que las demas cuentan con 500.
Asi se veria la carga de datos con un registro de cada tabla:

```
INSERT INTO `Usuario` (`Nickname`, `Nombre`, `Apellidos`, `Talla`, `Email`, `Direccion_envio`, `Nivel_de_vendedor`) VALUES ('ndonnel0', 'Adélaïde', 'Donnel', '51', 'bdonnel0@va.gov', '793 Eliot Terrace', '8');

INSERT INTO `Cartera` (`Info_de_compras`, `Info_de_vendedor`, `Info_de_pago`, `Saldo_tarjeta_credito`, `Saldo_tarjeta_regalo`, `ID_Usuario`) VALUES ('Info_de_compras1', 'Info_de_vendedor1', 'Info_de_pago1', '8386', '233', '344');

INSERT INTO Producto_Neto  (`Marca`, `Tipo_de_producto`, `Detalles_del_producto`, `Modelo`, `Precio_retail`) VALUES ('Nike', 'Botines', 'Cierre: Cremallera', 'Nike Air Zoom Mercurial Superfly 9 Elite FG ', '574');

INSERT INTO Compras (ID_Compra, Fecha_de_compra, Precio_de_compra, Envio_rapido, ID_Usuario_Vendedor, ID_Cartera_Comprador) VALUES ('1', '2024-12-09', '544.5', '0', '311', '292');

INSERT INTO Producto_Ventas (ID_Compra, Precio, Talla, ID_Producto) VALUES ('144', '675.47', '27', '39');

INSERT IGNORE INTO `Favoritos` (`ID_Usuario`, `ID_Producto`) VALUES ('329', '234');
```

5. Consultas

Estas son las consultas que he hecho para mi base de datos, las fotos tendrán pocos resultados ya las consultas devuelven muchos:

```
-- 1. Articulos que se venden a mayor precio que su precio de salida
select Modelo, Precio_retail, avg(pv.Precio),
TRUNCATE(AVG(pv.Precio)-Precio_retail,2) as Beneficio
from Producto_Neto pn
inner join Producto_Ventas pv on pn.ID_Producto = pv.ID_Producto
group by pn.Modelo, pn.Precio_retail
having Beneficio > 0
order by Beneficio asc;
```

	Modelo	Precio_retail	avg(pv.Precio)	Beneficio
1	Converse Classic Chuck Tee_6	404	404,25	0,25
2	Yeezy 500 High_8	277	277,3333	0,33
3	New Balance Impact Run Jogger_3	530	531	1
4	Supreme Nylon Pants_4	589	593	4
5	Adidas Predator Accuracy+_2	386	393,3333	7,33
6	Converse Essentials Fleece Joggers_6	222	232	10
7	New Balance Light Pack Jacket_7	595	605	10
8	Adidas Essentials Linear Joggers_8	525	535	10
9	Yeezy Strapback Cap_2	496	507	11

```
-- 2. Articulo/s que mas se añaden a la lista de favoritos
select pn.Modelo , count(f.ID_Producto) as Favoritos
from Favoritos f inner join Producto_Neto pn on f.ID_Producto =
pn.ID_Producto
group by 1
having count(f.ID_Producto) >= all(
select count(f.ID_Producto)
from Favoritos f
inner join Producto_Neto pn on f.ID_Producto = pn.ID_Producto
group by pn.Modelo);
/*Esta seria la misma consulta pero con limit*/
select pn.Modelo , count(f.ID_Producto)
from Favoritos f inner join Producto_Neto pn on f.ID_Producto =
pn.ID_Producto
group by pn.Modelo
order by count(f.ID_Producto) desc
limit 2;
```

	Modelo	Favoritos
1	Reebok Training Jacket_9	5
2	New Balance Accelerate Hoodie_9	5

```sql
-- 3. Tallas mas usadas para cada producto
select pn.Modelo, pv.Talla, count(pv.Talla) as Ventas
from Producto_Neto pn
inner join Producto_Ventas pv on pn.ID_Producto = pv.ID_Producto
group by pn.Modelo, pv.Talla
having count(pv.Talla) = all(
   select max(Talla_count)
   from (
       select count(pv2.Talla) as Talla_count
       from Producto_Ventas pv2
       inner join Producto_Neto pn2 on pn2.ID_Producto = pv2.ID_Producto
       where pn2.Modelo = pn.Modelo
       group by pv2.Talla
   ) as subconsulta
)
order by 3 desc;
```

	Modelo	Talla	Ventas
1	Yeezy Season 8 Cargo Pants_1	41	3
2	Nike Sportswear Windrunner	37	2
3	Nike Heritage 86 Cap	39	2
4	Yeezy Strapback Cap	37	2
5	Nike Sportswear Windrunner_1	47	2
6	New Balance Sport Cap_1	26	2

```sql
-- 4. Nombre del usuario que mas ventas ha hecho
select c.ID_Usuario_Vendedor , u.Nombre ,count(ID_Usuario_Vendedor) as
Ventas_Realizadas
from Compras c inner join Usuario u on c.ID_Usuario_Vendedor =
u.ID_Usuario
group by c.ID_Usuario_Vendedor
having count(ID_Usuario_Vendedor) >= all(
select count(ID_Usuario_Vendedor)
from Compras c
inner join Usuario u on c.ID_Usuario_Vendedor = u.ID_Usuario
group by c.ID_Usuario_Vendedor);
```

	ID_Usuario_Vendedor	Nombre	Ventas_Realizadas
1	36	Yè	7

```sql
-- 5. Usuarios que han comprado productos con tallas diferentes a la
suya
SELECT u.ID_Usuario ,u.Talla as Talla_Usuario, pv.Talla as Talla_Comprada
from Usuario u
inner join Cartera c on u.ID_Usuario = c.ID_Usuario
inner join Compras cp on cp.ID_Cartera_Comprador = c.ID_Cartera
inner join Producto_Ventas pv on pv.ID_Compra = cp.ID_Compra
where u.Talla != pv.Talla;
```

	ID_Usuario	Talla_Usuario	Talla_Comprada
1	2	40	39
2	4	48	37
3	4	48	28
4	4	48	50

6. Vistas

```
-- 1. Vista masVentas
create view masVentas as
select c.ID_Usuario_Vendedor , u.Nombre ,count(ID_Usuario_Vendedor)
as Ventas_Realizadas
from Compras c inner join Usuario u on c.ID_Usuario_Vendedor =
u.ID_Usuario
group by c.ID_Usuario_Vendedor
having count(ID_Usuario_Vendedor) >= all(
select count(ID_Usuario_Vendedor)
from Compras c
inner join Usuario u on c.ID_Usuario_Vendedor = u.ID_Usuario
group by c.ID_Usuario_Vendedor);
```

	123 ID_Usuario_Vendedor	A-Z Nombre	123 Ventas_Realizadas
1	36	Yè	7

```
-- 2. Vista Beneficios
create view Beneficios as
select Modelo, Precio_retail, avg(pv.Precio),
TRUNCATE(AVG(pv.Precio)-Precio_retail,2) as Beneficio
from Producto_Neto pn
inner join Producto_Ventas pv on pn.ID_Producto = pv.ID_Producto
group by pn.Modelo, pn.Precio_retail
having Beneficio > 0
order by Beneficio asc;
```

	A-Z Modelo	123 Precio_retail	123 avg(pv.Precio)	123 Beneficio
1	Converse Classic Chuck Tee_6	404	404,25	0,25
2	Yeezy 500 High_8	277	277,3333	0,33
3	New Balance Impact Run Jogger_3	530	531	1
4	Supreme Nylon Pants_4	589	593	4
5	Adidas Predator Accuracy+_2	386	393,3333	7,33
6	Converse Essentials Fleece Joggers_6	222	232	10
7	New Balance Light Pack Jacket_7	595	605	10
8	Adidas Essentials Linear Joggers_8	525	535	10
9	Yeezy Strapback Cap_2	496	507	11
10	Supreme Nylon Pants_3	455	468,5	13,5
11	Yeezy Season 7 T-Shirt_3	455	472,5	17,5
12	Reebok Classic Full Zip Hoodie_1	371	388,75	17,75
13	Converse Zip Up Hoodie_9	371	390	19
14	Adidas Originals Cap_6	741	764	23
15	Yeezy Strapback Cap_7	380	405,3333	25,33
16	Nike Dri-FIT Legend_3	731	763	32
17	Supreme Box Logo Tee_3	339	375	36
18	Adidas Originals Cap_4	211	247	36

7. Funciones

```
-- 1. Compras realizadas por la id seleccionada
delimiter //
create function comprasRealizadas(p_usuario_id int)
returns int
deterministic
begin
   declare salida int default 0;
/*Consulta*/
   select count(*) into salida
   from Compras cp
   inner join Cartera c on cp.ID_Cartera_Comprador = c.ID_Cartera
   inner join Usuario u on u.ID_Usuario = c.ID_Usuario
   where u.ID_Usuario = p_usuario_id;
/*Return*/
   return salida;
end //
delimiter ;
/*Select*/
select comprasRealizadas(4);
```

	123 comprasRealizadas(4)
1	3

```
-- 2. Saldo total de un usuario
delimiter //
create function saldoTotal(p_usuario_id int)
returns decimal(30,2)
deterministic
begin
      declare salida decimal default 0;
/*Consulta*/
      select sum(c.Saldo_tarjeta_credito)+sum(c.Saldo_tarjeta_regalo) as
      Saldo_Total into salida
      from Usuario u
      inner join Cartera c on c.ID_Usuario = u.ID_Usuario
      where u.ID_Usuario = p_usuario_id;
/*Return*/
      return salida;
end //
delimiter ;
/*Select*/
select saldoTotal(2);
```

	123 saldoTotal(2)
1	25.775

8. Procedimientos

```
-- 1. Saldo total de un usuario mejorado (Usa la funcion saldoTotal)
delimiter //
create procedure consultarSaldoUsuario(in p_usuario_id int)
begin
   declare p_saldo decimal(30,2);
   declare p_saldo_msg varchar(255);
   declare p_usuario_existe int;
   -- Verifica si el usuario existe
   select count(*) into p_usuario_existe from Usuario where ID_Usuario =
p_usuario_id;
   if p_usuario_existe = 0 then
       -- Si el usuario no existe, mostrar un mensaje de error
       select 'Error: El usuario no existe en la base de datos' as mensaje;
   else
       -- Obtiene el saldo total del usuario
       select sum(c.Saldo_tarjeta_credito) + sum(c.Saldo_tarjeta_regalo)
       into p_saldo
       from Cartera c
       where c.ID_Usuario = p_usuario_id;
       -- Resultado del saldo total medido con nivel
       if p_saldo > 1000 then
           set p_saldo_msg = concat('Saldo total: ', p_saldo, '€. Nivel: Alto');
       elseif p_saldo between 500 and 1000 then
           set p_saldo_msg = concat('Saldo total: ', p_saldo, '€. Nivel: Medio');
       else
           set p_saldo_msg = concat('Saldo total: ', p_saldo, '€. Nivel: Bajo');
       end if;
       -- Mostrar resultado
       select p_saldo_msg as saldo;
   end if;
end //
delimiter ;
-- Llamada a el procedimiento
call consultarSaldoUsuario(3);
```

	A-Z saldo
1	Saldo total: 9607.00€. Nivel: Alto

```
-- 2. Agregar producto a favoritos de un usuario
delimiter //
create procedure agregarAFavoritos(in p_id_usuario int, in p_id_producto int)
begin
   declare p_usuario_existe int;
   declare p_producto_existe int;
   declare p_favorito_existe int;
   declare p_modelo_producto varchar(60);
   -- Verifica si el usuario existe
   select count(*) into p_usuario_existe from Usuario where ID_Usuario =
p_id_usuario;
   -- Verifica si el producto existe
   select count(*) into p_producto_existe from Producto_Neto where ID_Producto =
p_id_producto;
   -- Obtiene modelo del producto si existe
   if p_producto_existe > 0 then
       select Modelo into p_modelo_producto from Producto_Neto where ID_Producto =
p_id_producto;
   end if;
   -- Verifica si el producto ya está en favoritos
   select count(*) into p_favorito_existe from Favoritos where ID_Usuario =
p_id_usuario and ID_Producto = p_id_producto;
      -- Mensajes de error en caso de que algo no exista
   if p_usuario_existe = 0 then
       select 'Error: El usuario no existe' as mensaje;
   elseif p_producto_existe = 0 then
       select 'Error: El producto no existe' as mensaje;
   elseif p_favorito_existe > 0 then
       select 'Error: El producto ya está en la lista de favoritos' as mensaje;
   else
       -- Inserta en la tabla de favoritos
       insert into Favoritos (ID_Usuario, ID_Producto) values (p_id_usuario,
p_id_producto);
       -- Muestra el modelo del producto agregado
       select concat('Producto "', p_modelo_producto, '" agregado a favoritos') as
mensaje;
   end if;
end //
delimiter ;
-- Llamada a el procedimiento
call agregarAFavoritos(3, 12);
```

	mensaje
1	Error: El producto ya está en la lista de favoritos

	mensaje
1	Producto "Puma Future Z 1.4 " agregado a favoritos

```
-- 3. Eliminar favorito
delimiter //
create procedure eliminarDeFavoritos(in p_id_usuario int, in p_id_producto int)
begin
   declare p_usuario_existe int;
   declare p_producto_existe int;
   declare p_favorito_existe int;
   declare p_modelo_producto varchar(60);
   -- Verifica si el usuario existe
   select count(*) into p_usuario_existe from Usuario where ID_Usuario =
p_id_usuario;
   -- Verifica si el producto existe
   select count(*) into p_producto_existe from Producto_Neto where ID_Producto =
p_id_producto;
   -- Obtiene modelo del producto si existe
   if p_producto_existe > 0 then
       select Modelo into p_modelo_producto from Producto_Neto where ID_Producto =
p_id_producto;
   end if;
   -- Verifica si el producto está en favoritos
   select count(*) into p_favorito_existe from Favoritos where ID_Usuario =
p_id_usuario and ID_Producto = p_id_producto;
   -- Mensajes de error en caso de que algo no exista o no esté en favoritos
   if p_usuario_existe = 0 then
       select 'Error: El usuario no existe' as mensaje;
   elseif p_producto_existe = 0 then
       select 'Error: El producto no existe' as mensaje;
   elseif p_favorito_existe = 0 then
       select 'Error: El producto no está en la lista de favoritos' as mensaje;
   else
       -- Elimina de la tabla de favoritos
       delete from Favoritos where ID_Usuario = p_id_usuario and ID_Producto =
p_id_producto;
       -- Muestra el modelo del producto eliminado
       select concat('Producto "', p_modelo_producto, '" eliminado de favoritos')
as mensaje;
   end if;
end //
delimiter ;
```

	mensaje
1	Error: El producto no está en la lista de favoritos

	mensaje
1	Producto "Puma Future Z 1.4 " eliminado de favoritos

9. Triggers

```
-- 1. Verifica que la talla de un ProductoVenta cuando se inserta esta dentro de rangos normales
delimiter //
create trigger beforeProductoVentasInsert
before insert on Producto_Ventas
for each row
begin
    -- Verifica si la talla está fuera del rango permitido (menor que 25 o mayor que 60)
    if new.Talla < 25 or new.Talla > 60 then
        signal sqlstate '45000'
        set message_text = 'Error: La talla del producto debe estar entre 25 y 60';
    end if;
end //
delimiter ;
-- Insertamos un dato en Producto_Vetnas para activar el trigger
INSERT INTO Producto_Ventas (Precio, Talla, ID_Producto, ID_Compra)
VALUES (100.0, 70, 1, 1);
```

SQL Error [1644] [45000]: Error: La talla del producto debe estar entre 25 y 60

```
-- 2. Calcula si el saldo es suficiente para una compra
delimiter //
create trigger beforeCompraInsert
before insert on Compras
for each row
begin
    declare p_saldo_total float;
    -- Obtiene el saldo total del comprador antes de realizar la compra
    select sum(c.Saldo_tarjeta_credito) + sum(c.Saldo_tarjeta_regalo)
    into p_saldo_total
    from Cartera c
    where c.ID_Cartera = new.ID_Cartera_Comprador;
    -- Verifica si el saldo es suficiente para la compra
    if p_saldo_total is null or p_saldo_total < new.Precio_de_compra then
        signal sqlstate '45000' -- Error personalizado
        set message_text = 'Error: Saldo insuficiente para realizar la compra';
    end if;
end //
delimiter ;
-- Insertamos un dato para compras para activar el trigger
insert into Compras (Fecha_de_compra, Precio_de_compra, Envio_rapido,
ID_Usuario_Vendedor, ID_Cartera_Comprador)
values (CURDATE(), 1500000.00, 1, 1, 1);
```

SQL Error [1644] [45000]: Error: Saldo insuficiente para realizar la compra

10. Conclusión

Con este proyecto he aprendido como funciona AWS y los servidores y bases de datos que funcionan en la nube. También me ha servido para repasar Procedimientos, Funciones y Triggers. He visto que puede haber alguna mejora frente al funcionamiento de la tabla Compras y su relación con Producto Ventas.