CODERHOUSE

Proyecto Final - Curso SQL

Base de datos relacional para "Poly Clothes" y" Marle Indumentaria"

Alumno: Martinez Ledo, Ignacio

Comisión: 43420

Profesor: Gabriel Almiñana

Tutor: José Ignacio López Saez

Índice

Introducción:

En este documento detallaré el proceso de creación de una base de datos relacional, aplicando todo lo aprendido durante el curso. Esto incluye la implementación de todo tipo de sentencias SQL, y de los subtipos de lenguajes que lo componen, tales como DML, DCL, DDL, y TCL. Además, se implementarán Stored Procedures, Funciones, Triggers, Vistas, tablas de tipo log/bitácora y creación de usuarios. Dicho documento hablará sobre dos negocios de ropa masculina y femenina.

Modelo de negocio:

La temática que se me ocurrió es muy interesante para realizar, sería un gran desafío y que me serviría mucho es que tengo dos emprendimientos de ropa llamado Poly Clothes y Marle Indumentaria. Este primer proyecto lleva dos años y este último lleva 6 meses. Uno es indumentaria masculina y el otro femenina. En donde debemos tener contacto con proveedores, elegir los productos, control de stock, realización de facturas, etc. Tenemos distintos canales de venta como puede ser MercadoLibre, nuestro sitio web y por último nuestro Instagram. No tenemos un local al público es decir todos nuestros productos se venden de manera online. Para que se den una idea del tamaño del negocio tenemos aproximadamente unas 600 ventas por mes, un total de 20 por día. El 50% de ellas es a través de la plataforma de mercado libre, 40% a través de nuestro sitio web y un 10% es por chat de Instagram pero no olvidemos que el tráfico hacia nuestro sitio web es gracias a Instagram.

Objetivo:

Crear una base de datos relacional utilizando MySQL Workbench, que sea completamente funcional para el usuario final.

Se buscará desarrollar una solución a una problemática a nuestro de negocio de indumentaria completando todas las necesidades que un

cliente puede necesitar, brindándole diferentes herramientas que faciliten las transacciones, el procesamiento y la explotación de datos dentro de la base de datos

- Las Vistas tendrán como objetivo primordial presentar distintos datos como el tipo de envío que un cliente realizó, ver las facturas detalladas de cada cliente, ver controles de stock bajo es decir productos que le queden poco stock para tomar medidas sobre dicha información y también una vista para cuando el producto haya quedado directamente sin stock, en la cual debemos dar de baja en nuestros canales de venta para que no surjan problemas.

- Los Stored Procedures que nos van a brindar la inserción de nuevos datos.

- Los Triggers que nos servirán para las modificaciones del negocio, nos brindarán un panorama y monitoreo de la información de la base de datos.

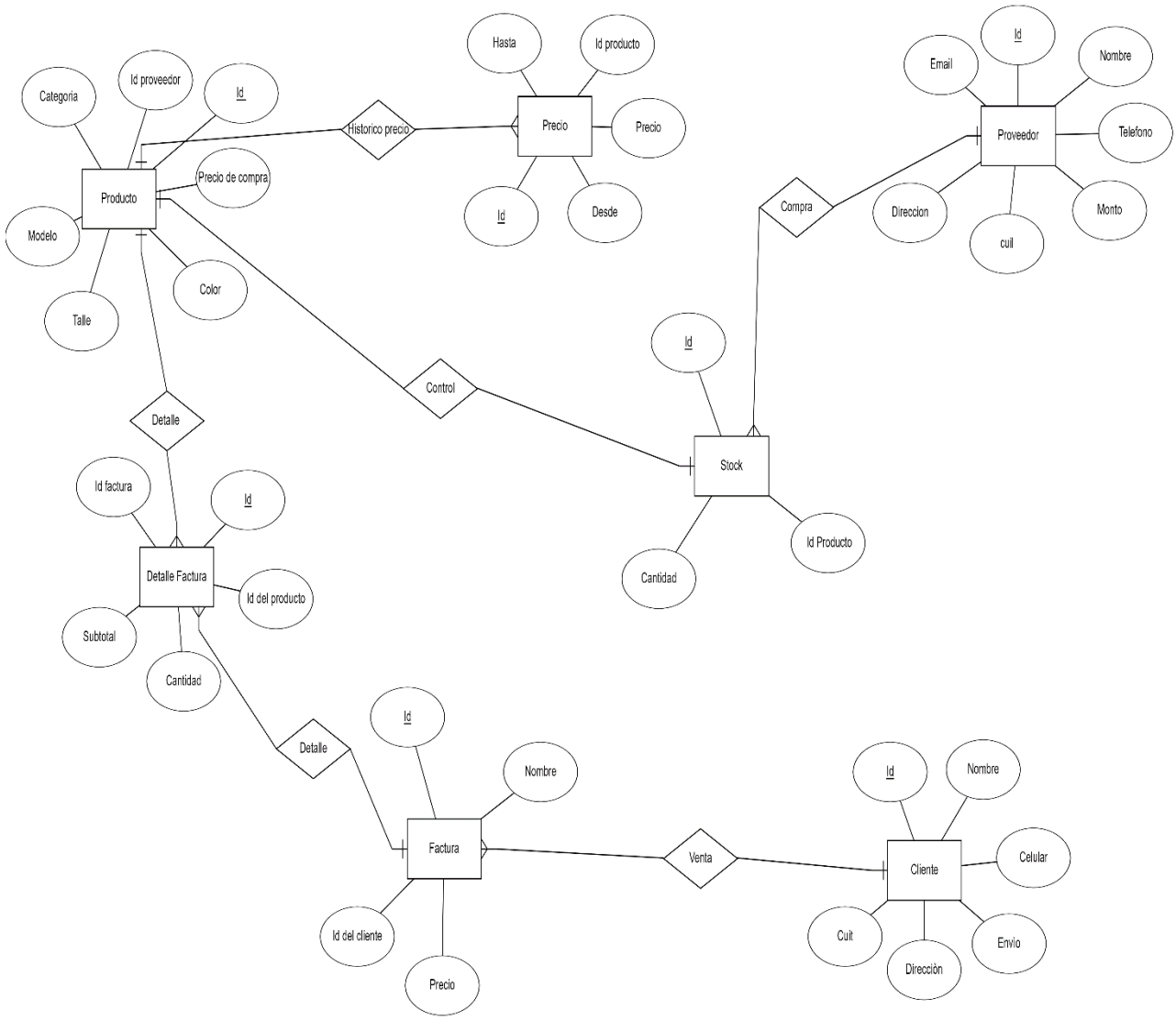
- Las Funciones por supuesto de clave importancia para poder realizar nuestras operaciones, formulas claves para nuestro negocio y de esta manera facilitar la obtención de datos.

Situación problemática:

Será brindar una base de datos de nuestro negocio de indumentaria para facilitar el funcionamiento de la empresa. Esta brindara una gran cantidad de datos relevantes como todos nuestros productos, nuestros clientes, proveedores, la realización de las facturas y también en detalle cada una. Luego un control de stock que es clave para nuestra empresa para no vender productos que no tenemos. Por último, los precios y control de tal ya que sufrimos cotidianamente a una alta inflación, debemos estar atentos para no perder dinero en cada venta.

Para comenzar explicando un poco más nuestro negocio, les mostraremos el diagrama de entidad- relación.

Diagrama entidad relación:



Listado de tablas:

TABLA	CAMPO ABREVIADO	NOMBRE DEL CAMPO COMPLETO	CLAVE PRIMARIA (PK)	CLAVE FORANEA (FK)	TIPO DE DATOS
PRODUCTO	Id_producto	Identificador del producto	PK		INT
	Color	Color del producto			VARCHAR (50)
	Talle	Talle del producto			VARCHAR (20)
	Modelo	Modelo del producto			VARCHAR(100)
	Categoria	Categoria del producto			VARCHAR (50)
	Id_proveedor	Identificador del proveedor		FK	INT
	Precio_compra	Precio de la compra del producto			DECIMAL (10,2)
PRECIO	Id_precio	Identificar de precio	PK		INT
	Precio_compra	Precio de compra			DECIMAL (10,2)
	Id_producto	Identificador de producto		FK	INT
	Desde	Fecha desde que comenzo este precio			DATE
	Hasta	Fecha que finalizo este precio			DATE
STOCK	Id_stock	Identificador de stock	PK		INT
	cantidad	Cantidad del producto			INT
	Id_producto	Identificador de producto		FK	INT
PROVEEDOR	Id_proveedor	Identificador del proveedor	PK		INT
	Nombre	Nombre del proveedor			VARCHAR(100)
	Monto	Monto de la compra al proveedor			DECIMAL (10,2)
	Email	Email del proveedor			VARCHAR(100)
	Telefono	Telefono del proveedor			VARCHAR (20)
	Cuil	Cuil del proveedor			VARCHAR (20)
	Direccion	Direccion del proveedor			VARCHAR(100)
DETALLE DE FACTURA	Id_detalle	Identificado de detalle de factura	PK		INT
	Id_producto	Identificador de producto		FK	INT
	Cantidad	Cantidad de productos comprados			INT
	Subtotal	Subtotal de la factura			DECIMAL (10,2)
	Id_factura	Identificador de factura		FK	INT
FACTURA	Id_factura	Identificador de factura	PK		INT
	Nombre	Nombre del cliente			VARCHAR(100)
	Precio_compra	Precio de la factura			DECIMAL (10,2)
	Id_cliente	Identificador del cliente		FK	INT
CLIENTE	Id_cliente	Identificador del cliente	PK		INT
	Nombre	Nombre del Cliente			VARCHAR(100)
	Celular	Celular del cliente			VARCHAR (20)
	Envio	Metodo de envio del cliente			VARCHAR (100)
	Direccion	Direccion de envio			VARCHAR (100)
	Cuit	Cuit del cliente			VARCHAR (20)

Listado de vistas:

1- Vista de Datos de Clientes con Envío Estándar: Esta vista mostrará solo los datos de los clientes que han seleccionado el envío "Estándar". Lo compone la tabla clientes.

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `clientesconenvioestandar` AS
    SELECT
        `clientes`.`Id_cliente` AS `Id_cliente`,
        `clientes`.`Nombre` AS `Nombre`,
        `clientes`.`Celular` AS `Celular`,
        `clientes`.`Envio` AS `Envio`,
        `clientes`.`Direccion` AS `Direccion`,
        `clientes`.`Cuit` AS `Cuit`
    FROM
        `clientes`
    WHERE
        (`clientes`.`Envio` = 'Estándar')
```

2- Vista de Facturas Detalladas: Esta vista te mostrará los detalles completos de cada factura, incluyendo el nombre y los datos del cliente asociado a cada factura. Se compone de la tabla facturas.

```sql
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `facturasdetalladas` AS
    SELECT
        `f`.`Id_factura` AS `Id_factura`,
        `f`.`Nombre` AS `Nombre_Factura`,
        `f`.`Precio_compra` AS `Total_Factura`,
        `c`.`Nombre` AS `Nombre_Cliente`,
        `c`.`Celular` AS `Celular`,
        `c`.`Envio` AS `Envio`,
        `c`.`Direccion` AS `Direccion`
    FROM
        (`facturas` `f`
        JOIN `clientes` `c` ON ((`f`.`Id_cliente` = `c`.`Id_cliente`)))
```

3- Vista de Productos con Stock Bajo: Esta vista te mostrará los productos con un stock actual inferior a 10 unidades. Se compone de la tabla productos.

```sql
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `productosconstockbajo` AS
    SELECT
        `s`.`Id_producto` AS `Id_producto`,
        `p`.`Color` AS `Color`,
        `p`.`Talle` AS `Talle`,
        `p`.`Modelo` AS `Modelo`,
        `p`.`Categoria` AS `Categoria`,
        `p`.`Id_proveedor` AS `Id_proveedor`,
        `s`.`cantidad` AS `Stock_Actual`
    FROM
        (`productos` `p`
        JOIN `stock` `s` ON ((`p`.`Id_producto` = `s`.`Id_producto`)))
    WHERE
        (`s`.`cantidad` < 10)
```

4- Vista de Productos sin Stock: Esta vista te mostrará los productos que actualmente no tienen ningún registro en la tabla de "Stock", lo que significa que no tienen stock disponible. Se utiliza la tabla productos.

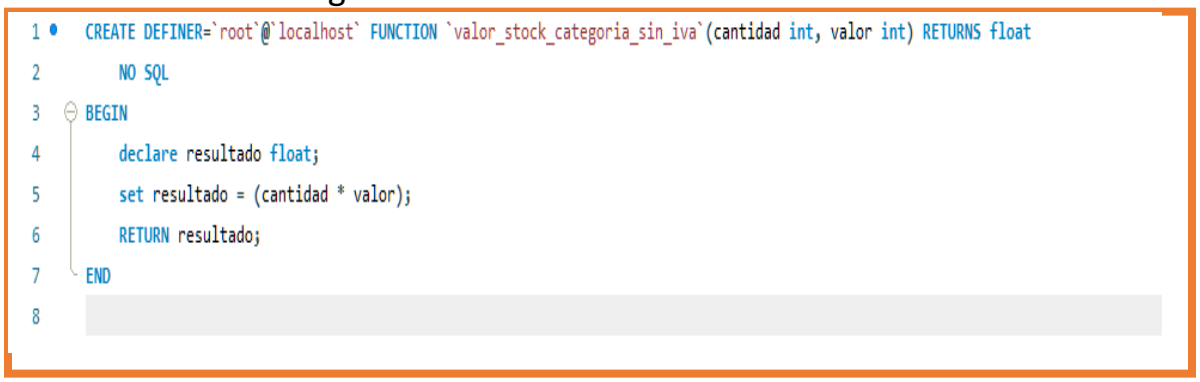
```sql
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `productossinstock` AS
    SELECT
        `p`.`Id_producto` AS `Id_producto`,
        `p`.`Color` AS `Color`,
        `p`.`Talle` AS `Talle`,
        `p`.`Modelo` AS `Modelo`,
        `p`.`Categoria` AS `Categoria`,
        `p`.`Id_proveedor` AS `Id_proveedor`
    FROM
        (`productos` `p`
        LEFT JOIN `stock` `s` ON ((`p`.`Id_producto` = `s`.`Id_producto`)))
    WHERE
        (`s`.`Id_producto` IS NULL)
```

Listado de Funciones:

1- Función para obtener el proveedor de un producto: Esta función es muy útil para saber de tal producto que nos gusto por ejemplo ver quien es el proveedor para realizarle otra compra.

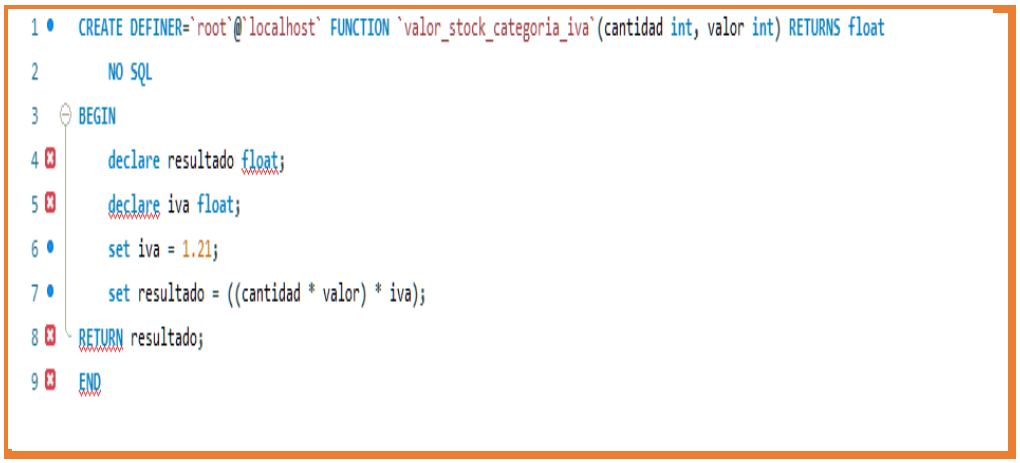
```sql
CREATE DEFINER=`root`@`localhost` FUNCTION `ObtenerProveedorProducto`(IdProducto INT) RETURNS varchar(100) CHARSET utf8mb4
    NO SQL
BEGIN
  DECLARE proveedor VARCHAR(100);
  SELECT Proveedores.Nombre INTO proveedor
  FROM Proveedores
  JOIN Productos ON Proveedores.Id_proveedor = Productos.Id_proveedor
  WHERE Productos.Id_producto = IdProducto;
  RETURN proveedor;
END
```

2-Valor del stock de la categoría: Dicha función la queremos para saber cuanto dinero tenemos en una categoría de producto, es decir el valor del stock de una categoría.



```
CREATE DEFINER=`root`@`localhost` FUNCTION `valor_stock_categoria_sin_iva`(cantidad int, valor int) RETURNS float
    NO SQL
BEGIN
    declare resultado float;
    set resultado = (cantidad * valor);
    RETURN resultado;
END
```

3-Valor del stock de la categoría + iva Dicha función es parecida a la anterior, pero se le suma el porcentaje de iva.



```
CREATE DEFINER=`root`@`localhost` FUNCTION `valor_stock_categoria_iva`(cantidad int, valor int) RETURNS float
    NO SQL
BEGIN
    declare resultado float;
    declare iva float;
    set iva = 1.21;
    set resultado = ((cantidad * valor) * iva);
    RETURN resultado;
    END
```

Listado de Stored Procedures:

1-Stored Procedure para Calcular el Subtotal de una Factura: Este procedimiento almacenado calculará el subtotal de una factura y actualizará el campo "Precio compra" en la tabla "Facturas" con el valor resultante.

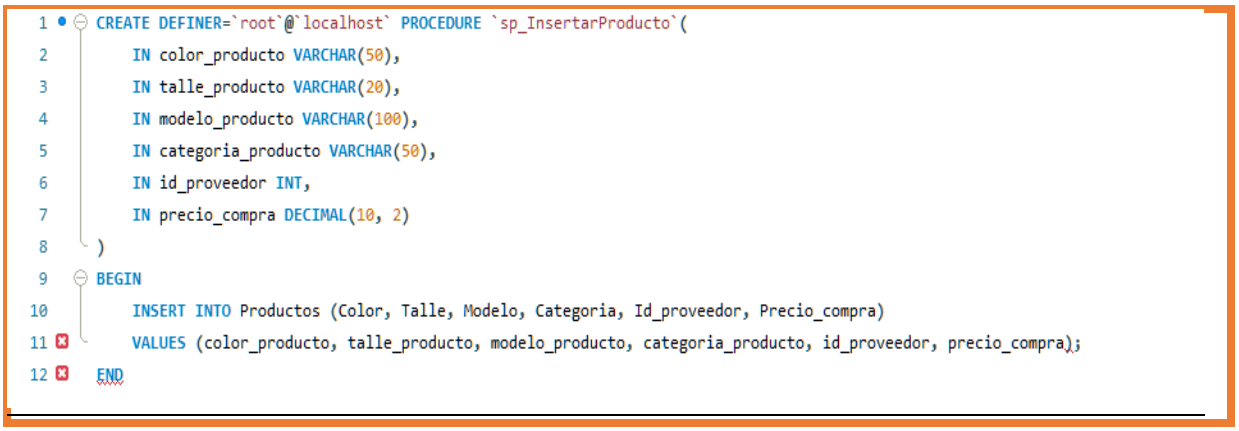
```
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_CalcularSubtotalFactura`(
    IN id_factura INT
)
BEGIN
    DECLARE subtotal DECIMAL(10, 2);
    SELECT SUM(Subtotal) INTO subtotal
    FROM DetalleFactura
    WHERE Id_factura = id_factura;

    UPDATE Facturas
    SET Precio_compra = subtotal
    WHERE Id_factura = id_factura;
END
```

2-Stored Procedure para Insertar un Nuevo Producto: Este procedimiento almacenado te permitirá insertar un nuevo producto en la tabla "Productos".

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_InsertarCliente`(
    IN nombre_cliente VARCHAR(100),
    IN celular_cliente VARCHAR(20),
    IN envio_cliente VARCHAR(100),
    IN direccion_cliente VARCHAR(100),
    IN cuit_cliente VARCHAR(20)
)
BEGIN
    INSERT INTO Clientes (Nombre, Celular, Envio, Direccion, Cuit)
    VALUES (nombre_cliente, celular_cliente, envio_cliente, direccion_cliente, cuit_cliente);
END
```

3-Stored Procedure para Insertar un Cliente: Este procedimiento almacenado te permitirá insertar un nuevo cliente en la tabla "Clientes".



```
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_InsertarProducto`(
    IN color_producto VARCHAR(50),
    IN talle_producto VARCHAR(20),
    IN modelo_producto VARCHAR(100),
    IN categoria_producto VARCHAR(50),
    IN id_proveedor INT,
    IN precio_compra DECIMAL(10, 2)
)
BEGIN
    INSERT INTO Productos (Color, Talle, Modelo, Categoria, Id_proveedor, Precio_compra)
    VALUES (color_producto, talle_producto, modelo_producto, categoria_producto, id_proveedor, precio_compra);
END
```

Listado de Triggers:

1- Trigger "ActualizarStockProducto": Esta trigger se dispara después de insertar un nuevo registro en la tabla "DetalleFactura". Su función es actualizar la cantidad en el stock del producto correspondiente según la cantidad de ese producto vendida en la factura. Básicamente, después de cada venta, reduce la cantidad en stock del producto vendido.

```
DELIMITER //
CREATE TRIGGER trg_ActualizarStockProducto
AFTER INSERT ON DetalleFactura
FOR EACH ROW
BEGIN
    DECLARE stock_actual INT;
    SELECT cantidad INTO stock_actual
    FROM Stock
    WHERE Id_producto = NEW.Id_producto;

    SET stock_actual = stock_actual - NEW.Cantidad;

    UPDATE Stock
    SET cantidad = stock_actual
    WHERE Id_producto = NEW.Id_producto;
END;
//
DELIMITER ;
```

2- Trigger "VerificarStock": Esta trigger se dispara antes de insertar un nuevo registro en la tabla "DetalleFactura". Su objetivo es verificar si hay suficiente stock disponible para el producto que se intenta vender en la factura. Si el stock actual es menor que la cantidad que se intenta vender, la trigger genera un error, lo que impide la inserción y asegura que no se vendan productos sin suficiente stock.

```
20    DELIMITER //
21    CREATE TRIGGER trg_VerificarStock
22    BEFORE INSERT ON DetalleFactura
23    FOR EACH ROW
24    BEGIN
25        DECLARE stock_actual INT;
26        SELECT cantidad INTO stock_actual
27        FROM Stock
28        WHERE Id_producto = NEW.Id_producto;
29
30        IF stock_actual < NEW.Cantidad THEN
31            SIGNAL SQLSTATE '45000'
32            SET MESSAGE_TEXT = 'No hay suficiente stock para este producto.';
33        END IF;
34    END;
35    //
36    DELIMITER ;
```

3- Trigger "EliminarDetalleFactura": Esta trigger se dispara después de eliminar un registro de la tabla "Facturas". Su función es eliminar automáticamente los detalles de factura asociados a la factura que se ha eliminado. Esto asegura que no queden detalles de factura huérfanos en la base de datos después de eliminar una factura.

```
38    DELIMITER //
39    CREATE TRIGGER trg_EliminarDetalleFactura
40    AFTER DELETE ON Facturas
41    FOR EACH ROW
42    BEGIN
43        DELETE FROM DetalleFactura
44        WHERE Id_factura = OLD.Id_factura;
45    END;
46    //
47    DELIMITER ;
```

En resumen, estas triggers tienen los siguientes propósitos:

- Mantener actualizado el stock de productos después de cada venta.
- Prevenir la venta de productos sin suficiente stock.
- Mantener la integridad de la base de datos eliminando automáticamente los detalles de factura cuando se elimina una factura.

Estas triggers contribuyen a automatizar y garantizar la integridad de los datos en tu base de datos al responder automáticamente a eventos específicos.

Herramientas utilizadas:

- Google drive 
- Google Docs 
- My sql 
- Microsoft Excel 
- Mokaro 
- Paint 3d