SANTIAGO ALBERTO RAMIREZ HENAO

ENTREGA FINAL SQL CODERHOUSE

BASE DE DATOS PARA ECOMMERCE RAMBEDJEANS

TIPO DE NEGOCIO

Comercio electrónico de jeans para hombre y mujer, dirigido a un rango de edad entre 30 y 50 años, con alcance internacional.

DEFINICION DEL PROBLEMA

La empresa RAMBED, que se dedica a la comercialización de jeans de hombre, al tener su negocio en la web, ha desarrollado su propio e-commerce. Para este proyecto, necesita una base de datos donde pueda guardar la información relacionada con sus ventas. La finalidad del proyecto es: guardar registros de los clientes, procesar sus compras y pagos, y llevar registro y actualización del despacho del pedido.

OBJETIVOS ENTREGA 1

El objetivo principal es construir una base de datos relacional (SQL) para el manejo, venta y control eficiente. Además, el proyecto debe cumplir con criterios clave como la Integridad de Datos, garantizando 0% de pérdida o corrupción de datos de transacciones, y la Disponibilidad, asegurando que la base de datos esté operativa el 99.9% del tiempo.

OBJETIVOS ENTREGA

Realizar procedimientos y consultas en la base de datos, para revisar su funcionamiento y buen desempeño. Además de insertar datos que nos permitan hacer el trabajo correspondiente con la entrega los cuales son.

- Listado de Vistas más una descripción detallada, su objetivo, y qué tablas las componen.
- Listado de Funciones que incluyan una descripción detallada, el objetivo para la cual fueron creadas y qué datos o tablas manipulan y/o son implementadas.
- Listado de Stored Procedures con una descripción detallada, qué objetivo o beneficio aportan al proyecto, y las tablas que lo componen y/o tablas con las que interactúa.
- Inserción de datos manual y con importación.
- Creación de triggers que ayuden con la automatización de reglas del negocio.

ARQUITECTURA TECNICA

1. Motor de Base de Datos: MySQL WorkBench (Versión 8.0 o superior).

2. Diseño del Modelo: Modelo Relacional Normalizado para asegurar la integridad de los datos y optimizar las consultas. El diseño se basará en el Diagrama Entidad-Relación (DER).

3. Entorno de Despliegue:
Fase de Desarrollo/Pruebas: Servidor local.
Fase de Producción (Definitiva): Se migrará a un proveedor de hosting pago.

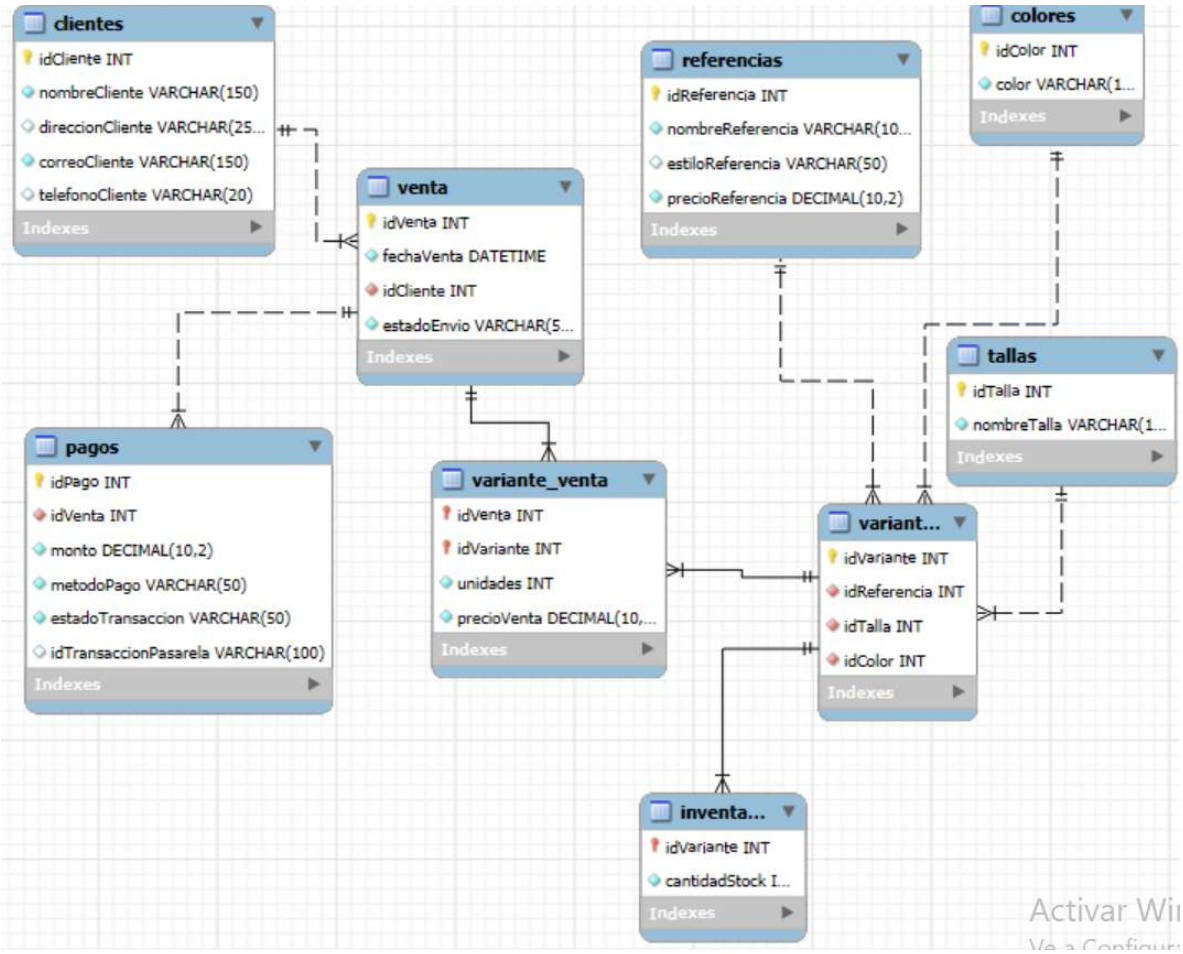
Módulos / Entidades Principales
Clientes: (idCliente, nombreCliente, direccionCliente, correoCliente, telefonoCliente)
Tallas: (idTalla, nombreTalla)
Colores: (idColor, nombreColor)
Referencias(Jeans): (idRerefencia, nombreReferencia, estiloReferencia)
Variantes: (idVariante, IdReferencia, idColor, idTalla)
Inventario: (idVariante, cantidadStock)
Venta: (idVenta, fechaVenta, idCliente, estadoVenta, estadoEnvio)
Variante_Venta: (idVenta, idVariante, unidades, precioVenta)
Pagos: (idPagos, idVenta, monto, metodoPago, estadoTransaccion, idTransaccionPasarela)
Auditoria_inventario: (idAuditoria, idVariante, tipoMovimiento, cantidad, stockAntes, stockDespues, referencia, fechaMovimiento)

TABLAS

TABLAS	CAMPOS	DETALLE CAMPO	PK	FK	TIPO DE DATO
VENTA (PEDIDO)	idVenta	Identificador único de cada pedido.	Sí		INT aoutoincrement
	fechaVenta	Fecha y hora de la transacción.			DATETIME
	idCliente	Cliente que realizó la compra.		Sí	INT
	estadoEnvio	Estado actual de la logística del pedido.			VARCHAR(50)
VARIANTES_VENTAS	idVenta	Venta a la que pertenece el detalle.	Sí	Sí	INT
	idVariante	La variante (Ref/Talla/Color) comprada.	Sí	Sí	INT
	unidades	Cantidad de unidades de esa variante vendidas.			INT
	precioVenta	Precio final de venta de la unidad (para registrar ofertas).			DECIMAL(10, 2)
REFERENCIAS	idReferencia	Identificador único del modelo de jean.	Sí		INT
	nombreReferencia	Nombre comercial del jean (e.g., "Slim Fit").			VARCHAR(100)
	estiloReferencia	Corte o estilo del jean (e.g., "Regular").			VARCHAR(50)
	precioReferencia	Precio base de venta unitario.			DECIMAL(10, 2)
CLIENTES	idCliente	Identificador único del cliente.	Sí		INT
	nombreCliente	Nombre completo del cliente.			VARCHAR(150)
	direccionCliente	Dirección principal de envío.			VARCHAR(255)
	correoCliente	Correo electrónico de contacto.			VARCHAR(150)
	telefonoCliente	Número de contacto.			VARCHAR(20)
TALLAS	idTallas	Identificador único de cada talla.	Sí		INT aoutoincrement
	nombreTallas	Nomenclatura de la talla (e.g., "32", "L").			VARCHAR(10)
COLORES	idColor	Identificador único de cada color.	Sí		INT aoutoincrement
	color	Nombre del color (e.g., "Azul Índigo").			VARCHAR(50)
VARIANTES	idVariante	Identificador único de la combinación Talla + Color + Referencia (SKU).	Sí		INT aoutoincrement
	idReferencia	Referencia de jean a la que pertenece esta variante.		Sí	INT
	idTallas	Talla específica de esta variante.		Sí	INT

	idColor	Color específico de esta variante.		Sí	INT
INVENTARIO	idVariante	Variante de jean con stock. (Ref. a VARIANTES)	Sí	Sí	INT
	cantidadStock	Número de unidades disponibles actualmente.			INT
PAGOS	idPago	Identificador único de cada intento/transacción de pago.	Sí		INT
	idVenta	Pedido al que pertenece el pago.		Sí	INT
	fechaPago	Fecha y hora del intento de pago.			DATETIME
	montoPago	Cantidad de dinero de esta transacción específica.			DECIMAL(10, 2)
	metodoPago	Medio usado (Tarjeta, Transferencia, etc.).			VARCHAR(50)
	estadoTransaccion	Resultado final del intento de pago (Aprobado/Rechazado).			VARCHAR(50)
	idTransaccionPasarela	Código de seguimiento (MERCADOPAGO)			VARCHAR(100)

DIGRAMA DE RELACIONES



INSERCION DE DATOS

Se generan datos para poder hacer búsquedas relacionadas con la base de datos.

Se generan 10 clientes, 7 colores comunes en ventas, 6 tallas comunes en hombre y 6 referencias, la cuales al mezclarlas definimos 24 variantes para asignar cantidades y continuar con las ventas.

Se generan 10 ventas en los meses de octubre, noviembre y diciembre del año 2024, repitiendo clientes. Además se asigna las variantes y las cantidades con el estado de pago al final.

VISTAS

Las siguientes vistas se usan para tener nuevas tablas con información actualizada de manera más global del negocio.

1. INVETARIO ACTUAL

Muestra la variante con su respetiva ref, nombre tallas color y stock actual. Las tablas que la componen son: Inventario, variantes, referencias, tallas y colores.

idVariante	idReferencia	nombreReferencia	nombreTalla	color	cantidadStock
1	2501	Jean Clásico Azul	28	azulOscuro	3
2	2501	Jean Clásico Azul	30	azulOscuro	5
3	2501	Jean Clásico Azul	32	azulOscuro	2
4	2501	Jean Clásico Azul	34	azulOscuro	4
5	2502	Jean Slim Oscuro	30	azulMedio	1
6	2502	Jean Slim Oscuro	32	azulMedio	3
7	2502	Jean Slim Oscuro	34	azulMedio	5
8	2502	Jean Slim Oscuro	36	azulMedio	2
9	2503	Jean Cargo Urbano	32	negro	4
10	2503	Jean Cargo Urbano	34	negro	0
11	2503	Jean Cargo Urbano	36	negro	3
12	2503	Jean Cargo Urbano	40	negro	5
13	2504	Jean Skinny Negro	28	gris	2
14	2504	Jean Skinny Negro	30	gris	4
15	2504	Jean Skinny Negro	32	gris	1
16	2504	Jean Skinny Negro	34	gris	3
17	2505	Jean Regular Stone	30	azulPetro	5
18	2505	Jean Regular Stone	32	azulPetro	2
19	2505	Jean Regular Stone	34	azulPetro	4
20	2505	Jean Regular Stone	36	azulPetro	1
21	2506	Jean Oversize Claro	32	dirty	3
22	2506	Jean Oversize Claro	34	dirty	5
23	2506	Jean Oversize Claro	36	dirty	2
24	2506	Jean Oversize Claro	40	dirty	4

2. VENTAS DETALLE CLIENTES

Muestra al estado de la venta por nombre de cliente y sus estado, para tener mejor panorama de lo envios. Tablas: Ventas y clientes.

idVenta	fechaVenta	idCliente	nombreCliente	estadoEnvio
1	2024-10-05 10:15:00	1023456789	Juan Pérez	enviado
4	2024-11-03 14:20:00	1023456789	Juan Pérez	entregado
7	2024-12-02 12:30:00	1023456789	Juan Pérez	enviado
10	2024-12-27 10:00:00	1023456789	Juan Pérez	entregado
11	2024-10-05 10:15:00	1023456789	Juan Pérez	enviado
14	2024-11-03 14:20:00	1023456789	Juan Pérez	entregado
17	2024-12-02 12:30:00	1023456789	Juan Pérez	enviado
20	2024-12-27 10:00:00	1023456789	Juan Pérez	entregado
2	2024-10-12 16:40:00	1034567890	María Gómez	entregado
5	2024-11-10 09:50:00	1034567890	María Gómez	enviado
8	2024-12-09 15:45:00	1034567890	María Gómez	entregado
12	2024-10-12 16:40:00	1034567890	María Gómez	entregado
15	2024-11-10 09:50:00	1034567890	María Gómez	enviado
18	2024-12-09 15:45:00	1034567890	María Gómez	entregado
3	2024-10-20 11:05:00	1045678901	Carlos Rodríguez	enviado
6	2024-11-22 18:10:00	1045678901	Carlos Rodríguez	entregado
9	2024-12-18 17:25:00	1045678901	Carlos Rodríguez	enviado
13	2024-10-20 11:05:00	1045678901	Carlos Rodríguez	enviado
16	2024-11-22 18:10:00	1045678901	Carlos Rodríguez	entregado
19	2024-12-18 17:25:00	1045678901	Carlos Rodríguez	enviado

3. VENTAS TOTALES

Resume del total de la ventas. Tablas ventas, variante_venta y clientes.

idVenta	fechaVenta	nombreCliente	totalVenta
3	2024-10-20 11:05:00	Carlos Rodríguez	92000
6	2024-11-22 18:10:00	Carlos Rodríguez	110000
9	2024-12-18 17:25:00	Carlos Rodríguez	392000
1	2024-10-05 10:15:00	Juan Pérez	255000
4	2024-11-03 14:20:00	Juan Pérez	276000
7	2024-12-02 12:30:00	Juan Pérez	240000
10	2024-12-27 10:00:00	Juan Pérez	210000
2	2024-10-12 16:40:00	María Gómez	354000
5	2024-11-10 09:50:00	María Gómez	440000
8	2024-12-09 15:45:00	María Gómez	240000

4. PRODUCTOS MAS VENDIDOS

Vista que nos muestra los productos mas vendidos de la tienda. Tablas variante_venta, variantes, referencias.

nombreReferencia	unidadesVendidas
Jean Clásico Azul	5
Jean Slim Oscuro	6
Jean Cargo Urbano	5
Jean Skinny Negro	6
Jean Regular Stone	6

5. VISTA DE LAS VENTAS CREADAS, LUEGO DE ORGANIZAR UN PROCEDURE DE VENTAS

```
CREATE VIEW vista_total_venta AS
SELECT
   v.idVenta,
   SUM(vv.unidades * vv.precioVenta) AS totalVenta
FROM venta v
JOIN variante_venta vv ON v.idVenta = vv.idVenta
GROUP BY v.idVenta;
```

idVenta	totalVenta
1	255000.00
2	354000.00
3	92000.00
4	276000.00
5	440000.00
6	110000.00
7	240000.00
8	240000.00
9	392000.00
10	210000.00

FUNCIONES

Para la base de datos se crean 3 funciones que nos ayudan a revisar el estado actual del negocio. Estas son, ventas totales, stock actual y un stock por referencia. Las funciones no cambian datos solo nos devuelven un valor especifico.

fn_total_venta(id): Esta función recibe por parámetro el id de la tabla variante_venta y nos muestra la suma total del valor de las variantes asociadas a esta venta.

```
DELIMITER $$
CREATE FUNCTION fn_total_venta(p_idVenta INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
   DECLARE total DECIMAL(10,2);

   SELECT SUM(unidades * precioVenta)
   INTO total
   FROM variante_venta
   WHERE idVenta = p_idVenta;

   RETURN IFNULL(total, 0);
END$$

DELIMITER ;

-- ensayamos la funcion por id de la variante_venta
SELECT fn_total_venta(2);
```

fn_stock_varinate (id): Esta función recibe por parámetro el id de la variante y nos muestra en el inventario la cantidad de stock que se tiene por esa variante.

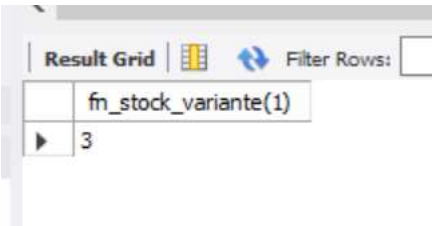
```
CREATE FUNCTION fn_stock_variante(p_idVariante INT)
RETURNS INT
DETERMINISTIC
BEGIN
   DECLARE stock INT;

   SELECT cantidadStock
   INTO stock
   FROM inventario
   WHERE idVariante = p_idVariante;

   RETURN IFNULL(stock, 0);
END$$

DELIMITER ;

SELECT fn_stock_variante(1);
```



fn_total_stock: Esta función nos suma todo el stock disponible, no usa parámetros.

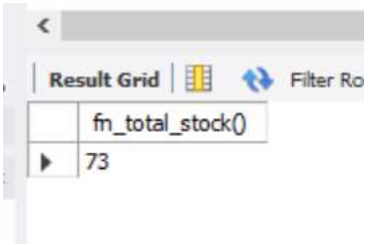
```
BEGIN
   DECLARE totalStock INT;

   SELECT SUM(cantidadStock)
   INTO totalStock
   FROM inventario;

   RETURN IFNULL(totalStock, 0);
END$$

DELIMITER ;

SELECT fn_total_stock();
```



PROCEDIMIENTOS

Los procedimientos son declaraciones de sql que nos ayudan a actualizar, editar o borrar datos de la base de datos de forma mas rápida, ya que quedan guardados para su uso.

Para la base de datos usaremos tres procedimientos, uno para registrar ventas, otro que nos asigna los detalles de la venta y otro para agregar pagos.

sp_registrar_venta(3 paramentros): Nos ayuda a registrar en la tabla de ventas una nueva venta con un cliente ya creado en la base de datos, pasamos datos de fecha, cliente y estado de la venta.

```
DELIMITER $$

CREATE PROCEDURE sp_registrar_venta(

   IN p_fecha DATETIME,

   IN p_idCliente INT,

   IN p_estadoEnvio VARCHAR(50)

)

BEGIN

   INSERT INTO venta (fechaVenta, idCliente, estadoEnvio)

   VALUES (p_fecha, p_idCliente, p_estadoEnvio);

END$$

DELIMITER ;

CALL sp_registrar_venta('2024-11-15', 1090123456, 'enviado');
```

20	2024-12-27 10:00:00	1023456789	entregado
22	2024-11-15 00:00:00	1090123456	enviado
NULL	NULL	NULL	NULL

sp_agregar_detalle_venta(4 parametros): A este procedimiento tomamaos el ID de la venta aterior y le pasamos otros 3 parametros que para asignar a la venta los jeans que se necesitan. Hay que revisar que hay disponible y asiganrlo.

```
DELIMITER $$

CREATE PROCEDURE sp_agregar_detalle_venta(

   IN p_idVenta INT,
```

```
    IN p_idVariante INT,
    IN p_unidades INT,
    IN p_precio DECIMAL(10,2)
)
BEGIN
    INSERT INTO variante_venta (idVenta, idVariante, unidades, precioVenta)
    VALUES (p_idVenta, p_idVariante, p_unidades, p_precio);
END$$
DELIMITER ;

CALL sp_agregar_detalle_venta(22, 17, 2, 98000);
```

9	19	1	98000.00
10	20	2	105000.00
22	17	2	98000.00
NULL	NULL	NULL	NULL

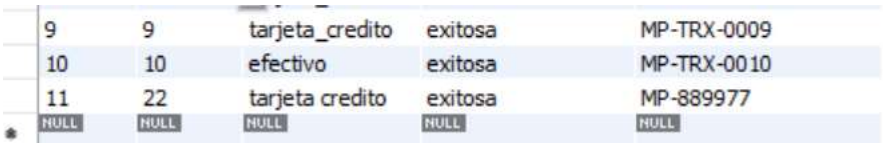
Sp_pregistrar_pago(4 parametros): Este procedimiento nos ayuda a registrar los pago que se hacen a la venta y sus variantes.

```
DELIMITER $$
CREATE PROCEDURE sp_registrar_pago(
    IN p_idVenta INT,
    IN p_metodoPago VARCHAR(50),
    IN p_estado VARCHAR(50),
    IN p_idTransaccion VARCHAR(100)
)
BEGIN
    INSERT INTO pagos (idVenta, metodoPago, estadoTransaccion, idTransaccionPasarela)
    VALUES (p_idVenta, p_metodoPago, p_estado, p_idTransaccion);
```

```
END$$

DELIMITER ;

CALL sp_registrar_pago(22, 'tarjeta credito', 'exitosa', 'MP-889977');
```

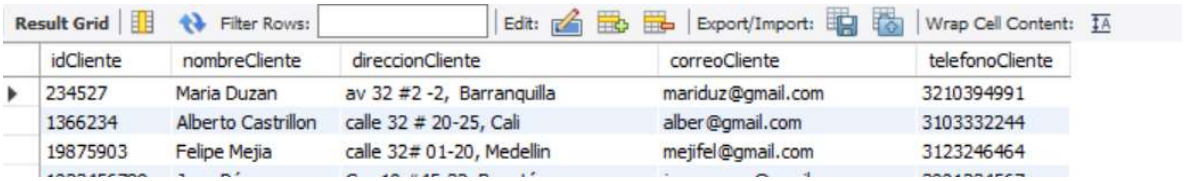


9	9	tarjeta_credito	exitosa	MP-TRX-0009
10	10	efectivo	exitosa	MP-TRX-0010
11	22	tarjeta credito	exitosa	MP-889977
NULL	NULL	NULL	NULL	NULL

INSERTAR DATOS DESDE UN ARCHIVO EXTERNO

Para este proceso se va anexar mas clientes desde un archivo de Excel.

El archivo en formato cvs, se llama ClientesNuevos.cvs. Se usa el wizard de mysqlWorbench para anexarlos importándolos desde la tabla clientes. Este es el resultado.



idCliente	nombreCliente	direccionCliente	correoCliente	telefonoCliente
234527	Maria Duzan	av 32 #2 -2, Barranquilla	mariduz@gmail.com	3210394991
1366234	Alberto Castrillon	calle 32 # 20-25, Cali	alber@gmail.com	3103332244
19875903	Felipe Mejia	calle 32# 01-20, Medellin	mejifel@gmail.com	3123246464

TRIGGERS

Son automatizaciones de eventos para cierta tabla. Solo pueden usarse para estos eventos de DML:

- INSERT
- UPDATE
- DELETE

1. TRIGGER PARA ACTUALIZAR INVENTARIO LUEGO DE SER APROBADA LA TRANSACCION

```
DELIMITER $$

CREATE TRIGGER t_pago_actualiza_venta
after insert on pagos
for each row
 begin
        if new.estadoTransaccion = 'Aprobada' then

    update venta
    set estadoVenta = 'Pagada'
```

```
      where idVenta = new.idVenta;
          end if;
end $$

DELIMITER ;
```

2. TIGGER PARA INVENTARIO NO NEGATIVO

```
DELIMITER $$

CREATE TRIGGER trg_no_stock_negativo
BEFORE UPDATE ON inventario
FOR EACH ROW
BEGIN
  IF NEW.cantidadStock < 0 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'El stock no puede ser negativo';
  END IF;
END$$

DELIMITER ;
```

3. TRIGGER DE AUDITORIA PARA ENTRADA DE STOCK MANUAL

```
DELIMITER $$

CREATE TRIGGER trg_auditoria_update_inventario
AFTER UPDATE ON inventario
FOR EACH ROW
BEGIN
```

```
    -- esto para cuando cambia el stock manual

    IF OLD.cantidadStock <> NEW.cantidadStock THEN

        INSERT INTO auditoria_inventario (
            idVariante,
            tipoMovimiento,
            cantidad,
            stockAntes,
            stockDespues,
            referencia
        )
        VALUES (
            NEW.idVariante,
            'ajuste',
            NEW.cantidadStock - OLD.cantidadStock,
            OLD.cantidadStock,
            NEW.cantidadStock,
            'Ajuste manual de inventario'
        );

    END IF;
END$$

DELIMITER ;
```

MANIPULACION DE DATOS FINAL

Se actualizará el inventario para revisar la tabla auditoria.

Se simularan 2 ventas con sus respectivos pagos para revisar funcionamiento de stock.

1. MANIPULACION DE STOCK MANUAL

Revisamos las vistas actuales de inventario usando una VISTA programada:

SELECT * FROM vista_inventario_actual;

idVariante	idReferencia	nombreReferencia	nombreTalla	color	cantidadStock
1	2501	Jean Clásico Azul	28	azulOscuro	3
2	2501	Jean Clásico Azul	30	azulOscuro	5
3	2501	Jean Clásico Azul	32	azulOscuro	2
4	2501	Jean Clásico Azul	34	azulOscuro	4
5	2502	Jean Slim Oscuro	30	azulMedio	1
6	2502	Jean Slim Oscuro	32	azulMedio	3
7	2502	Jean Slim Oscuro	34	azulMedio	5
8	2502	Jean Slim Oscuro	36	azulMedio	2
9	2503	Jean Cargo Urbano	32	negro	4
10	2503	Jean Cargo Urbano	34	negro	0
11	2503	Jean Cargo Urbano	36	negro	3
12	2503	Jean Cargo Urbano	40	negro	5
13	2504	Jean Skinny Negro	28	gris	2
14	2504	Jean Skinny Negro	30	gris	4
15	2504	Jean Skinny Negro	32	gris	1
16	2504	Jean Skinny Negro	34	gris	3
17	2505	Jean Regular Stone	30	azulPetro	5
18	2505	Jean Regular Stone	32	azulPetro	2
19	2505	Jean Regular Stone	34	azulPetro	4
20	2505	Jean Regular Stone	36	azulPetro	1
21	2506	Jean Oversize Claro	32	dirty	3
22	2506	Jean Oversize Claro	34	dirty	5
23	2506	Jean Oversize Claro	36	dirty	2
24	2506	Jean Oversize Claro	40	dirty	4

Insertamos para variante con idVarainte 10, 5 unidades:

update inventario set cantidadStock = 5 where idVariante = 10;

Revisamos de nuevo inventario que queda actualizado con:

SELECT fn_stock_variante(10);

y la tabla auditoria con datos:

select * from auditoria_inventario;

idAuditoria	idVariante	tipoMovimiento	cantidad	stockAntes	stockDespues	referencia	fechaMovimiento
1	10	ajuste	5	0	5	Ajuste manual de inventario	22/01/2026 16:36

2. REGISTRO DE VENTA, usando dos variantes incluida la 10, para revisar stock.

Para este registro usamos los procedimientos guardados.

Creamos la venta con un id registrado:

CALL sp_registrar_venta('2026-01-22', 1090123456, 'Pendiente');

Asignamos variante a la venta 11 que fue la generada en el paso anterior, asignamos variante 10, 2 und y variante 1, 1und:

CALL sp_agregar_detalle_venta(11, 10, 2, 98000);

CALL sp_agregar_detalle_venta(11, 1, 1, 100000);

Revisamos estados de venta con dos procedimientos

SELECT fn_total_venta(11);

SELECT fn_stock_variante(10);

Revisamos auditoria:

select * from auditoria_inventario;

TABLAS

SELECT * FROM venta_detalle;

idVenta	fechaVenta	idCliente	nombreCliente	estadoEnvio
1	5/10/2024 10:15	1023456789	Juan PÃ©rez	Enviado
4	3/11/2024 14:20	1023456789	Juan PÃ©rez	Enviado
7	2/12/2024 12:30	1023456789	Juan PÃ©rez	Enviado
10	27/12/2024 10:00	1023456789	Juan PÃ©rez	Enviado
2	12/10/2024 16:40	1034567890	MarÃa GÃ³mez	Enviado
5	10/11/2024 9:50	1034567890	MarÃa GÃ³mez	Enviado
8	9/12/2024 15:45	1034567890	MarÃa GÃ³mez	Enviado
3	20/10/2024 11:05	1045678901	Carlos RodrÃ-guez	Enviado
6	22/11/2024 18:10	1045678901	Carlos RodrÃ-guez	Enviado
9	18/12/2024 17:25	1045678901	Carlos RodrÃ-guez	Enviado
11	22/01/2026 0:00	1090123456	Natalia Herrera	Pendiente

SELECT * FROM vista_inventario_actual;

Vemos el cambio de inventario de la variante 1 y 10

idVariante	idReferencia	nombreReferencia	nombreTalla	color	cantidadStock
1	2501	Jean ClÃ¡sico Azul	28	azulOscuro	2
2	2501	Jean ClÃ¡sico Azul	30	azulOscuro	5
3	2501	Jean ClÃ¡sico Azul	32	azulOscuro	2
4	2501	Jean ClÃ¡sico Azul	34	azulOscuro	4
5	2502	Jean Slim Oscuro	30	azulMedio	1
6	2502	Jean Slim Oscuro	32	azulMedio	3
7	2502	Jean Slim Oscuro	34	azulMedio	5
8	2502	Jean Slim Oscuro	36	azulMedio	2
9	2503	Jean Cargo Urbano	32	negro	4
10	2503	Jean Cargo Urbano	34	negro	3
11	2503	Jean Cargo Urbano	36	negro	3
12	2503	Jean Cargo Urbano	40	negro	5

13	2504	Jean Skinny Negro	28	gris	2
14	2504	Jean Skinny Negro	30	gris	4
15	2504	Jean Skinny Negro	32	gris	1
16	2504	Jean Skinny Negro	34	gris	3
17	2505	Jean Regular Stone	30	azulPetro	5
18	2505	Jean Regular Stone	32	azulPetro	2
19	2505	Jean Regular Stone	34	azulPetro	4
20	2505	Jean Regular Stone	36	azulPetro	1
21	2506	Jean Oversize Claro	32	dirty	3
22	2506	Jean Oversize Claro	34	dirty	5
23	2506	Jean Oversize Claro	36	dirty	2
24	2506	Jean Oversize Claro	40	dirty	4

select * from auditoria_inventario;

idAuditoria	idVariante	tipoMovimiento	cantidad	stockAntes	stockDespues	referencia	fechaMovimiento
1	10	ajuste	5	0	5	Ajuste manual de inventario	22/01/2026 16:36
2	10	ajuste	-2	5	3	Ajuste manual de inventario	22/01/2026 16:56
3	10	venta	2	5	3	Venta ID 11	22/01/2026 16:56
4	1	ajuste	-1	3	2	Ajuste manual de inventario	22/01/2026 16:57
5	1	venta	1	3	2	Venta ID 11	22/01/2026 16:57

CONCLUSIONES

- El uso de los procedimientos, vistas, funciones y triggers nos ayudan a establecer una buena estructura de base de datos y la manipulación de estas. Los procedimientos y funciones ayudan que los datos se procesen siempre bajo las mismas reglas, lo que facilita enormemente el **mantenimiento** y la **escalabilidad**. Al automatizar acciones como registros de auditoría o validaciones de stock aseguramos que la **integridad de los datos** no dependa de que el usuario.
- EL flujo del trabajo realizado y las pruebas fueron exitosas, quedando a la espera de la revisión.