**Propotipo Proyecto Final**

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

**José Domingo Olea Sarmiento**

**Juan Eduardo Jattin Ceballos**

**Bayron Paternina Morales**

**Rubén Enrique Baena Navarro**

**Base de datos II**

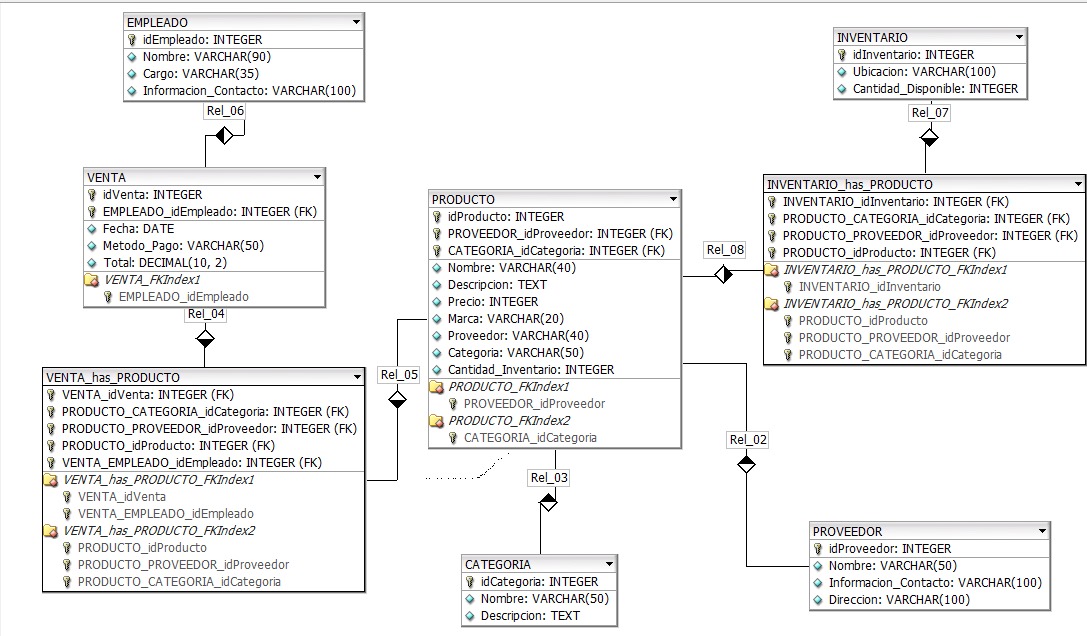
**Universidad de Córdoba**

**Facultad de ingenierías**

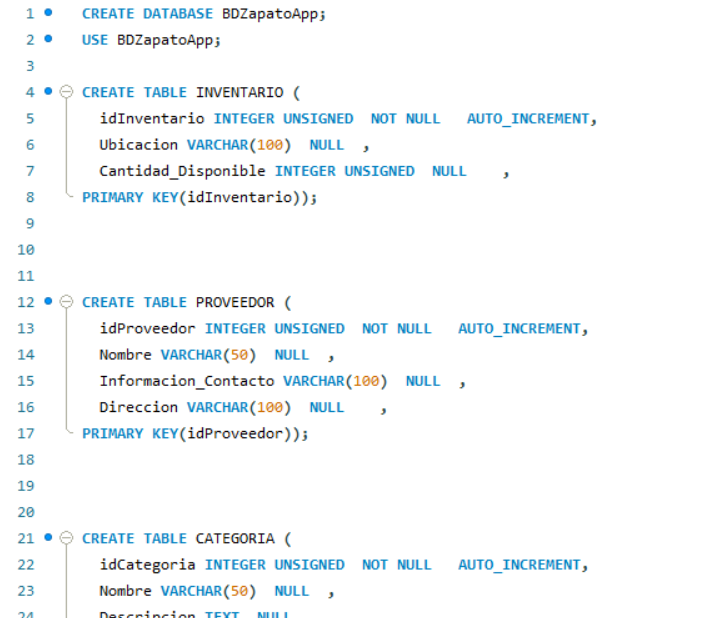
**Programa ingeniería de sistemas**

**2023**

**Modelo Entidad-Relación**



Base de datos creada:



**Tablas:**

**categorías:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**empleados:**



**inventario:**

**Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente**

**productos:**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

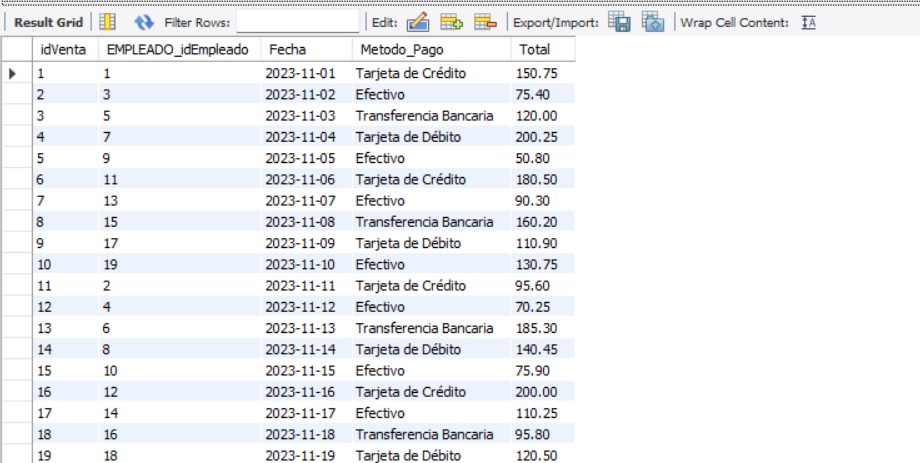
Descripción generada automáticamente**

**Proveedores:**

**Texto

Descripción generada automáticamente**

**Ventas:**

****

**SUBCONSULTAS:**

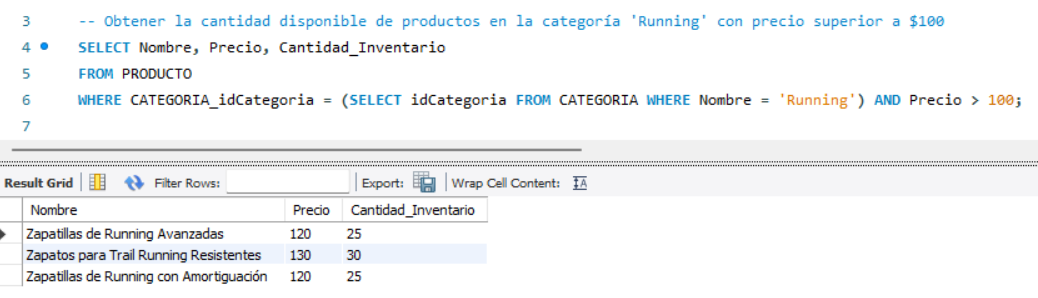
* **Obtener la cantidad disponible de productos en la categoría 'Running' con precio superior a $100**

SELECT Nombre, Precio, Cantidad\_Inventario

FROM PRODUCTO

WHERE CATEGORIA\_idCategoria = (SELECT idCategoria FROM CATEGORIA WHERE Nombre = 'Running') AND Precio > 100;

**Resultado:**

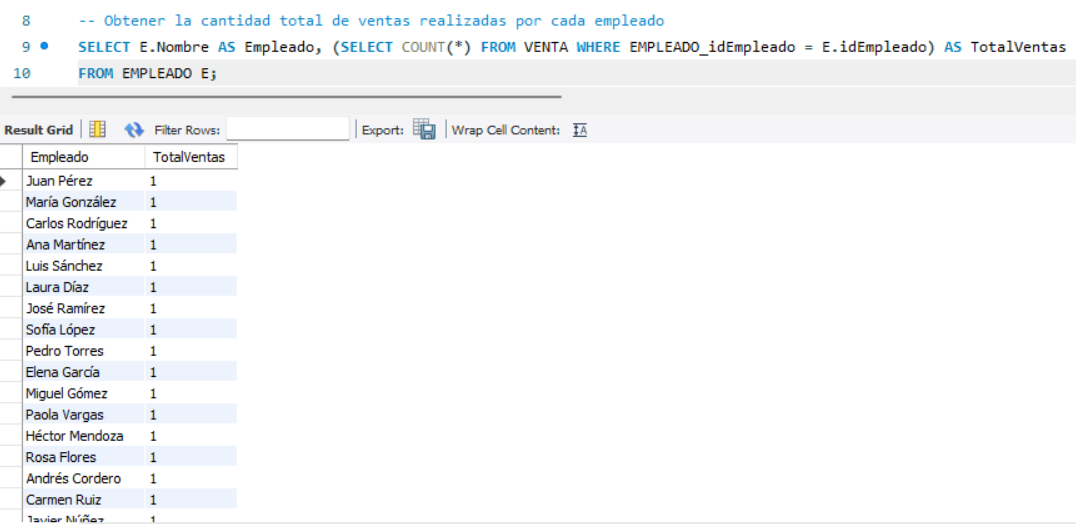


* **Obtener la cantidad total de ventas realizadas por cada empleado**

SELECT E.Nombre AS Empleado, (SELECT COUNT(\*) FROM VENTA WHERE EMPLEADO\_idEmpleado = E.idEmpleado) AS TotalVentas

FROM EMPLEADO E;

**Resultado:**



* **Obtener el nombre del producto y la diferencia entre su precio y el precio promedio de la categoría**

SELECT

Nombre,

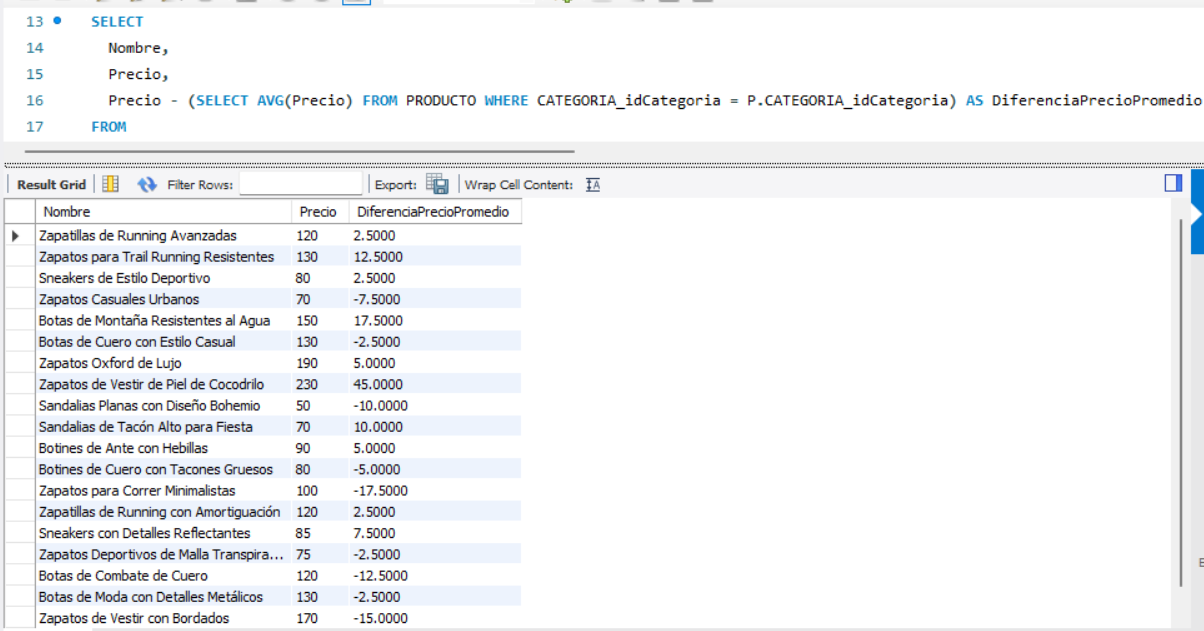
Precio,

Precio - (SELECT AVG(Precio) FROM PRODUCTO WHERE CATEGORIA\_idCategoria = P.CATEGORIA\_idCategoria) AS DiferenciaPrecioPromedio

FROM

PRODUCTO P;

**Resultado:**



**PROCEDIMIENTOS ALMACENADOS:**

* **Procedimiento para insertar un nuevo empleado**

DELIMITER //

CREATE PROCEDURE InsertarEmpleado(IN p\_Nombre VARCHAR(90), IN p\_Cargo VARCHAR(35), IN p\_Info\_Contacto VARCHAR(100))

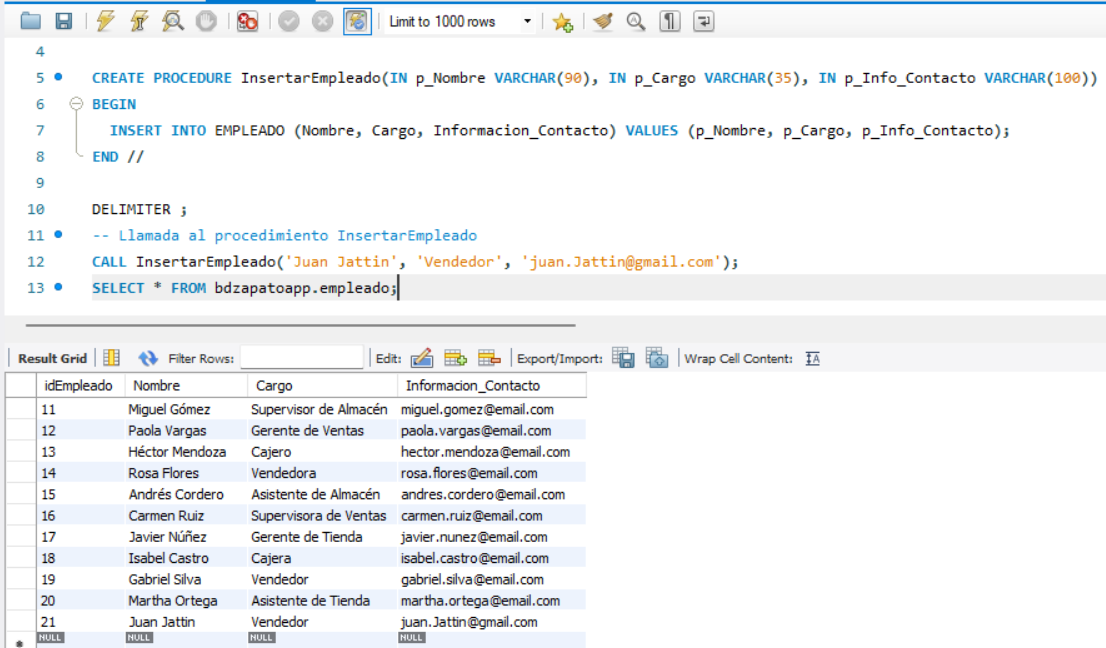
BEGIN

INSERT INTO EMPLEADO (Nombre, Cargo, Informacion\_Contacto) VALUES (p\_Nombre, p\_Cargo, p\_Info\_Contacto);

END //

DELIMITER ;

**Resultado:**

****

* **Procedimiento para actualizar el precio de productos de una categoría**

DELIMITER //

CREATE PROCEDURE ActualizarPrecioCategoria(IN p\_Categoria VARCHAR(50), IN p\_NuevoPrecio DECIMAL(10, 2))

BEGIN

UPDATE PRODUCTO

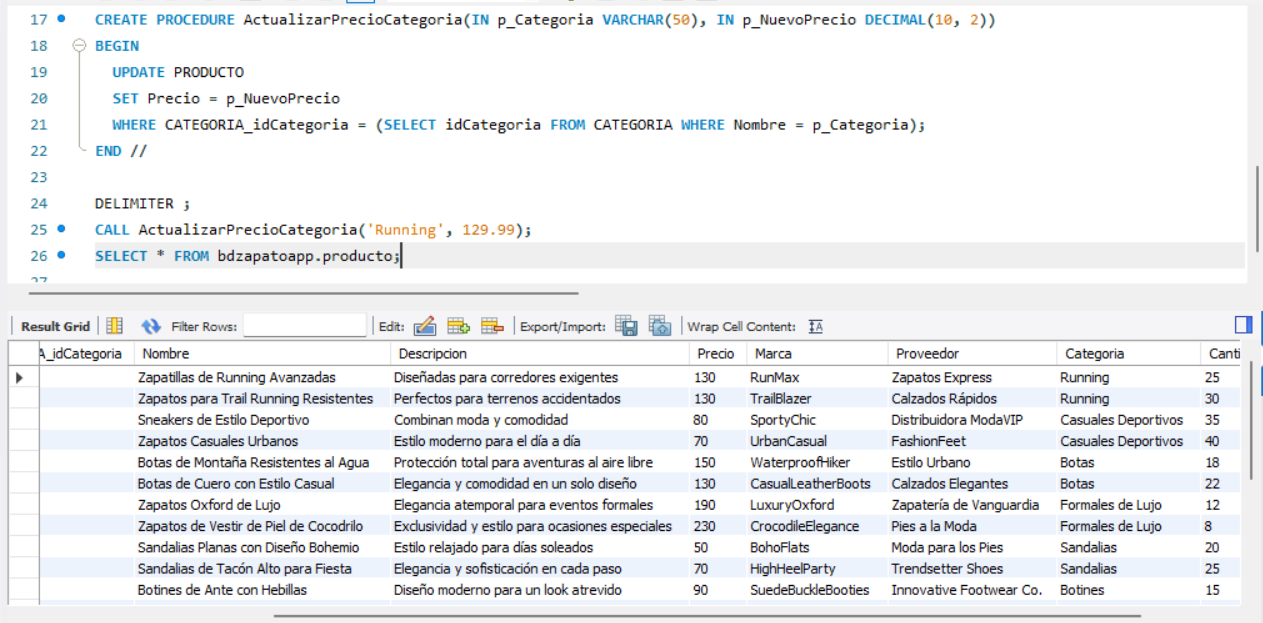
SET Precio = p\_NuevoPrecio

WHERE CATEGORIA\_idCategoria = (SELECT idCategoria FROM CATEGORIA WHERE Nombre = p\_Categoria);

END //

DELIMITER ;

**Resultado:**

****

* **Procedimiento para realizar una venta**

DELIMITER //

CREATE PROCEDURE RealizarVenta(IN p\_EmpleadoID INT, IN p\_Fecha DATE, IN p\_MetodoPago VARCHAR(50), IN p\_Total DECIMAL(10, 2))

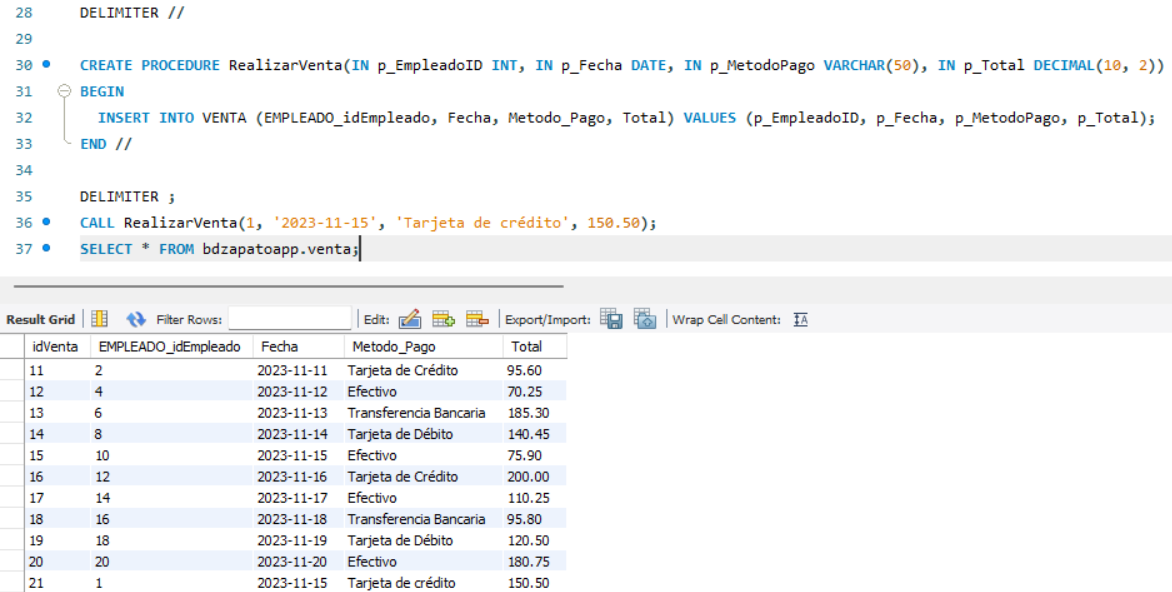
BEGIN

INSERT INTO VENTA (EMPLEADO\_idEmpleado, Fecha, Metodo\_Pago, Total) VALUES (p\_EmpleadoID, p\_Fecha, p\_MetodoPago, p\_Total);

END //

DELIMITER ;

**Resultado:**

****

**FUNCIONES:**

* **Función para calcular el total de una venta**

DELIMITER //

CREATE FUNCTION CalcularTotalVenta(p\_Precio DECIMAL(10, 2), p\_Cantidad INT) RETURNS DECIMAL(10, 2)

DETERMINISTIC

BEGIN

DECLARE total DECIMAL(10, 2);

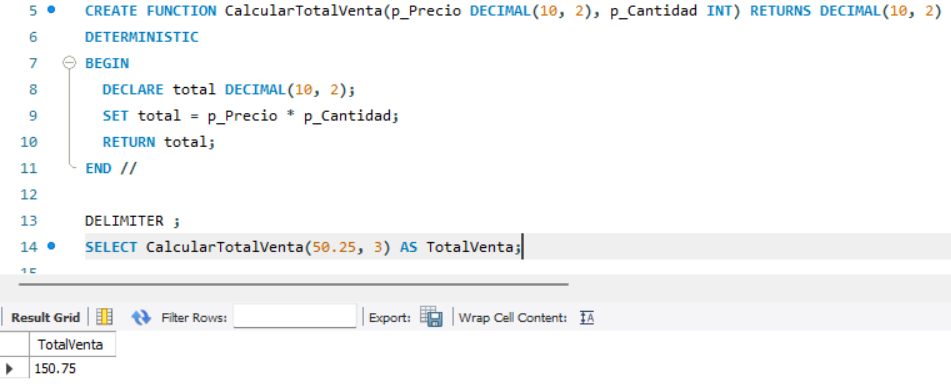
SET total = p\_Precio \* p\_Cantidad;

RETURN total;

END //

DELIMITER ;

**Resultado:**



* **Función para calcular el descuento aplicado a una venta, basándose en un porcentaje de descuento proporcionado**

CREATE FUNCTION CalcularDescuento(p\_TotalVenta DECIMAL(10, 2), p\_PorcentajeDescuento DECIMAL(5, 2)) RETURNS DECIMAL(10, 2)

DETERMINISTIC

BEGIN

DECLARE descuento DECIMAL(10, 2);

-- Calcular el descuento

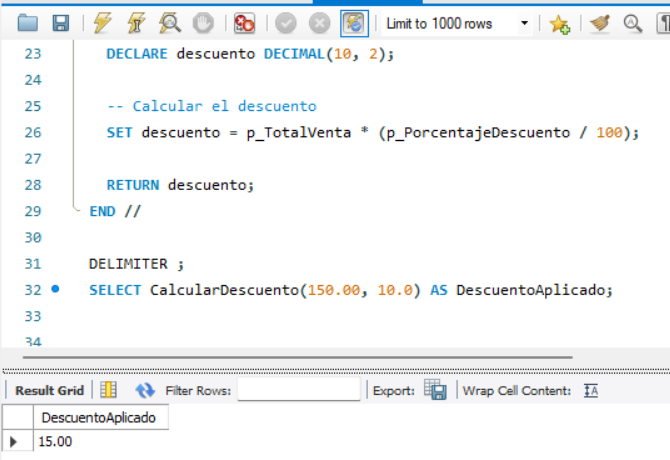
SET descuento = p\_TotalVenta \* (p\_PorcentajeDescuento / 100);

RETURN descuento;

END //

DELIMITER ;

**Resultado:**



* **Función para verificar la cantidad en inventario de un producto específico**

DELIMITER //

CREATE FUNCTION ObtenerCantidadEnInventario(p\_ProductoID INT) RETURNS INT

DETERMINISTIC

BEGIN

DECLARE cantidadDisponible INT;

SELECT Cantidad\_Disponible INTO cantidadDisponible

FROM INVENTARIO\_has\_PRODUCTO

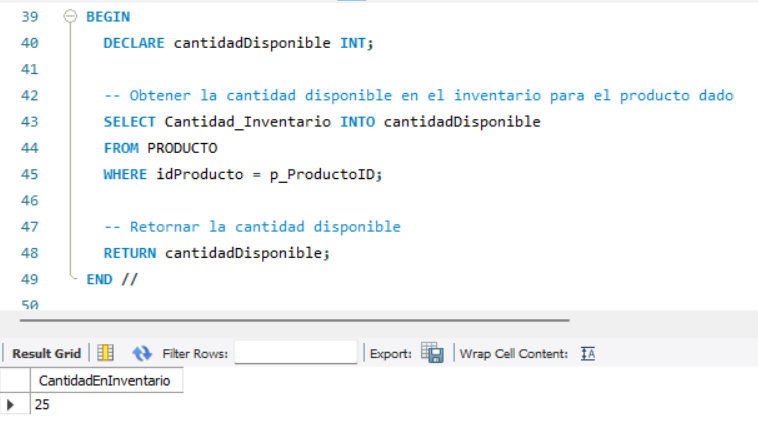
WHERE PRODUCTO\_idProducto = p\_ProductoID;

RETURN cantidadDisponible;

END //

DELIMITER ;

**Resultado:**



**TRIGGERS:**

* **Disparador para registrar cambios en el precio de un producto**

DELIMITER //

CREATE TRIGGER AfterUpdatePrecioProducto

AFTER UPDATE ON PRODUCTO

FOR EACH ROW

BEGIN

IF NEW.Precio != OLD.Precio THEN

INSERT INTO HistorialCambiosPrecio (ProductoID, PrecioAntiguo, PrecioNuevo, FechaCambio)

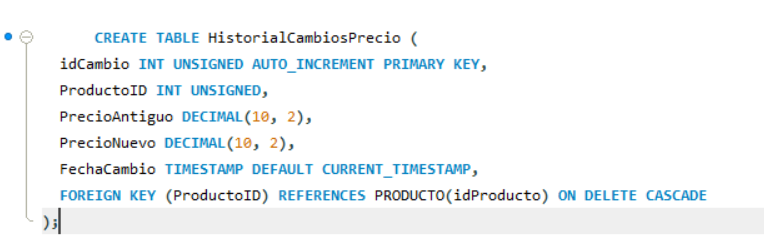
VALUES (NEW.idProducto, OLD.Precio, NEW.Precio, NOW());

END IF;

END //

DELIMITER ;

Se crea la nueva tabla:



**Resultado:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* **Disparador para mantener actualizado el total de ventas de un empleado**

CREATE TRIGGER ActualizarTotalVentas

AFTER INSERT ON VENTA

FOR EACH ROW

BEGIN

UPDATE EMPLEADO

SET TotalVentas = TotalVentas + NEW.Total

WHERE idEmpleado = NEW.EMPLEADO\_idEmpleado;

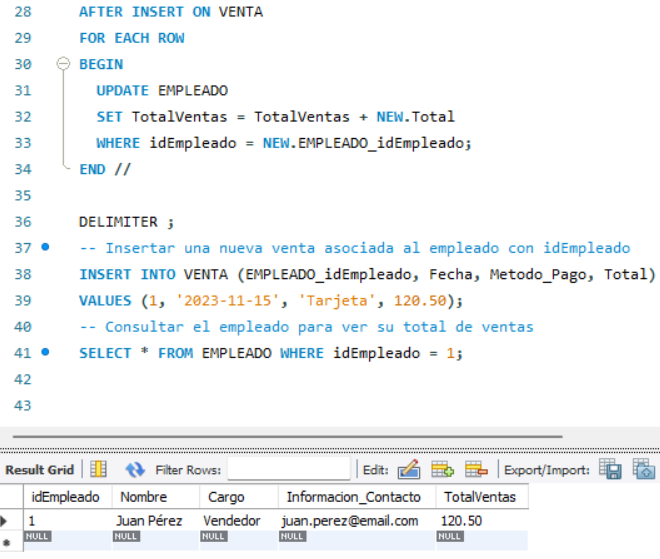
END //

DELIMITER ;

**Se crea una nueva columna en empleado:**



**Resultado:**



* **disparador actualiza automáticamente la fecha de modificación de un producto cada vez que se actualiza la descripción del mismo**

CREATE TRIGGER ActualizarFechaModificacion

BEFORE UPDATE ON PRODUCTO

FOR EACH ROW

BEGIN

SET NEW.FechaModificacion = NOW();

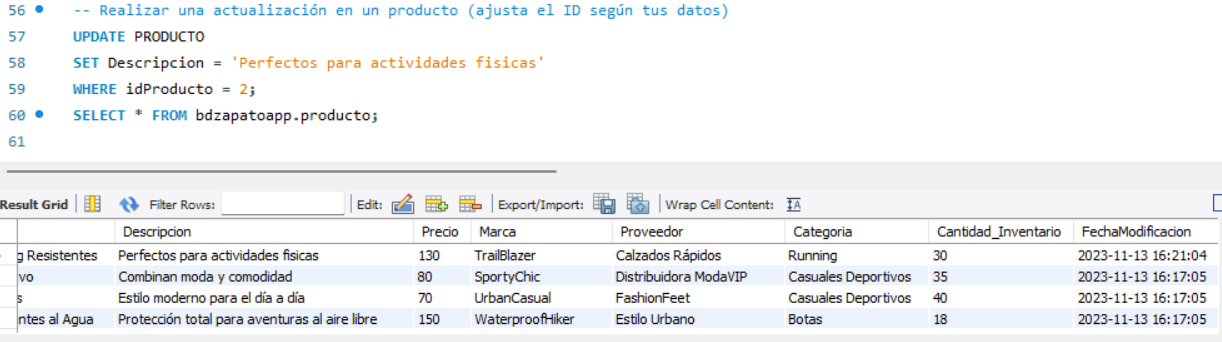
END //

DELIMITER ;

Se crea la nueva columna en la tabla:



**Resultado:**

****