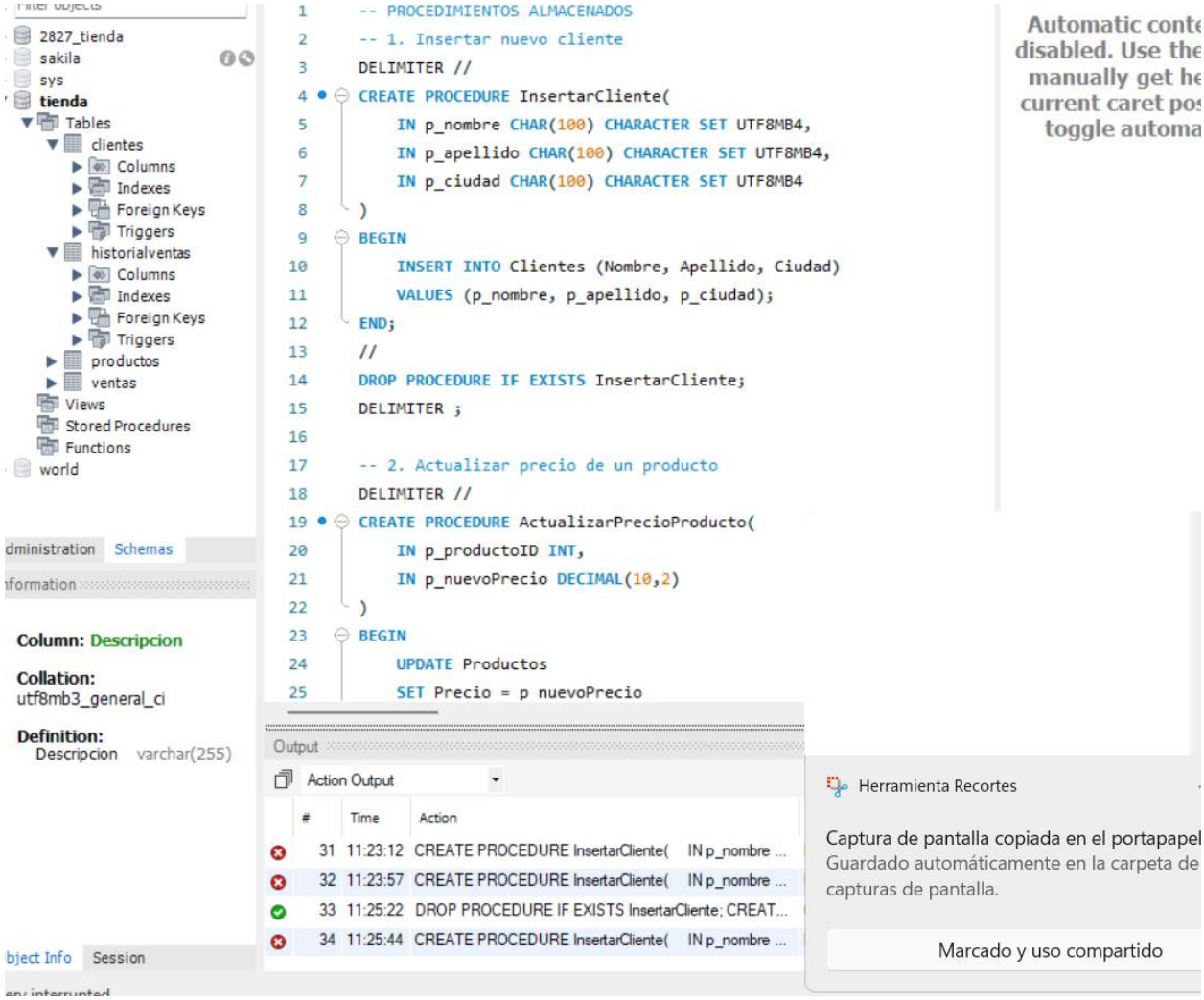
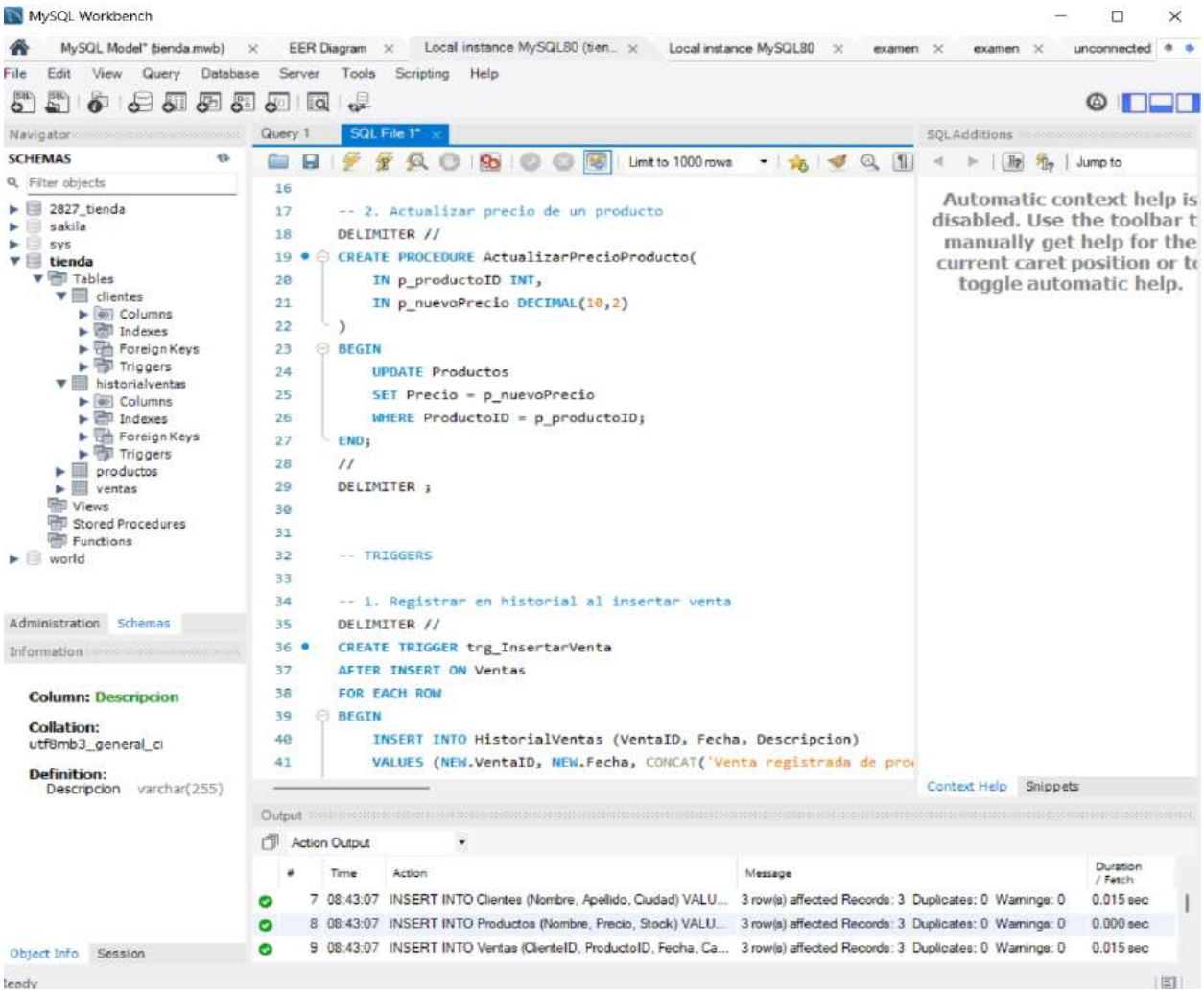
Procedimientos almacenados:

- Crea un procedimiento almacenado que inserte un nuevo cliente en la tabla Clientes




```
DELIMITER //
CREATE PROCEDURE InsertarCliente(
    IN p_nombre CHAR(100) CHARACTER SET UTF8MB4,
    IN p_apellido CHAR(100) CHARACTER SET UTF8MB4,
    IN p_ciudad CHAR(100) CHARACTER SET UTF8MB4
)
BEGIN
    INSERT INTO Clientes (Nombre, Apellido, Ciudad)
    VALUES (p_nombre, p_apellido, p_ciudad);
END;
//
DROP PROCEDURE IF EXISTS InsertarCliente;
DELIMITER ;
```

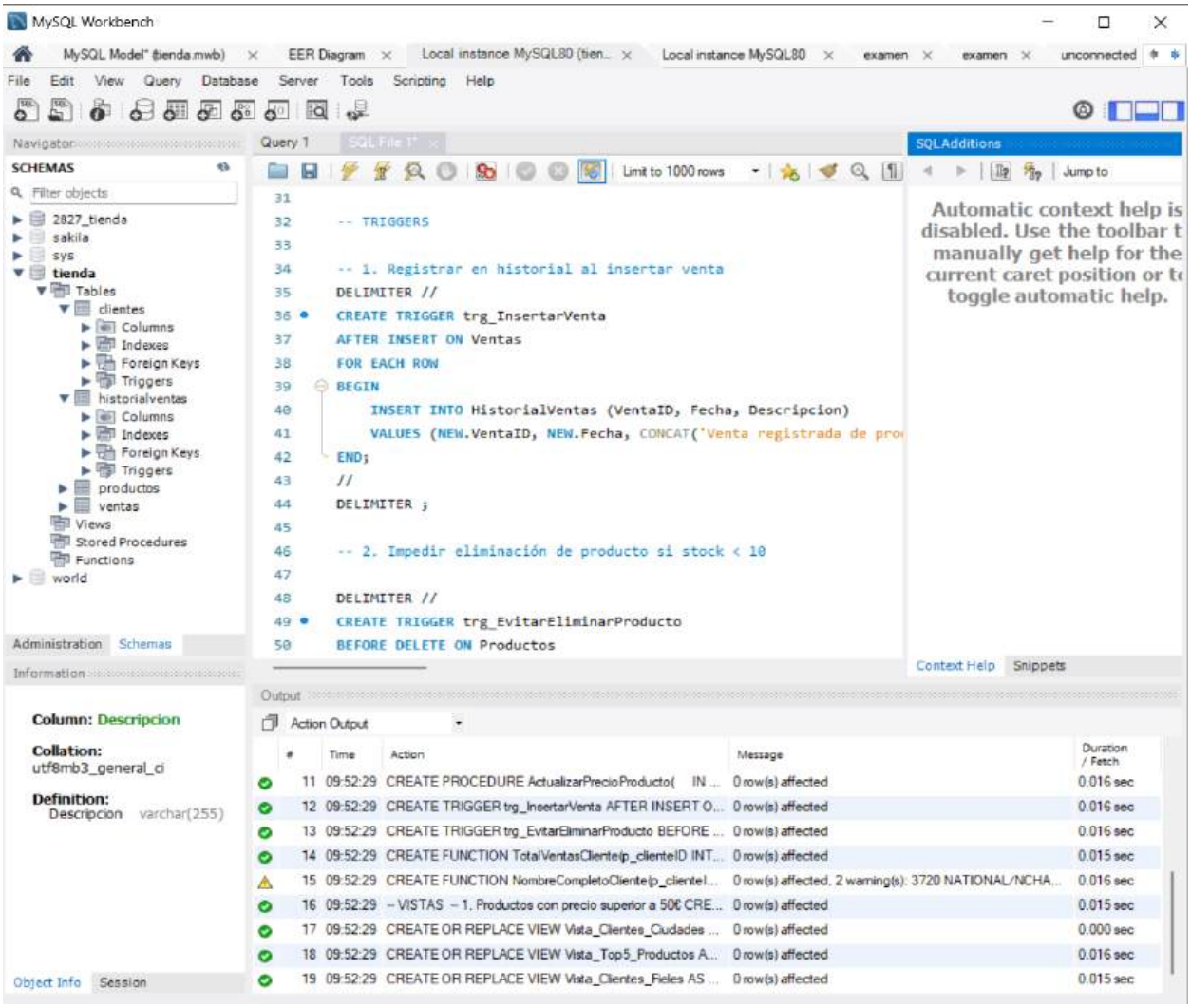
- Crea un procedimiento almacenado que actualice el precio de un producto en la tabla Productos.




```
DELIMITER //
CREATE PROCEDURE ActualizarPrecioProducto(
    IN p_productoID INT,
    IN p_nuevoPrecio DECIMAL(10,2)
)
BEGIN
    UPDATE Productos
    SET Precio = p_nuevoPrecio
    WHERE ProductoID = p_productoID;
END;
//
DELIMITER ;
```

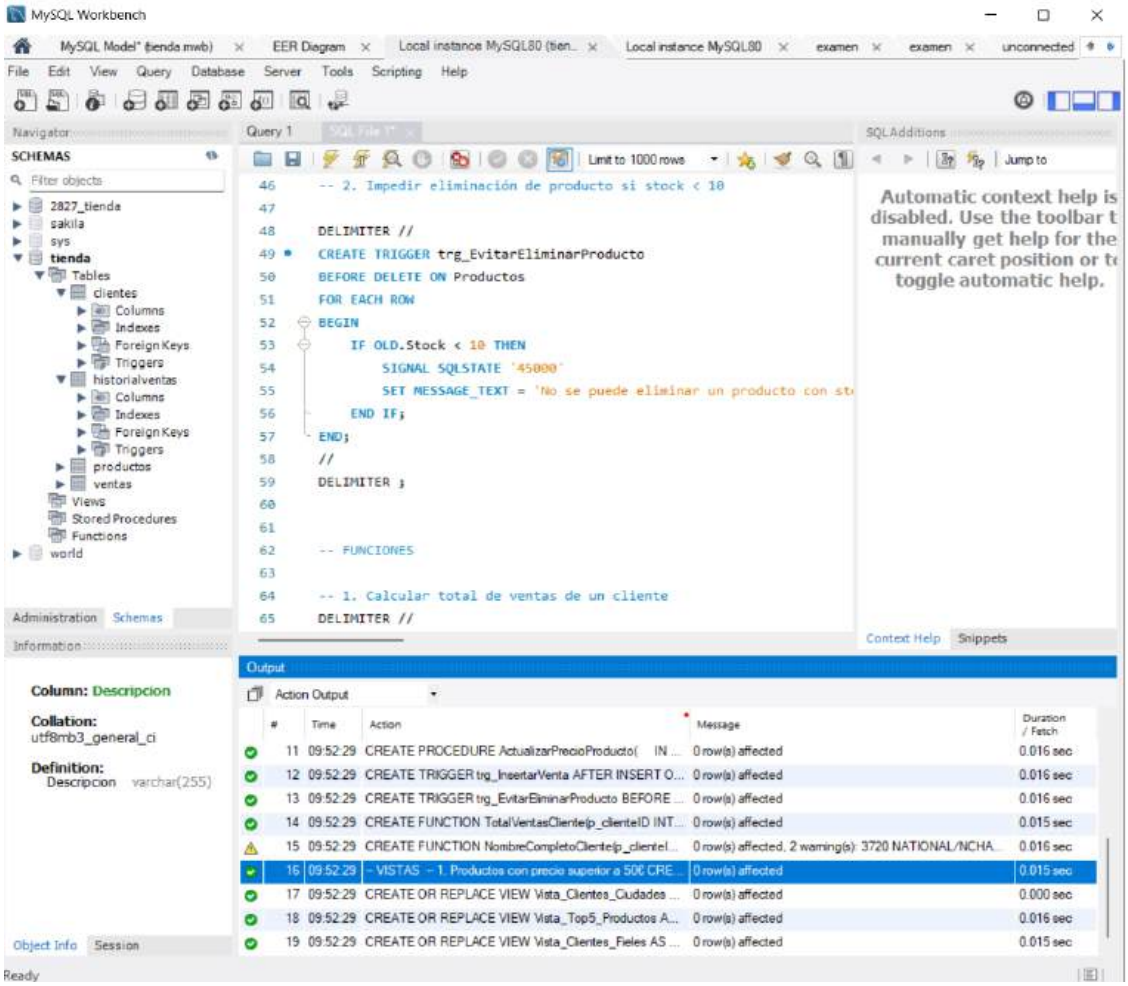
Triggers:

- Crea un trigger que registre en una tabla HistorialVentas cada vez que se inserte una nueva venta en la tabla Ventas.




```
DELIMITER //
CREATE TRIGGER tr_InsertarVenta
AFTER INSERT ON Ventas
FOR EACH ROW
BEGIN
   INSERT INTO HistorialVentas (VentaID, Fecha, Descripcion)
   VALUES (NEW.VentaID, NEW.Fecha, CONCAT('Venta registrada de producto ',
NEW.ProductoID));
END;
//
DELIMITER ;
```

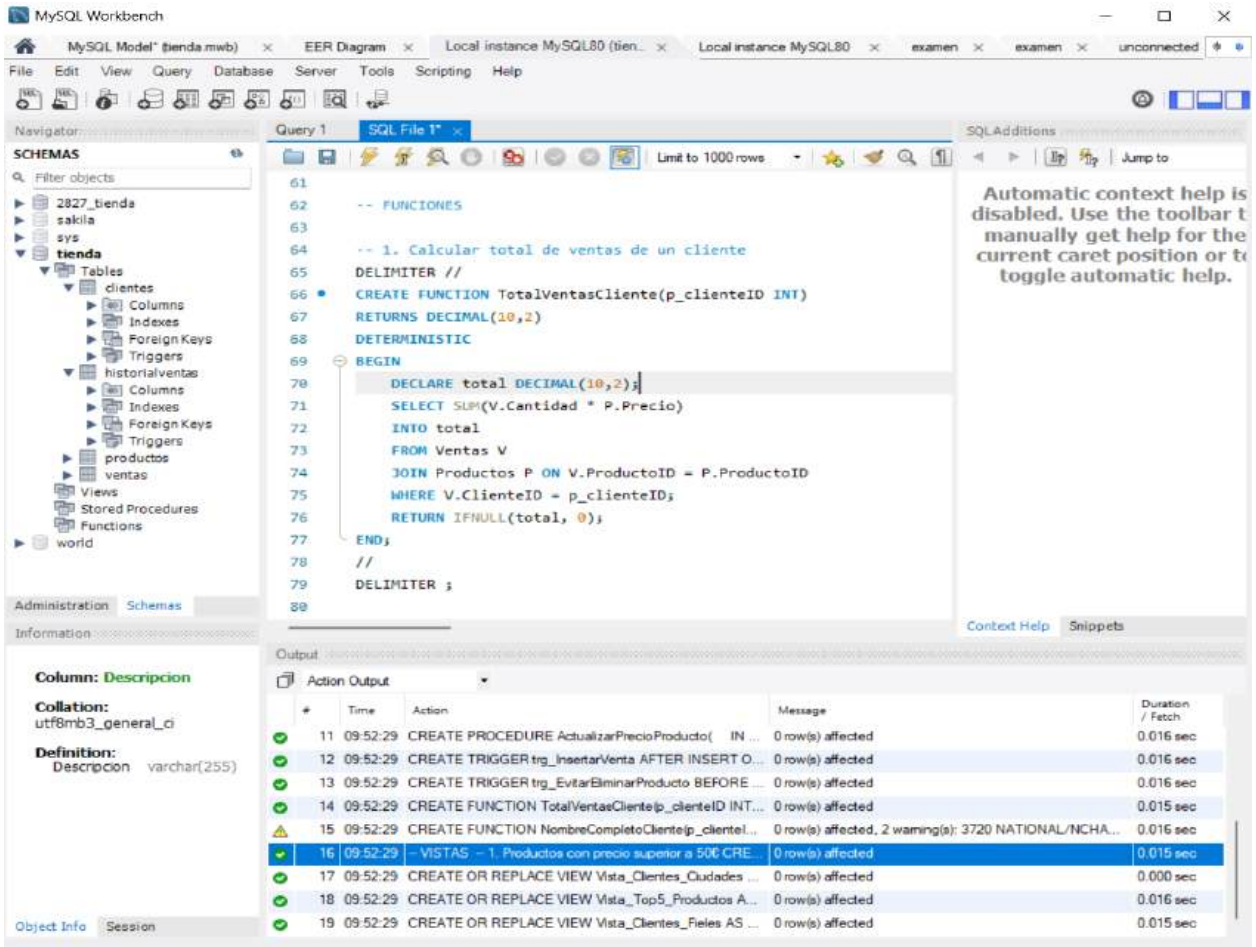
- Crea un trigger que impida la eliminación de productos con un stock inferior a 10 unidades.




```
DELIMITER //
CREATE TRIGGER tr_EvitarEliminarProducto
BEFORE DELETE ON Productos
FOR EACH ROW
BEGIN
   IF OLD.Stock < 10 THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'No se puede eliminar un producto con stock inferior a 10 unidades.';
   END IF;
END;
//
DELIMITER ;
```

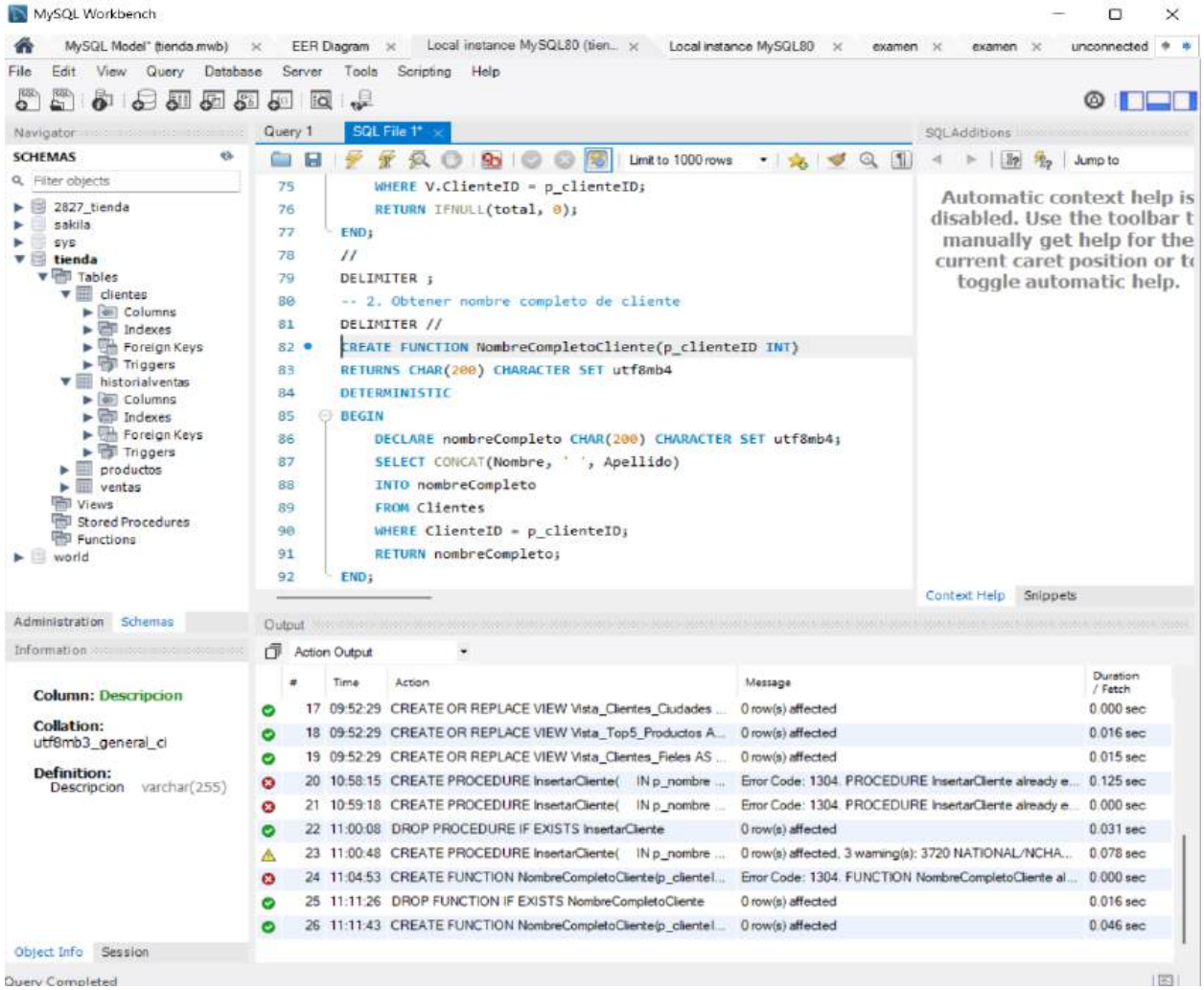
Funciones:

- Crea una función que calcule el total de ventas en un cliente dado su ID.




```
DELIMITER //
CREATE FUNCTION TotalVentasCliente(p_clienteID INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
   DECLARE total DECIMAL(10,2);
   SELECT SUM(V.Cantidad * P.Precio)
   INTO total
   FROM Ventas V
   JOIN Productos P ON V.ProductoID = P.ProductoID
   WHERE V.ClienteID = p_clienteID;
   RETURN IFNULL(total, 0);
END;
//
DELIMITER ;
```

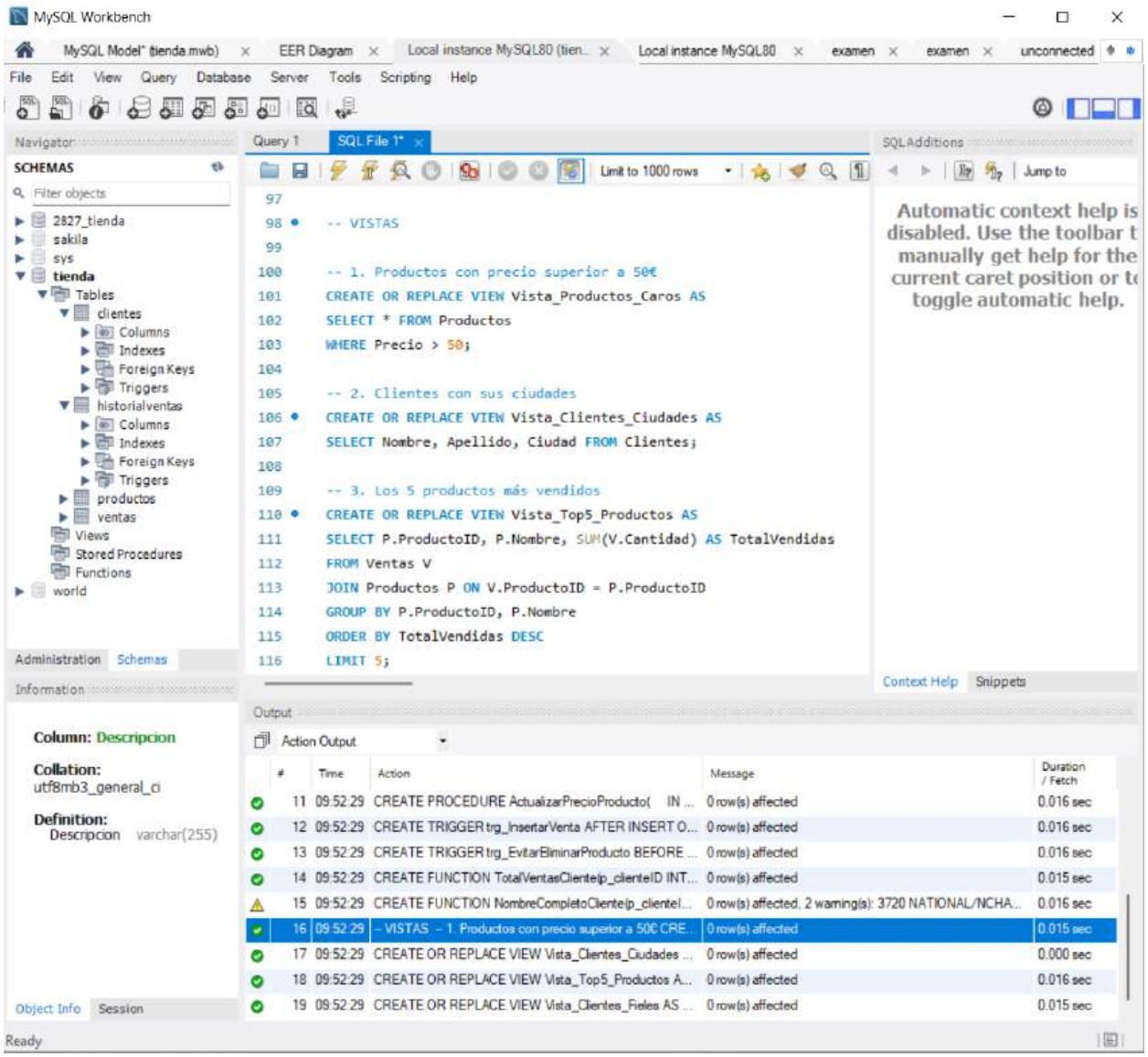
- Crear una función que devuelva el nombre completo de un cliente dado su ID.




```
DELIMITER //

CREATE FUNCTION NombreCompletoCliente(p_clienteID INT)
RETURNS CHAR(200) CHARACTER SET utf8mb4
DETERMINISTIC
BEGIN
   DECLARE nombreCompleto CHAR(200) CHARACTER SET utf8mb4;
   SELECT CONCAT(Nombre, ' ', Apellido)
   INTO nombreCompleto
   FROM Clientes
   WHERE ClienteID = p_clienteID;
   RETURN nombreCompleto;
END;

DELIMITER ;
```

Vistas:

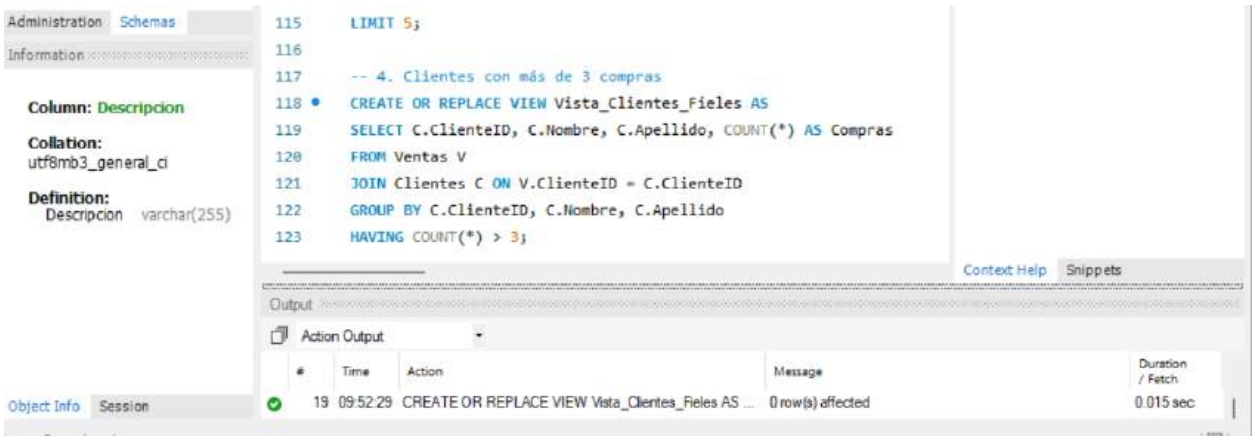
- Crea una vista que muestre los productos con un precio superior a 501€.
- Crea una vista que muestre los clientes y sus respectivas ciudades.
- Crea una vista que devuelva los 5 productos mas vendidos.




```
-- VISTAS

-- 1. Productos con precio superior a 50€
CREATE OR REPLACE VIEW Vista_Productos_Caros AS
SELECT * FROM Productos
```

```
WHERE Precio > 50;

-- 2. Clientes con sus ciudades
CREATE OR REPLACE VIEW Vista_Clientes_Ciudades AS
SELECT Nombre, Apellido, Ciudad FROM Clientes;

-- 3. Los 5 productos más vendidos
CREATE OR REPLACE VIEW Vista_Top5_Productos AS
SELECT P.ProductoID, P.Nombre, SUM(V.Cantidad) AS TotalVendidas
FROM Ventas V
JOIN Productos P ON V.ProductoID = P.ProductoID
GROUP BY P.ProductoID, P.Nombre
ORDER BY TotalVendidas DESC
LIMIT 5;
```

- Crea una vista que devuelva los clientes que han realizado masd e 3 compras.




```
CREATE OR REPLACE VIEW Vista_Clientes_Fieles AS
SELECT C.ClienteID, C.Nombre, C.Apellido, COUNT(*) AS Compras
FROM Ventas V
JOIN Clientes C ON V.ClienteID = C.ClienteID
GROUP BY C.ClienteID, C.Nombre, C.Apellido
HAVING COUNT(*) > 3;
```