Juan David Salinas Muñoz

Taller vistas y procedimientos almacenados

1. Cree una vista con el código, nombre del producto, código y nombre de la categoría y la cantidad total vendida del producto. (77 registros, requiere función de agregado y agrupamiento)

SP que reciba el código de una categoría y muestre el nombre de los productos y la cantidad total vendida de cada producto de la categoría recibida.

create view VwProductoDetails as

select p.idProducto, p.nombreProducto, c.idCategoria, c.nombreCategoria, dp.cantidad from producto as p inner join detalles\_de\_pedido as dp

on p.idProducto=dp.idProducto inner join categoria as c on p.idCategoria=c.idCategoria group by p.idProducto;

delimiter //

create procedure spProductDetails(in id int)

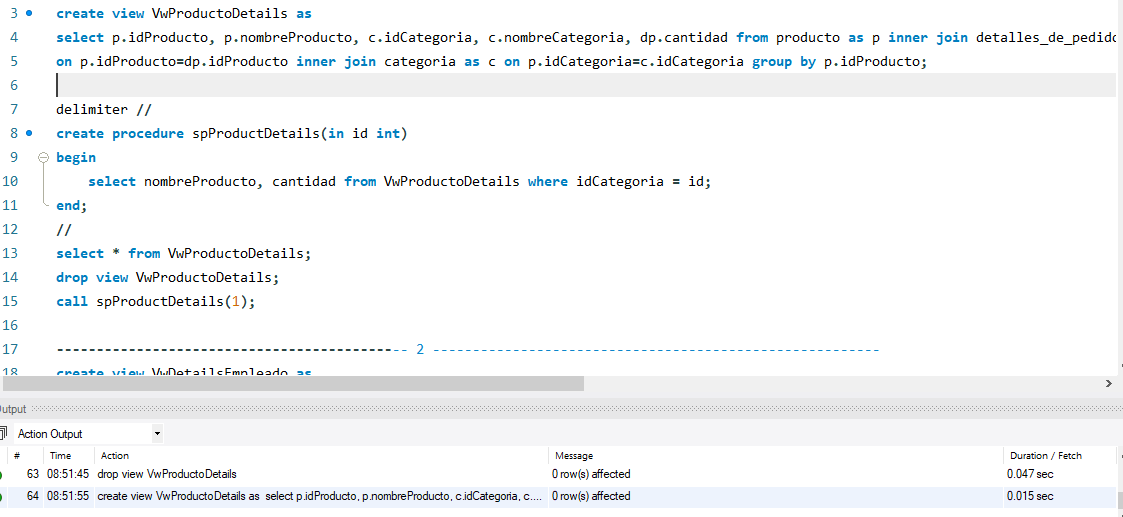
begin

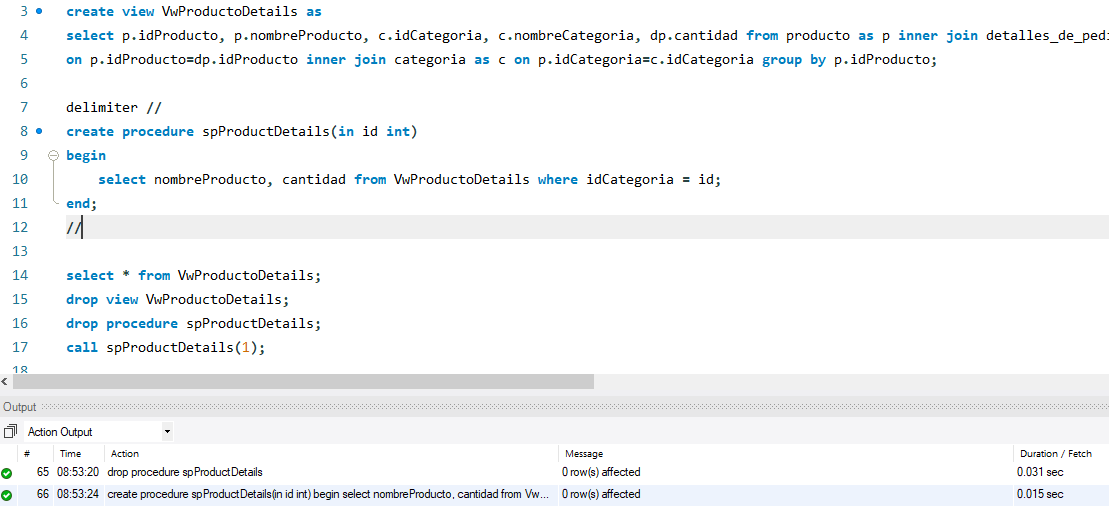
select nombreProducto, cantidad from VwProductoDetails where idCategoria = id;

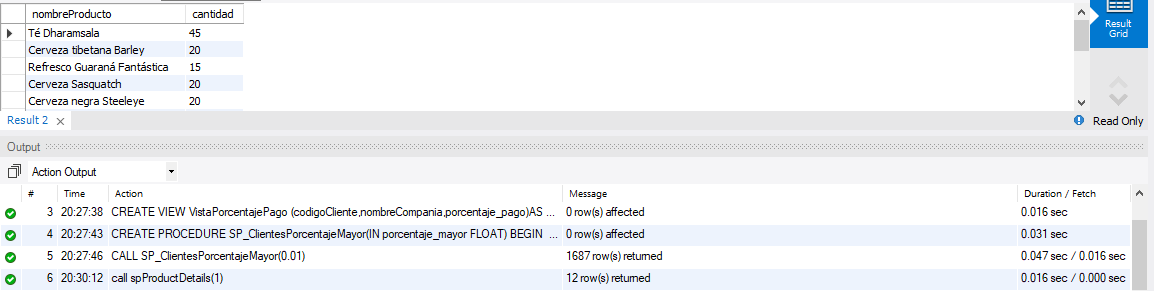
end;

//

call spProductDetails(1);







1. Cree una vista con el nombre completo del empleado, la edad y el total recaudado en ventas por cada uno. (9 registros, requiere función de agregado y agrupamiento)

SP que reciba un número y muestre los empleados que hayan recaudado menos del valor recibido.

create view VwDetailsEmpleado as

select e.nombre, e.apellidos, (2023-year(fechaNacimiento)) as edad, (dp.precioUnidad\*dp.cantidad) as totalVentas from empleado as e inner join pedido as p on p.idEmpleado=e.idEmpleado

inner join detalles\_de\_pedido as dp on dp.idPedido=p.idPedido where e.idEmpleado=p.idEmpleado group by e.idEmpleado;

delimiter //

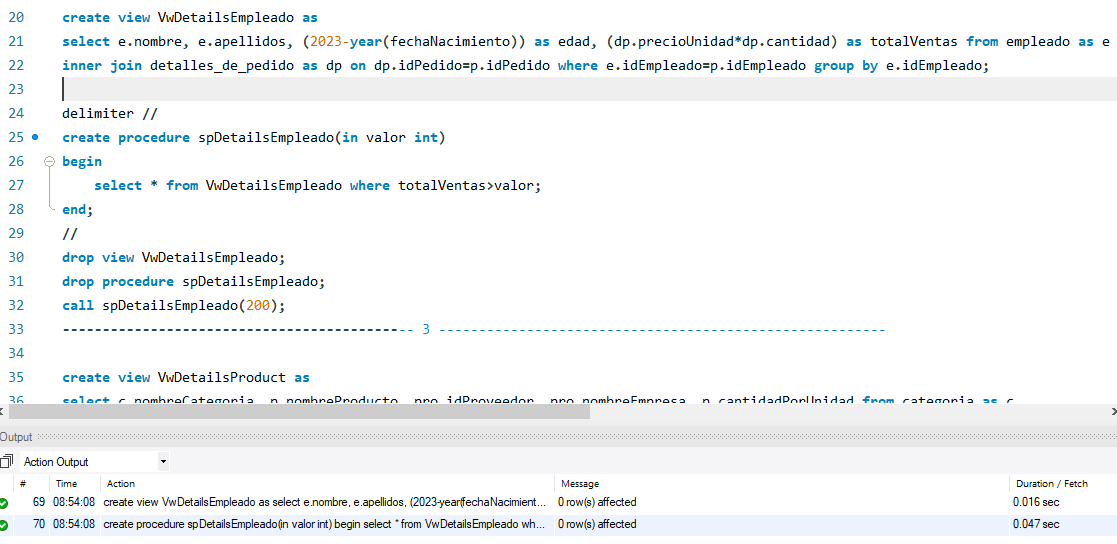
create procedure spDetailsEmpleado(in valor int)

begin

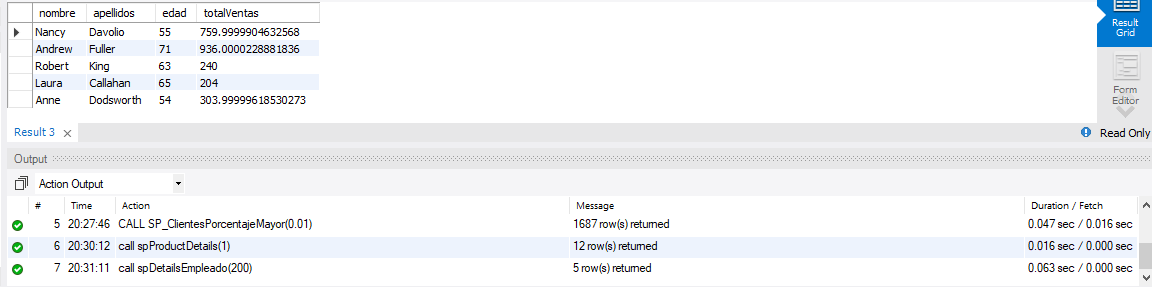
select \* from VwDetailsEmpleado where totalVentas>valor;

end;

//



call spDetailsEmpleado(200);



1. Cree una vista con el nombre de la categoría, el nombre de los productos de cada categoría, el código y nombre del proveedor de cada producto y la presentación del producto.  (77 registros, el empaque está en la columna CantidadPorUnidad de la tabla Producto)

SP que reciba el código de un proveedor y muestre los datos de los productos del proveedor recibido.

create view VwDetailsProduct as

select c.nombreCategoria, p.nombreProducto, pro.idProveedor, pro.nombreEmpresa, p.cantidadPorUnidad from categoria as c

inner join producto as p on p.idCategoria=c.idCategoria

inner join proveedor as pro on p.idProveedor=pro.idProveedor;

select \* from VwDetailsProduct;

delimiter //

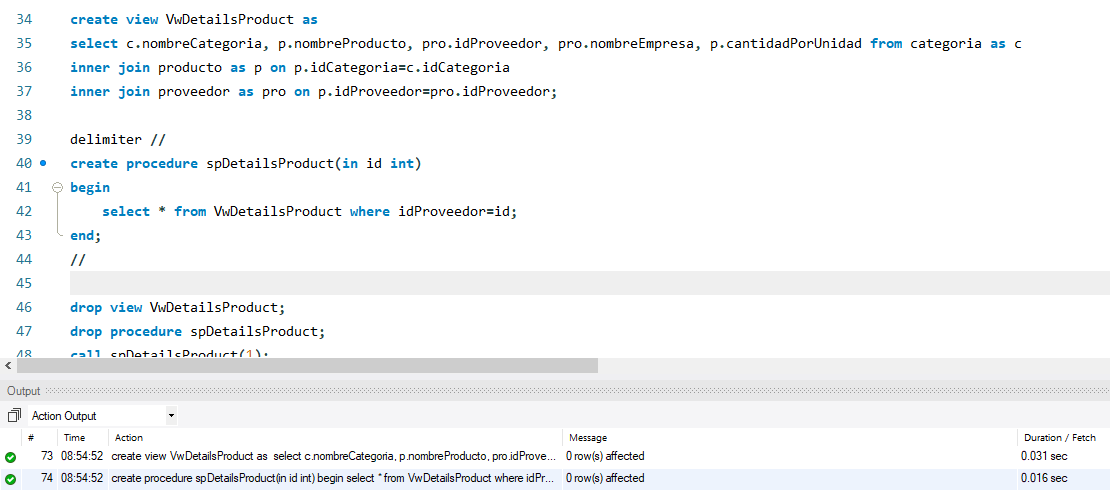
create procedure spDetailsProduct(in id int)

begin

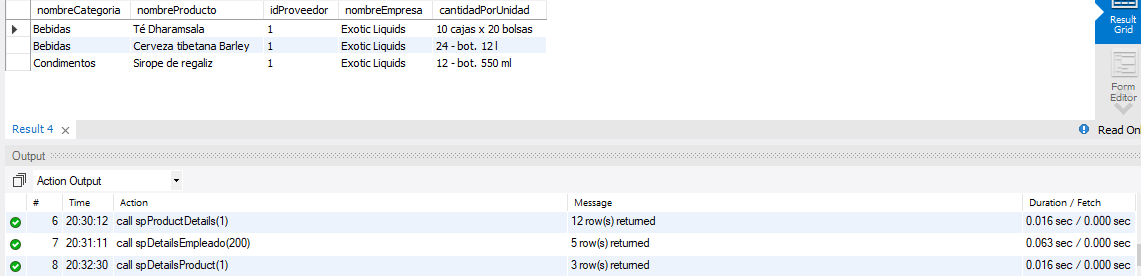
select \* from VwDetailsProduct where idProveedor=id;

end;

//



call spDetailsProduct(1);



1. Cree una vista con el código del cliente, el nombre del cliente y el total pagado por cada cliente en todas las compras realizadas por mes en cada año. (son 89 clientes, son 3 años y son 12 meses por año)

SP que reciba dos fechas que corresponden a un periodo de tiempo y muestre los datos contenidos en la vista correspondientes al periodo de tiempo recibido.

create view VwClienteDetails (idCliente, nombreEmpresa,totalPrecio,fechaPedido) as

select c.idCliente, c.nombreEmpresa, (dp.cantidad\*dp.precioUnidad) as totalPrecio, Month(p.fechaPedido) from cliente as c inner join pedido as p on p.idCliente=c.idCliente

inner join detalles\_de\_pedido as dp on dp.idPedido=p.idPedido order by c.idCliente and Month(p.fechaPedido);

drop view VwClienteDetails;

select \* from VwClienteDetails;

delimiter //

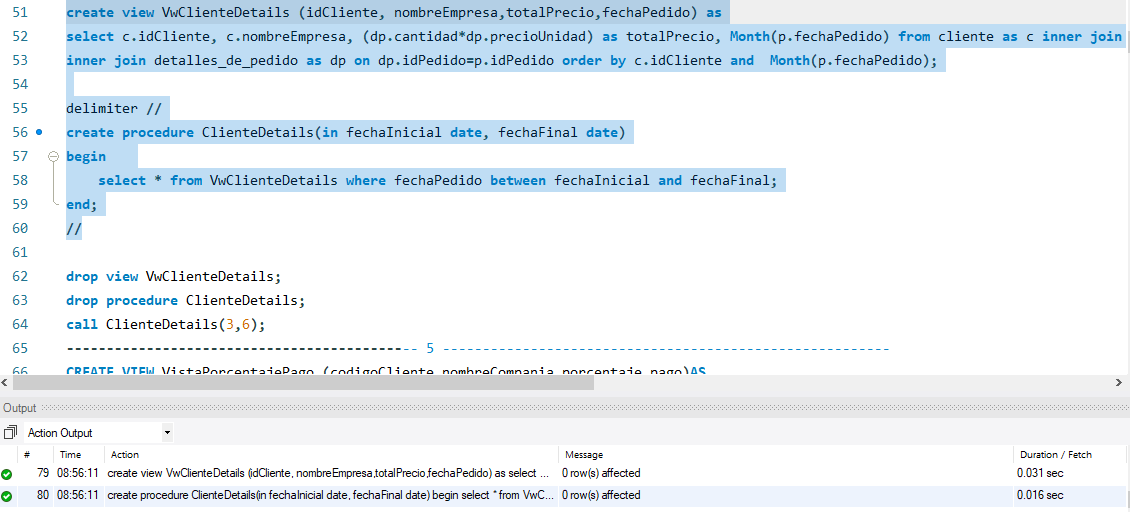
create procedure ClienteDetails(in fechaInicial date, fechaFinal date)

begin

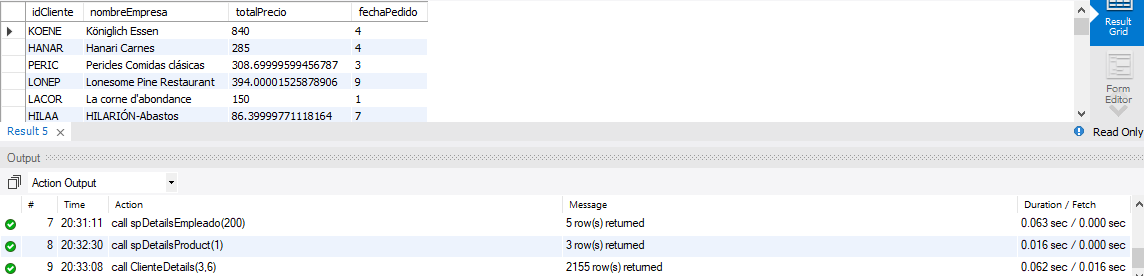
select \* from VwClienteDetails where fechaPedido between fechaInicial and fechaFinal;

end;

//



call ClienteDetails(3,6);



1. Cree una vista con el código del cliente, el nombre del cliente (nombre compañía) y el porcentaje que el total pagado por cada cliente representa frente al total de todas las ventas registradas.  Utilice una subconsulta para obtener el total de todas las ventas y la vista del punto 4 para obtener el porcentaje.

SP que reciba un valor de tipo float y muestre los clientes que hayan pagado un porcentaje mayor al dato recibido.

CREATE VIEW VistaPorcentajePago (codigoCliente,nombreCompania,porcentaje\_pago)AS

SELECT

IdCliente,

NombreEmpresa,

(totalPrecio / totalVentas) \* 100 AS porcentaje\_pago

FROM (

SELECT

vws.IdCliente,

vws.NombreEmpresa,

vws.totalPrecio,

(SELECT SUM(totalPrecio) FROM VwClienteDetails) AS totalVentas

FROM VwClienteDetails vws

) AS subconsulta;

DELIMITER //

CREATE PROCEDURE SP\_ClientesPorcentajeMayor(IN porcentaje\_mayor FLOAT)

BEGIN

SELECT

codigoCliente,

nombreCompania,

porcentaje\_pago

FROM

VistaPorcentajePago

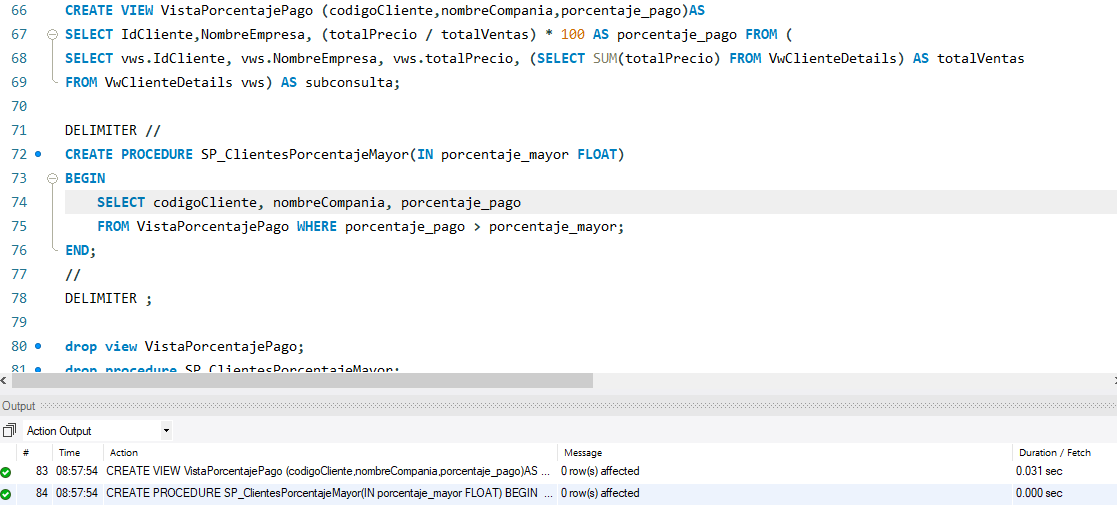
WHERE

porcentaje\_pago > porcentaje\_mayor;

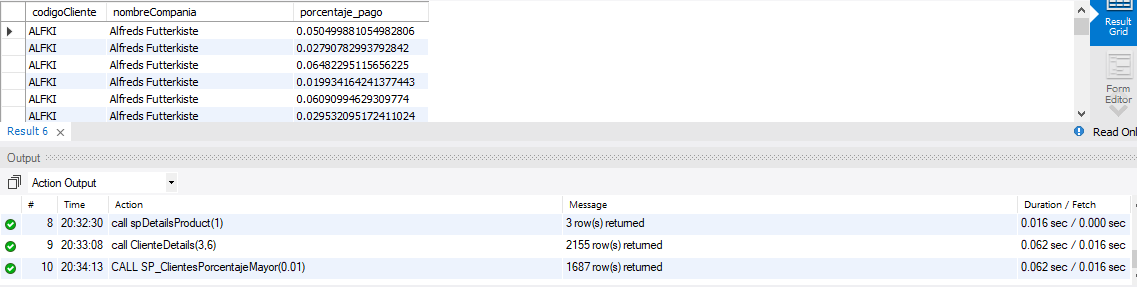
END;

//

DELIMITER ;



CALL SP\_ClientesPorcentajeMayor(0.01); -- el procedimiento mostrara los clientes con un porcentaje de pago mayor al 30%



1. Crear una vista con todas las columnas de la tabla productos, y solo las filas en las que el precio unidad es mayor a 20. Incluya el criterio with check option. Esta vista es para actualizar la tabla Productos.

CREATE VIEW vista\_productoV2 AS

SELECT \* FROM producto WHERE PrecioUnidad > 20

with check option;

Cree un SP para Insertar en la vista.

Para comprobar los objetos: envíe un registro a la vista a través del SP en el cual incluya como precio unidad 30. Describa lo que sucede.

DELIMITER //

CREATE PROCEDURE spInsertarProducto(in id\_Producto int,IN nombre\_producto VARCHAR(80), IN precio\_unidad FLOAT)

BEGIN

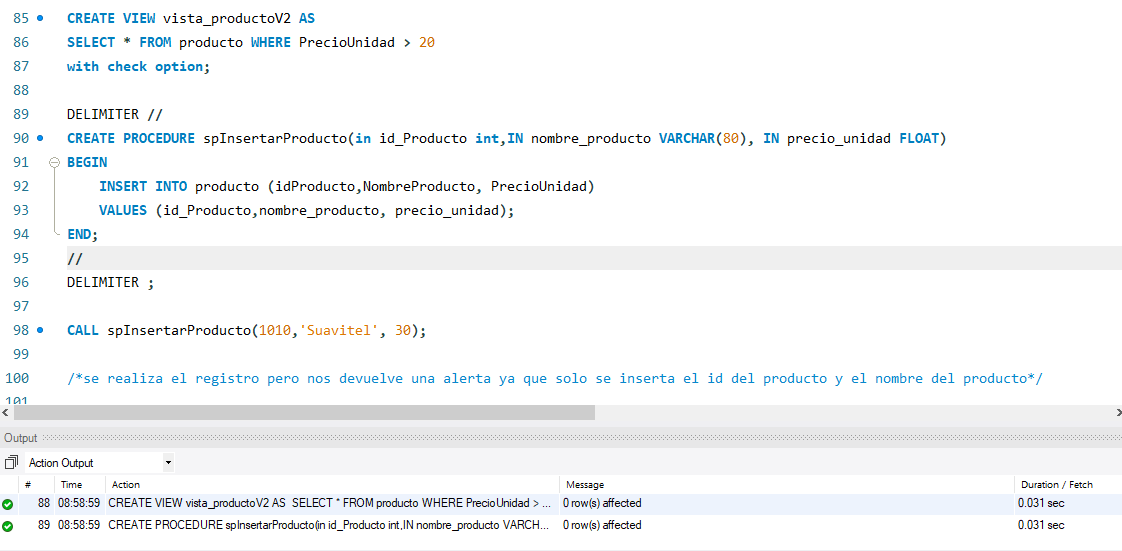
INSERT INTO producto (idProducto,NombreProducto, PrecioUnidad)

VALUES (id\_Producto,nombre\_producto, precio\_unidad);

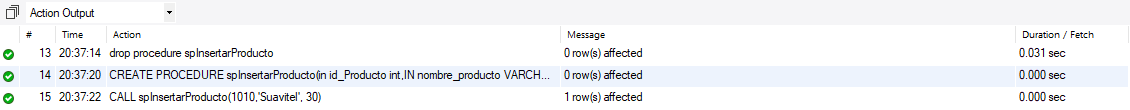
END;

//

DELIMITER ;



CALL spInsertarProducto(1010,'Suavitel', 30);



Cree un SP para modificar el precio de un producto a través de la vista, debe recibir el código del producto y el nuevo precio, pero además el nuevo precio no debe aumentarse o disminuirse más del 5% del precio actual del producto.

/\*se realiza el registro pero nos devuelve una alerta ya que solo se inserta el id del producto y el nombre del producto\*/

DELIMITER //

CREATE PROCEDURE SP\_modificarProduV2(codProducto int, precio float)

BEGIN

SET @PrecioActual = (SELECT PrecioUnidad FROM vista\_productoV2 WHERE idProducto = codProducto);

IF precio between @PrecioActual \* 0.95 AND @PrecioActual \* 1.05 THEN

UPDATE vista\_productoV2 SET PrecioUnidad = precio WHERE idProducto = codProducto;

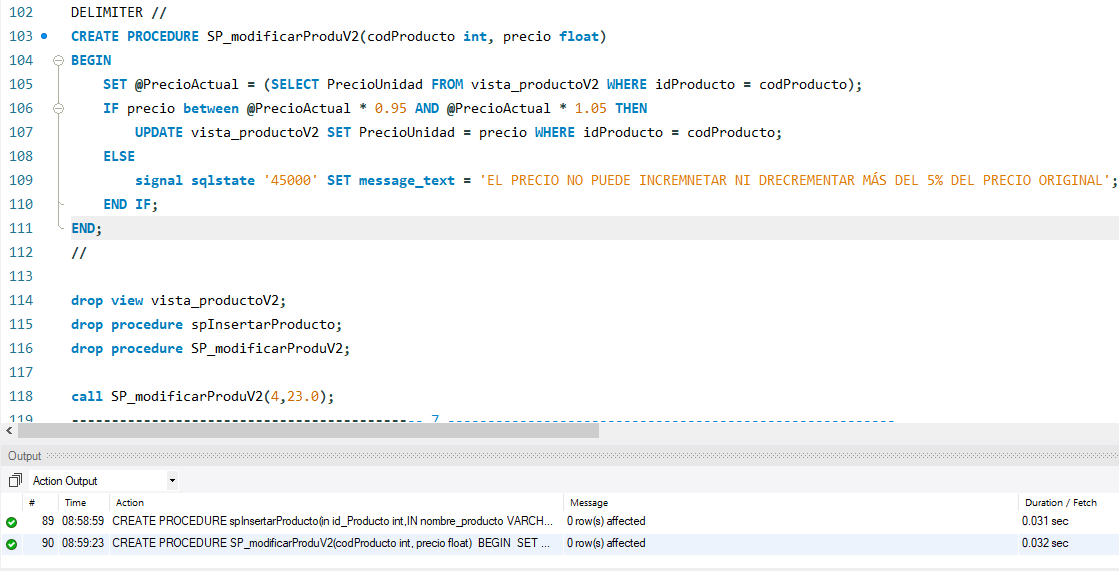
ELSE

signal sqlstate '45000' SET message\_text = 'EL PRECIO NO PUEDE INCREMNETAR NI DRECREMENTAR MÁS DEL 5% DEL PRECIO ORIGINAL';

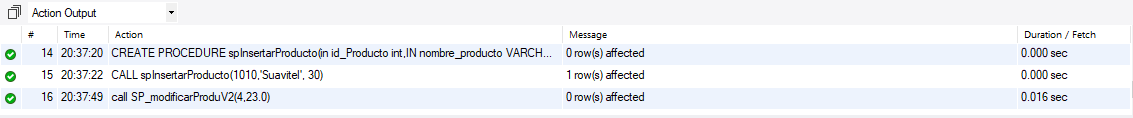
END IF;

END;

//



call SP\_modificarProduV2(4,23.0);



1. Cree una vista con el requisito Distinct con las regiones de origen de los clientes, omita las filas con valor nulo. Explique con sus palabras la funcionalidad del requisito Distinct, en la creación de vistas.

El funcionamiento del requisito distinct lo que busca es permitir que en la consulta se eliminen registros duplicados y de esta manera solo se mostraran aquellos datos que sean unicos

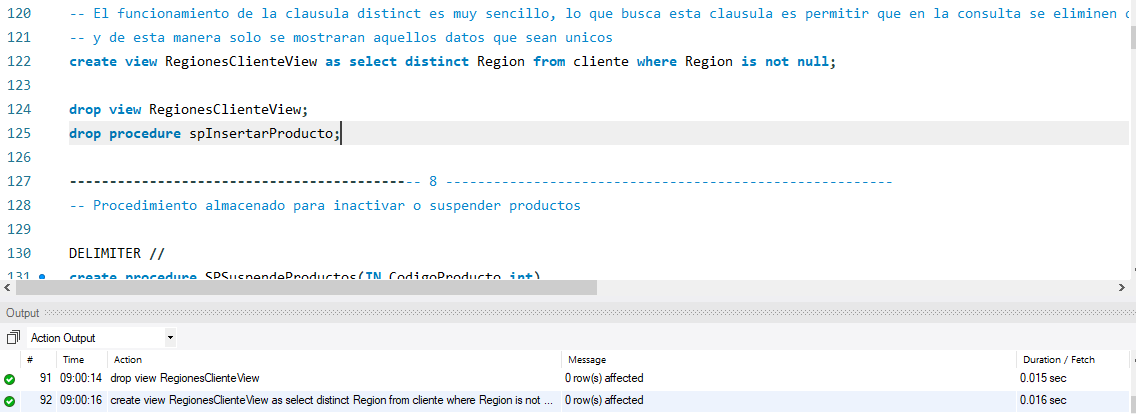
create view RegionesClienteView

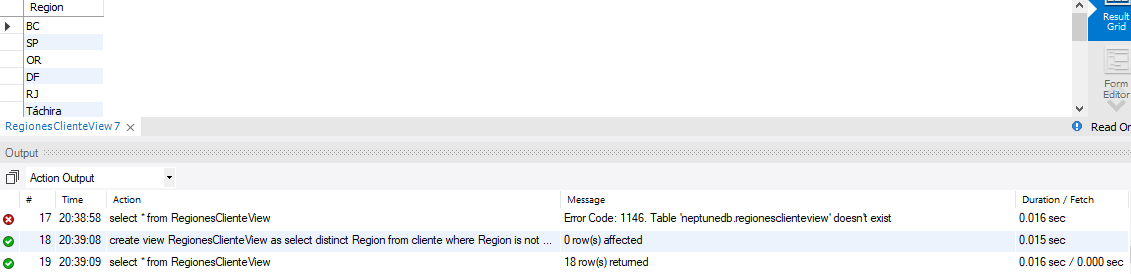
as

select distinct Region from cliente

where Region is not null;

select \* from RegionesClienteView;





1. Crear un SP que permita suspender los productos, debe recibir el código del producto, tenga en cuenta que solo debe permitir suspender el producto cuando las unidades en pedido sean 0.

-- Procedimiento almacenado para inactivar o suspender productos

DELIMITER //

create procedure SPSuspendeProductos(IN CodigoProducto int)

begin

-- variable de cantidades en pedido

declare UnidadEnPedido int;

-- variable de mensaje de error

declare ErrorExistencias varchar (70);

-- Asignacion de cantidad de existencias

set UnidadEnPedido = (select UnidadesEnExistencia from producto where IdProducto = CodigoProducto);

if UnidadEnPedido = 0 then

update producto

set Suspendido = 1

where IdProducto = CodigoProducto

;

select IdProducto, NombreProducto, Suspendido

from producto

where IdProducto=CodigoProducto;

else

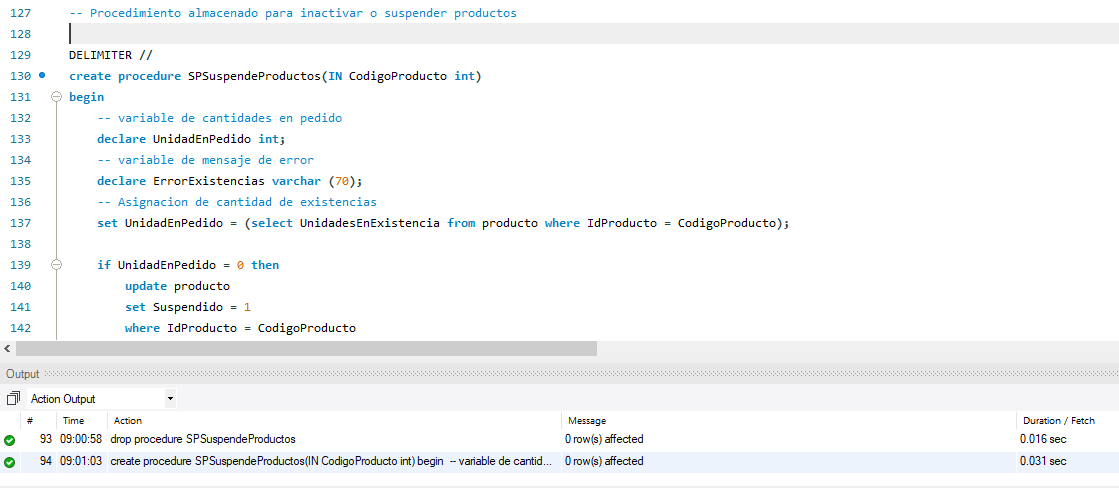
set ErrorExistencias = 'La cantidad de existencias es mayor o menor a 0 y no se puede suspender';

select ErrorExistencias;

end if;

end;

//



call SPSuspendeProductos(10);

