**1.- Crear la función Cantidad\_Efectivo que recibe como parámetro el DNI de un cliente y devuelve un entero indicando la cantidad de medicamentos comprados en efectivo por ese cliente.**

Delimiter //

create function cantidad\_efectivo(dni char(9)) returns int reads sql data

begin

declare total int default 0;

if exists(select \* from comp\_efec where dni=dni\_cliente) then

select sum(unidades) into total from comp\_efec where dni\_cliente=dni;

end if;

return total;

end//

delimiter ;

**2.- Crear la función Cantidad\_Credito que recibe como parámetro el DNI de un cliente y**

**devuelve un entero indicando la cantidad de medicamentos comprados a crédito por ese**

**cliente.**

Delimiter //

create function cantidad\_credito(dni char(9)) returns int reads sql data

begin

declare total int default 0;

if exists(select \* from comp\_cred where dni=dni\_cliente) then

select sum(unidades) into total from comp\_cred where dni\_cliente=dni;

end if;

return total;

end//

delimiter ;

**3.- Crear la función Cantidad\_Total que recibe como parámetro el DNI de un cliente y devuelve un entero indicando la cantidad total de medicamentos comprados por ese cliente, en efectivo y a crédito. Usar para ello las funciones Cantidad\_Efectivo y Cantidad\_Credito.**

Delimiter //

create function cantidad\_total (dni char(9)) returns int reads sql data

begin

declare total int default 0;

select sum(cantidad\_credito(dni) + cantidad\_efectivo(dni)) into total;

return total;

end//

delimiter ;

**4.- Crear la función Deuda\_Cliente, que recibe como parámetro el DNI de un cliente y devuelve un float(11,3) indicando la cantidad total que ese cliente nos debe por los medicamentos que se ha llevado a crédito y todavía no ha pagado.**

Delimiter//

create function deuda\_cliente(dni char(9)) returns float(11,3) reads sql data

begin

declare total float(11,3);

select sum(unidades\*precio) into total from comp\_cred c, medicamento m where dni\_cliente=dni and c.cod\_med=m.cod\_med and fecha\_pago is null;

return total;

end//

delimiter ;

**5.- Crear la función Caja\_Fecha que recibe como parámetro una fecha y devuelve un float(11,3) indicando el importe total de todos medicamentos que ese día se han vendido, tanto a crédito como en efectivo.**

Delimiter //

create function caja\_fecha(fecha date) returns float(11,3) reads sql data

begin

declare t\_efect float(11,3) default 0;

declare t\_cred float(11,3) default 0;

select sum(precio\*unidades) into t\_efect

from comp\_efect e

join medicamento m on e.cod\_med=m.cod\_med and fecha=fecha\_pago;

select sum(precio\*unidades) into t\_cred

from comp\_cred c

join medicamento m on c.cod\_med=m.cod\_med and fecha=fecha\_pago;

return t\_efect+t\_cred;

end//

delimiter ;

**6.- Crear el procedimiento Nuevo\_Cliente que recibe como parámetros los datos de un cliente y los inserta en la correspondiente tabla. Si el cliente ya existe o los datos bancarios coinciden con los de otro cliente el procedimiento no insertará nada y deberá hacer que aparezca un mensaje de error indicándolo.**

Delimiter //

create procedure nuevo\_cliente (dni char(9), tel char(9), dir varchar(30), banco varchar(25))

begin

declare dni\_cli char(9);

declare datos varchar(25);

select dni\_cliente, datos\_banco into dni\_cli, datos

from cliente

where dni\_cliente=dni or datos\_banco=banco;

if(dni\_cli is not null or datos is not null) then

set message\_text=’error: el cliente ya existe o los datos bancarios coinciden’;

else

insert into cliente(dni\_cliente, telefono, direccion, datos\_banco)

values(dni, tel, dir, banco);

end if;

end//

delimiter ;

**7.- Crear el procedimiento Borra\_Familia que recibirá como parámetro el código de una familia de medicamentos y deberá hacer lo siguiente:**

**1.- Comprobar que esa familia de medicamentos exista y en caso positivo:**

**2.- Iniciar una transacción**

**3.- Eliminar todas las compras en efectivo y a crédito de los medicamentos de esa familia. Antes de eliminarlas deberá guardarlas en una tabla de histórico compras con la misma estructura que las tablas de compra salvo el campo Fech\_Pago.**

**4.- Eliminar todos los medicamentos de esa familia.**

**5.- Eliminar la familia**

**6.- Validar la transacción.**

Delimiter //

create procedure borra\_familia(cod\_fam char(5))

begin

declare fam char(5);

select cod\_ familia into fam from familia where cod\_familia=cod\_fam;

start transaction;

if(fam is not null) then

insert into historico\_compras

select \* from comp\_cred where cod\_med in (select cod\_med from medicamento where cod\_familia=cod\_fam);

select \* from comp\_efect where cod\_med in (select cod\_med from medicamento where cod\_familia=cod\_fam);

delete from com\_efect where cod\_med in (select cod\_med from medicamento where cod\_familia=cod\_fam);

delete from com\_cred where cod\_med in (select cod\_med from medicamento where cod\_familia=cod\_fam);

delete from medicamento where cod\_familia=cod\_fam;

delete from familia where cod\_familia=cod\_fam;

commit;

ELSE

ROLLBACK;

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Error: La familia de medicamentos no existe.';

end if;

end//

delimiter ;

**8.- Crear el procedimiento Borra\_Cliente que recibirá como parámetro el DNI de un cliente y realizará los siguientes pasos:**

**1.- Comprobar que dicho cliente existe y que NO tiene compras sin pagar. En caso positivo:**

**2.- Iniciar una transacción**

**3.- Eliminar todas las compras en efectivo y a crédito de los medicamentos que ese cliente haya realizado. Antes de eliminarlas deberá guardarlas en una tabla de histórico compras con la misma estructura que las tablas de compra salvo el campo Fech\_Pago.**

**4.- Eliminar ese cliente.**

**6.- Validar la transacción.**

delimiter //

create procedure borra\_cliente(dni char(9))

begin

declare cli char(9);

select dni\_cliente into cli from cliente where dni\_cliente=dni;

if(cli is not null) and (deuda\_cliente(dni)=0.0)

then

start transaction;

insert into historico\_compras

select \* from comp\_efect

where dni\_cliente =dni;

insert into historico\_compras

select \* from comp\_cred

where dni\_cliente=dni;

delete from comp\_efect where dni\_cliente=dni;

delete from comp\_cred where dni\_cliente=dni;

delete from cliente where dni\_cliente =dni;

commit;

else

rollback;

signal sqlstate ‘45000’

set message\_text=’error: la familia no existe’;

end if

end//

delimiter ;

9.- Crear el procedimiento Cambia\_Familia que recibe cómo parámetros el código de un medicamento y el código de una familia de medicamentos y asigna dicha familia a ese medicamento, comprobando previamente que el medicamento y la familia existan.

10.- Crear un disparador para que en caso de que un medicamento cambie de precio y pase a valer menos que antes, se registre en una tabla Cambios\_Precio el código del medicamento, la fecha en que se le cambió de precio y el usuario que llevó a cabo el cambio. Se pueden usar las funciones Curdate y User.

11.- Crear un disparador para que en caso de que un medicamento pase de no requerir receta a requerirla, se registre en una tabla Cambios\_Receta el código del medicamento, la fecha en que se le cambió de precio y el usuario que llevó a cabo el cambio. Se pueden usar las funciones Curdate y User.

12.- Crear un disparador que haga que en caso de cambio del teléfono o la dirección de un cliente, los anteriores teléfonos o direcciones se guarden en una tabla de histórico de cambios para no perderlos.

13.- Crear los disparadores necesarios para evitar que se pueda registrar una venta de un medicamento, a crédito o en efectivo, si el número de unidades en stock de ese medicamento es inferior a la cantidad que se pretende vender.