```
-- • Carga la BD Chollolandia
use chollolandia;
-- • En todos los ejercicios se debe indicar como se utiliza dicha rutina. En caso de no indicarlo, la puntuación del ejercicio se verá perjudicada
-- • TODOS LOS EJERCICIOS PUNTÚAN 1 punto
describe clientes;
describe productos;
describe carrito;
describe vendedores;



-- 1. Crea un procedimiento llamado: nivel_Cliente. El cual reciba por parámetro el nombre de un cliente y sus apellidos
-- y en función del número de pedidos realizados, se guardará en una variable de salida el nivel en el que se encuentra el cliente:
-- ◦ Pedidos =0 – AMEBA
-- ◦ Pedidos <=2 – CHAMPIÑON
-- ◦ Pedidos <=5 – DODO
-- ◦ Pedidos <=10 – IGUANODON
-- ◦ Pedidos <=15 – CARNOTAURUS
-- ◦ Pedidos >=16 –TITANOSAURIO
drop procedure nivel_cliente;
delimiter $$
create procedure nivel_cliente(in inNombre varchar(50), inApellidos varchar(200), out outNivel varchar(20))
begin
        declare dniClienteAux char (9) default (select dniCliente from clientes where nombre = inNombre and apellidos = inApellidos);
        declare numPedidos int default (select count(*) from pedidos where dniCliente = dniClienteAux);
   if dniClienteAux is null then
                signal sqlstate '45000'
      set message_text = 'El cliente insertado no existe';
   end if;
   if numPedidos = 0 then
                set outNivel = 'AMEBA';
   elseif numPedidos <= 2 then
                set outNivel = 'CHAMPIÑON';
   elseif numPedidos <= 5 then
                set outNivel = 'DODO';
   elseif numPedidos <= 10 then
                set outNivel = 'IGUANODON';
   elseif numPedidos <= 15 then
                set outNivel = 'CARNOTAURUS';
   elseif numPedidos <= 16 then
                set outNivel = 'TITANOSAURIO';
   end if;
end $$
delimiter ;

call nivel_cliente('Cayo','Martínez', @nivel1);
```

```
call nivel_cliente('Ca','Martín', @nivel1);
select @nivel1;

-- 2. Realiza un procedimiento llamado: calcular_total_compra que reciba como parámetro
-- de entrada el nombre del cliente y sus apellidos y como salida el importe total de la posible
compra.
-- NOTAS:
-- • Hay que añadir el IGIC (7%)
-- • NO EXISTE UNA TABLA COMPRAS
-- usa handler
-- multiplicar

drop procedure calcular_total_compra;
delimiter $$
create procedure calcular_total_compra(in inNombre varchar(50), inApellidos varchar(200), out
outImporte int)
begin
        declare dniClienteAux char (9) default (select dniCliente from clientes where nombre =
inNombre and apellidos = inApellidos);
   if dniClienteAux is null then
                signal sqlstate '45000'
      set message_text = 'El cliente insertado no existe';
   end if;

  drop temporary table if exists compra_tot;
  create temporary table if not exists compra_tot
   select carrito.dniCliente, carrito.id_producto, carrito.cantidadProducto, productos.precio,
(carrito.cantidadProducto * productos.precio) as 'coste_total'
                from carrito join productos on carrito.id_producto = productos.id_producto where
carrito.dniCliente = dniClienteAux;
   set outImporte = ((select sum(coste_total) from compra_tot) * 1.07);
end $$
delimiter ;
select * from compra_tot;
call calcular_total_compra('Abundio','García', @total1);
call calcular_total_compra('Cayo','Martínez', @total1);
call calcular_total_compra('Petra','Blanco', @total1);
call calcular_total_compra('Narciso','Navarro', @total1);
select @total1;


-- 3. Crea un procedimiento almacenado llamado: actualizacion_carrito, que reciba por parámetro el
nombre de un producto y
-- el nombre de un cliente y la cantidad que desea comprar. Posteriormente, realice los siguientes
pasos:
-- • Inicie una transacción
-- • Compruebe el stock del producto. Si no es suficiente se cancelará la compra, en caso de haber
suficiente stock, se decrementará.
-- • Añade el producto al carrito del cliente
-- • En el caso que se pueda producir un error tratarlo, en caso contrario, confirmar la transacción.
drop procedure actualizacion_carrito;
delimiter $$
```

```
create procedure actualizacion_carrito(in inNombrePro varchar(150), inNombreCli varchar(50),
inCantidad int)
begin
	declare enStock int default (select stock from productos where nombre = inNombrePro);
	declare dniClienteAux char (9) default (select dniCliente from clientes where nombre =
inNombreCli);
   declare codProductoAux int default (select id_producto from productos where nombre =
inNombrePro);
   declare stockRestar int default (0);
   if (dniClienteAux is null) then
		signal sqlstate '45000'
      set message_text = 'El cliente insertado no existe';
	    end if;
	  if (codProductoAux is null) then
		signal sqlstate '45000'
      set message_text = 'El producto insertado no existe';
   end if;
	start transaction;
		 if (enStock < inCantidad) then
			signal sqlstate '45000'
	  set message_text = 'No se dispone de esa cantidad para ese producto en el stock';
		end if;
      set stockRestar = enStock - inCantidad;
      update productos set stock = stockRestar where nombre = inNombrePro;
      update carrito set cantidadProducto = inCantidad where dniCLiente = dniClienteAux and
id_producto = codProductoAux;
      commit;
end $$
delimiter ;

call actualizacion_carrito('Honda Shadow VT750C','Abundio',1);
call actualizacion_carrito('Nike Air Force','Abundio',1);

-- 4. Crea un procedimiento crear_pedido que reciba por parámetro el dni del cliente.
-- Posteriormente, realizará el pedido y lo establecerá con el estado: ‘En tránsito’. La fecha tomará
el valor actual
-- NOTAS:
-- • La función debe funcionar para cualquier día del año y para cualquier año.
-- • El total se calculará en función de la cantidadProducto establecida en carrito y el precio de
dichos productos. El ejercicio 2 te puede ayudar.
-- • Si se produce algún error se debe manejar. Ante el error, no se realizará el pedido.
drop procedure crear_pedido;
delimiter $$
create procedure crear_pedido(in inDniCli char(9))
begin
	declare totalPedido int;

	drop temporary table if exists compra_tot;
	create temporary table if not exists compra_tot
	select carrito.dniCliente, carrito.id_producto, carrito.cantidadProducto, productos.precio,
(carrito.cantidadProducto * productos.precio) as 'coste_total'
```

```
		from carrito join productos on carrito.id_producto = productos.id_producto where
carrito.dniCliente = inDniCli;
	set totalPedido = ((select sum(coste_total) from compra_tot) * 1.07);
	insert into pedidos values(default, totalPedido,curdate(), 'En transito');
end $$
delimiter ;
call crear_pedido('00000000A');

-- 5. Realiza una función llamada misProductos que reciba como parámetro el nombre de un
vendedor y muestre los productos que tiene a la venta en formato JSON.
drop function misProductos;
delimiter $$
create function misProductos(inNombreVendedor varchar(150)) returns json deterministic
begin
return (select json_arrayagg(json_object('Nombre', productos.nombre, 'descripcion', descripcion))
		from productos join vendedores on productos.id_vendedor =
vendedores.id_vendedor where vendedores.id_vendedor =
	(select id_vendedor from vendedores where nombre = inNombreVendedor));
end $$
delimiter ;

select misProductos('Gloria Sánchez');


-- 6. Realiza una función llamada anios_vendiendo que permita calcular los años que lleva un
vendedor dado de alta en la aplicación.
-- Se facilitará el id del vendedor. Utiliza dicha función para mostrar el nombre de todos los
vendedores y los años que llevan trabajando.
drop function anios_vendiendo;
delimiter $$
create function anios_vendiendo(inIdVendedor varchar(150)) returns int deterministic
begin
	declare fechaAux int default (select year(fecha_alta) from vendedores where id_vendedor =
inIdVendedor);
	declare fechaAux2 int default (year(curdate()));
	return fechaAux2 - fechaAux;

end $$
delimiter ;
select anios_vendiendo(1);
select anios_vendiendo(id_vendedor) from vendedores;
-- 7. Realiza una función llamada isPerfect que reciba un número entero y devuelva y devuelva 1 si
es perfecto o 0 si no lo es.

drop function isPerfect;
delimiter $$
create function isPerfect(inNumero int) returns int deterministic
begin
	declare numAux int default inNumero;
   declare sumaAux int default 0;
	if inNumero <= 0 then
		signal sqlstate '45000'
```

```
        set message_text = 'EL numero debe ser mayor a 0';
    end if;
        while numAux > 0 do
                set numAux = numAux - 1;
        if inNumero%numAux = 0 then
                        set sumaAux = sumaAux + numAux;
                end if;
    end while;
        if sumaAux = inNumero then
                return 1;
        else
                return 0;
        end if;
end $$
delimiter ;

select isPerfect(-1);
select isPerfect(6);
-- EXPLICACIÓN DE UN NÚMERO PERFECTO:
-- • Un número perfecto es un número entero positivo que es igual a la suma de sus divisores propios positivos (excluyendo él mismo).
-- ◦ Un divisor de un número entero es otro número entero que puede dividir al número original exactamente, es decir, sin resto.
-- • Por ejemplo, 28 es perfecto (la función debería retornar 1), ya que sus divisores son 1,2,4,7 y 14 y su suma es igual a 28.
-- • Si la función recibe un 0 o un número negativo devolverá 0.
-- • Si la función recibe un 12 devolverá 0, ya que no es perfecto.
-- • Te facilito un listado de números perfectos para realizar pruebas. NO PUEDES UTILIZAR ESTE LISTADO PARA COMPROBAR DIRECTAMENTE EL NÚMERO:
--   ◦ 6 (1 + 2 + 3)
--   ◦ 28 (1 + 2 + 4 + 7 + 14)
--   ◦ 496 (1 + 2 + 4 + 8 + 16 + 31 + 62 + 124 + 248)
--   ◦ 8128 (1 + 2 + 4 + 8 + 16 + 32 + 64 + 127 + 254 + 508 + 1016 + 2032 + 4064)
--   ◦ 130816 (1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 + 256 + 512 + 1024 + 2048 + 4096 + 8192 + 16384 + 32768 + 65536 + 131072 + 262144 + 524288)



-- 8. Crea un trigger que compruebe cuando se modifica el stock de un producto.
-- Se deberá comprobar si es mayor o igual a 0. Si se intenta establecer otro valor, se pondrá su valor a 0.

drop trigger producto_BU_trigger;
delimiter $$
create trigger producto_BU_trigger before update
        on productos for each row
begin
        if new.stock < 0 then
                set new.stock = 0;
        end if;
end $$
delimiter ;
```

-- 9. Añade a la tabla clientes un campo llamado nombreUsuario tipo varchar(25).
-- Posteriormente, realiza un trigger que genere el nombre de usuario de un cliente, cada vez que se registre.
-- El nombre de usuario estará compuesto por:
-- • Las dos primeras letras de su nombre (en minúscula)
-- • La primera letra de su apellido (en mayúscula)
-- • La letra de su DNI (en mayúscula)
-- • Su fecha de nacimiento
-- (Si no introduce la fecha de nacimiento se deberá mostrar un mensaje de error, impidiendo que se inserte dicho cliente).

```
alter table clientes add nombreUsuario varchar(50);
drop trigger clientes_BI_trigger;
delimiter $$
create trigger clientes_BI_trigger before INSERT
        on clientes for each row
begin
--      declare exit handler for 1136
        declare nomAux char(3) default lower((select substring(new.nombre,1,2)));
        declare apeAux char(1) default upper((select substring(new.apellidos,1,1) ));
        declare dniAux char(1) default upper((select substring(new.dniCLiente,9,9) ));
        declare fechaAux varchar(10) default (select new.fecha_nacimiento);

   if new.fecha_nacimiento is null then
                signal sqlstate '45000'
      set message_text = 'Debe introducir fecha de nacimiento';
   end if;
   set new.nombreUsuario = concat(nomAux,apeAux,dniAux,fechaAux);
end $$
delimiter ;

insert into clientes (dniCliente, nombre, apellidos, email, passwd, direccion, fecha_nacimiento)
        values('12345678M', 'Josef', 'Joestar', 'josito4@hot', 'passwo', 'Callejon', '1960-01-01');
insert into clientes (dniCliente, nombre, apellidos, email, passwd, direccion, fecha_nacimiento)
        values('11223344d', 'KENY', 'ARTON', 'KENCHAN3@hot', 'passwoRR', 'Callejon', '1960-01-01');
```

-- 10. Realiza un trigger para que cada vez que se genere un pedido (nuevo). Se realice un descuento al pedido en función del nivel del usuario (ejercicio 2).
-- Los descuentos serán:
-- ◦ AMEBA -> 5% menos
-- ◦ CHAMPIÑON -> 10% menos
-- ◦ DODO -> 12% menos
-- ◦ IGUANODON-> 20% menos
-- ◦ CARNOTAURUS -> 30% menos
-- ◦ TITANOSAURIO -> 70% menos