

# JARDINERÍA

CHRISTIAN MILLÁN SORIA

1º DAW TARDE

## 1. Mostrar el nombre de un cliente dado su código.

```
drop procedure if exists nombre_cliente;
delimiter //
create procedure nombre_cliente(codigo int)
begin
select nombre_cliente, codigo_cliente from cliente where codigo_cliente=codigo;
end//
delimiter ;
call nombre_cliente(1);
```

nombre_cliente	codigo_cliente
GoldFish Garden	1

## 2. Mostrar el precioVenta y la gama de un producto dado su código.

```
drop procedure if exists precio_venta;
delimiter //
create procedure precio_venta(codigo int)
begin
select precio_venta from producto where codigo_producto=codigo;
end//
delimiter ;
call precio_venta(21636);
```

precio_venta
14.00

## 3. Mostrar toda la información de un pedido dado su código (fechaEsperada, fechaEntrega, fechaPedido, estado y comentarios).

```
drop procedure if exists info_pedido;
delimiter //
create procedure info_pedido(codigo int)
begin
select * from pedido where codigo_pedido=codigo;
end//
```


```
delimiter ;  
call info_pedido(1);
```

codigo_pedido	fecha_pedido	fecha_esperada	fecha_entrega	estado	comentarios	codigo_cliente
1	2006-01-17	2006-01-19	2006-01-19	Entregado	Pagado a plazos	5


**4. Realizar una función que me devuelva la suma de pagos que ha realizado un cliente. Pasa el código por parámetro.**

```
drop procedure if exists suma_pagos_cliente;  
delimiter //  
create procedure suma_pago_cliente(codigo int)  
begin  
select sum(total) from pago inner join cliente on  
pago.codigo_cliente=cliente.codigo_cliente where codigo_cliente=codigo;  
end//  
delimiter ;  
call suma_pago_cliente(1);
```

## Error

SQL query: [Copy](#) 

```
create procedure suma_pago_cliente(codigo int)  
begin  
select sum(total) from pago inner join cliente on pago.codigo_cliente=cliente.codigo_cliente where codigo_cliente=codigo;  
end;
```

MySQL said: 

#1304 - PROCEDURE suma\_pago\_cliente already exists

**5. Realizar un método o procedimiento que muestre el total en euros de un pedido. Pasa el código por parámetro.**

```
drop procedure if exists total_euros;  
delimiter //  
create procedure total_euros(codigo int)  
begin  
select  
end//  
delimiter ;  
call total_euros();
```



**6. Mostrar el nombre de un cliente dado su código. Controla en caso de que no se encuentre, mostrando un mensaje por ejemplo.**

```
drop procedure if exists nombre_cliente_controlado;
delimiter //
create procedure nombre_cliente_controlado(codigo int)
begin
if not exists(select 1 from cliente where codigo_cliente=codigo) then
select 'Error: El cliente no existe.' as mensaje;
else
select nombre_cliente from cliente where codigo_cliente=codigo;
end if;
end //
delimiter ;
call nombre_cliente_controlado(2);
```

mensaje

Error: El cliente no existe.

**7. Realizar una función que me devuelva la suma de pagos que ha realizado un cliente. Pasa el código por parámetro. Controla en caso de que no se encuentre, en ese caso devuelve un -1.**

```
drop procedure if exists ;
delimiter //
create procedure ()
begin

end//
delimiter ;
call ();
```



**8. Realizar un método o procedimiento que muestre el total en euros de un pedido. Pasa el código por parámetro. Controla en caso de que no se encuentre, devolviendo un 0. Pasa otro parámetro como límite, si lo supera, se lanza una excepción propia y devuelve un 0.**

```
drop procedure if exists ;
delimiter //
create procedure ()
begin

end//
```

```
delimiter ;  
call ();
```



**9. Realiza un resumen con informe de las estadísticas de los pedidos realizados por meses y por años.**

```
drop procedure if exists ;  
delimiter //  
create procedure ()  
begin  
  
end//  
delimiter ;  
call ();
```

