```
-- 1. Crear una función con el nombre ejemploXX y sin parámetros de entrada. AL consultar la función esta debe imprimir en pantalla el siguiente texto: Hola minombreXX¡
-- (Donde pone XX debes poner tu número de prácticas y donde pone minombre debes poner tu nombre).
drop function if exists ejemplo18;
delimiter $$
create function ejemplo18() returns char(13) deterministic
begin
	return 'Hola Daniel18';
end $$
delimiter ;
select ejemplo18();

-- 2. Ejecutar la siguiente sentencia: SELECT ejemploXX() AS ‘Saludo’;
-- Crear una función con el nombre saludito que reciba una cadena de caracteres (cad_ent) y devuelve el siguiente mensaje:
-- ‘Chacho (cad_ent)¡ Que pasó?’
select ejemplo18() as 'Saludo';

drop function if exists saludito;
delimiter $$
create function saludito(cad_ent varchar(20)) returns varchar(100) deterministic
begin
	declare mensaje varchar(100) default concat('Chacho ', cad_ent, '! Que pasó!');
	return mensaje;
end $$
delimiter ;

-- 3. Llamar a la función anterior desde un SELECT usando la frase.
select saludito('Daniel');
-- 4. Crear una función con el nombre frase(). Esta función recibirá una cadena (frase_entrada) y devolverá el siguiente texto:
-- 'La frase: (frase_entrada) es muy larga'.
-- Si frase_entrada tiene más de 40 caracteres.
-- Si por le contrario frase_entrada tiene 40 o menos caracteres devolverá: 'La frase: (frase_entrada) es muy corta'

drop function if exists frase;
delimiter $$
create function frase(frase_entrada varchar(100)) returns varchar(100) deterministic
begin
	declare mensaje varchar(100);
	if length(frase_entrada) > 40 then
		set mensaje = concat('La frase (', frase_entrada, ') es muy larga');
	else
		set mensaje = concat('La frase (', frase_entrada, ') es muy corta');
	end if;
   return mensaje;
end $$
delimiter ;

-- 5. Llamar a la función anterior desde un SELECT insertando una frase corta primero y una frase
```

```
larga después.
select frase('12');
select frase('12345678912345678912345678912345678912345');
SELECT length('12345678912345678912345678912345678912345');
-- 6. Crear una función que determine si un número es par o no.
drop function if exists par;
delimiter $$
create function par(numero int) returns char(9) deterministic
begin
	declare mensaje varchar(100);
	if numero%2 = 1 then
		set mensaje = 'No es par';
	else
		set mensaje = 'Es par';
	end if;
   return mensaje;
end $$
delimiter ;

-- 7. Llamar a la función anterior desde un SELECT con números pares e impares.
select par(2);
select par(5);
-- 8. Crear una función de nombre peli_getName(cod INT) que en base a un código (número entero)
de película
-- devuelva su nombre o la cadena 'DESCONOCIDO' en caso que no exista.
use filmStore;
describe peliculas;
drop function if exists peli_getName;
delimiter $$
create function peli_getName(inCod int) returns varchar(100) deterministic
begin
--	declare mensaje varchar(100);
	declare nombreAux varchar(40) default (select nombre from peliculas where codPelicula =
inCod);
	if nombreAux is null then
		return 'DESCONOCIDO';
	else
		return nombreAux;
	end if;
end $$
delimiter ;

-- 9. Llamar a la función anterior desde un SELECT con distintos valores.
select peli_getName(0);
select peli_getName(1);
select peli_getName(2);
select peli_getName(3);

-- 10.  Crear una función de nombre peli_getMaxPrecio(num int) que en base a un DNI de cliente
devuelva
-- el nombre de la película más cara que ha comprado (utiliza la función de agregado
MAX(columna)).
```

```
use filmStore;
describe clientes;
describe peliculas;
drop function if exists peli_getMaxPrecio;
delimiter $$
create function peli_getMaxPrecio(inDni char(9)) returns varchar(40) deterministic
begin
	declare codPeliAUx int;
 select codPelicula from peliculas where codPelicula in
	(select codPelicula from compras where dnicliente  = inDni ) order by precio desc limit 1
into codPeliAux;
		return (select peliculas.nombre from peliculas where codPelicula = codPeliAux);
end $$
delimiter ;

-- 11.  Llamar a la función anterior desde un SELECT con distintos valores.
select peli_getMaxPrecio('45525540A');
select peli_getMaxPrecio('48532544T');

-- 12.  Crea una función de nombre lastyear() que en base al código de una película devuelva
cuantas veces ha sido compradas en el último año.
drop function if exists lastyear;
delimiter $$
create function lastyear(inCodPeli int) returns int deterministic
begin
--	return (select count(*) from compras where codPelicula = inCodPeli and fechaCompra
between(curdate() and ));
	return (select count(*) from compras where codPelicula = inCodPeli and year(fechaCompra)
>= (year(curdate()) -1 ));
end $$
delimiter ;

-- 13.  Llamar a la función anterior desde un SELECT con distintos códigos de películas.
select lastyear(20);
select curdate();
select curdate() - INTERVAL 365 DAY;

-- 14.  AVANZADO: Crear una función llamada factorial que le pasemos como parámetro un
número
-- y devuelva el factorial de dicho número. Llamar a la función desde un SELECT con distintos
valores.
drop function if exists factorial;
delimiter $$
create function factorial(inNumero int) returns int deterministic
begin
	declare acumulado int default 1;
	while inNumero > 1 do
		set acumulado = inNUmero * acumulado;
      set inNumero = inNumero - 1;
	end while;
   return acumulado;
end $$
```

```
delimiter ;
select factorial(12);
```