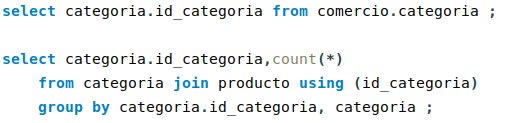
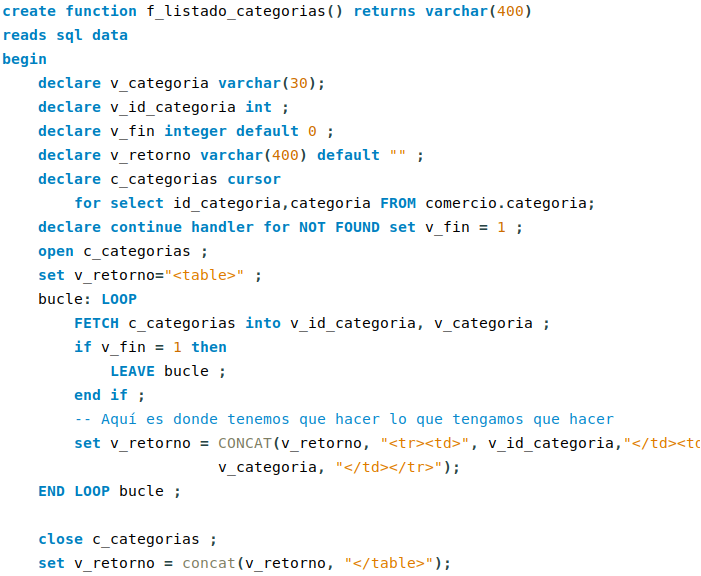
1. **Crea una función f\_listado\_categorias() que devuelva varchar(600) con el código HTML de una tabla con dos columnas: código y categoría, con los datos que aparecen en la base de datos.**







TABLA

**3. Nuestro comercio ha ganado mucho dinero y no sabe qué hacer con él. Por lo que ha decidido repartir una gratificación a los empleados, usando el siguiente criterio:  
   
 \* Un 10% sobre las ventas realizadas a los empleados que han vendido más que el promedio de ventas realizadas por empleado**

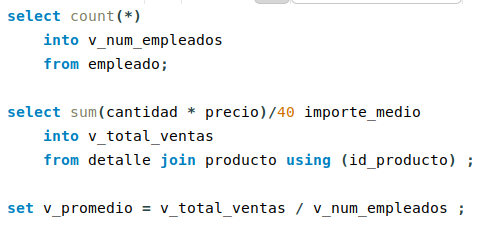
**\* Un 5% sobre las ventas realizadas a los empleados que NO han vendido más que el promedio de ventas realizadas por empleado  
  
Para ello, hay que crear dos nuevas columnas, llamadas:**

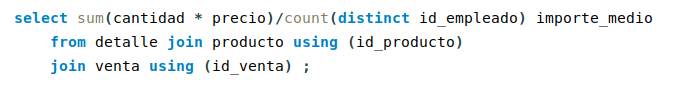
* **gratificación: importe de la gratificación**
* **pct\_gratificacion : porcentaje (5 ó 10)**

PACTO GRATIFICACION

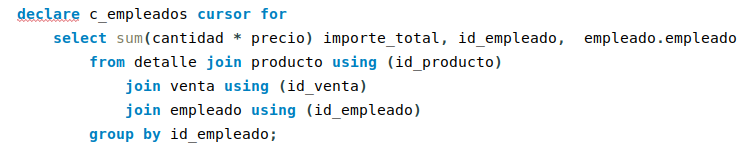
**PASOS PARA REALIZARLO:**

* *Calcular el promedio de ventas por empleado*

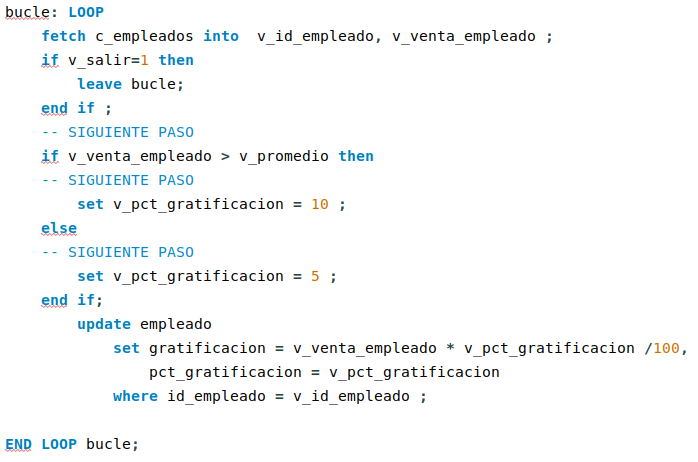




* *Consultar los empleados, con el importe de ventas que ha realizado*



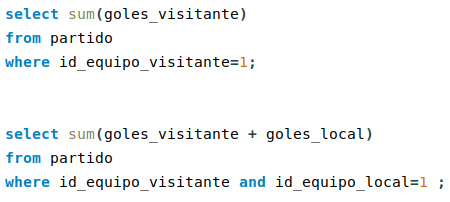
* Para cada empleado:
  + ¿El importe de ventas del empleado > promedio?
    - SI: Aplicar un 10%
    - NO: Aplicar un 5%



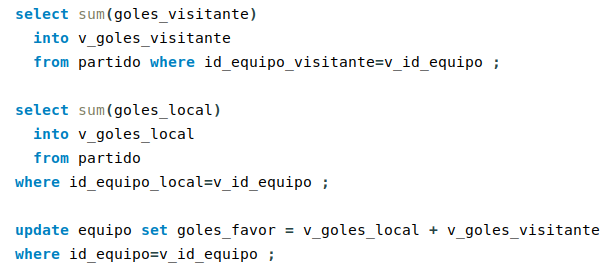
EQUIPOS DE FÚTBOL:

**Crear un procedimiento denominado sp\_actualizar\_puntuacion que, usando un cursor, actualice las siguientes columnas de la tabla equipo:**

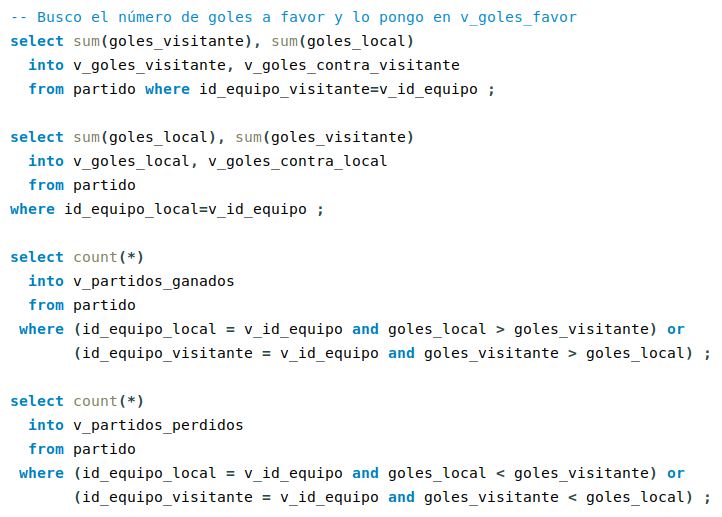
* **goles\_favor**
* **goles\_contra**



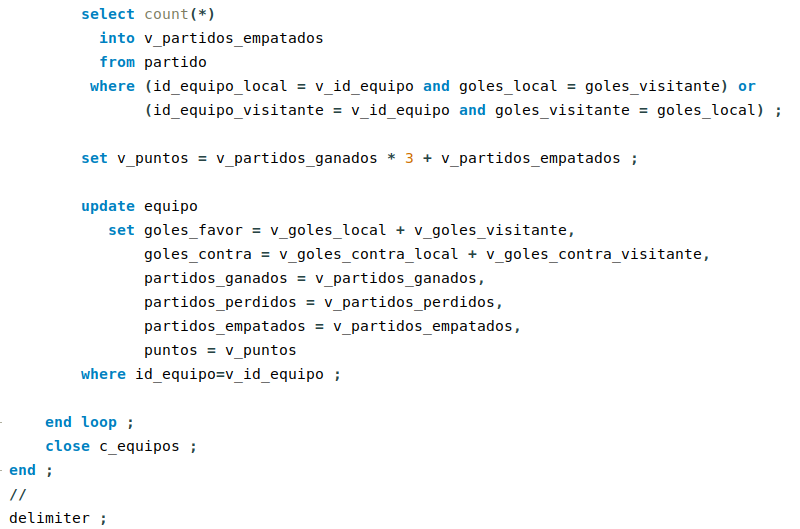
Dentro del procedimiento:



* **partidos\_ganados**
* **partidos\_empatados**
* **partidos\_perdidos**



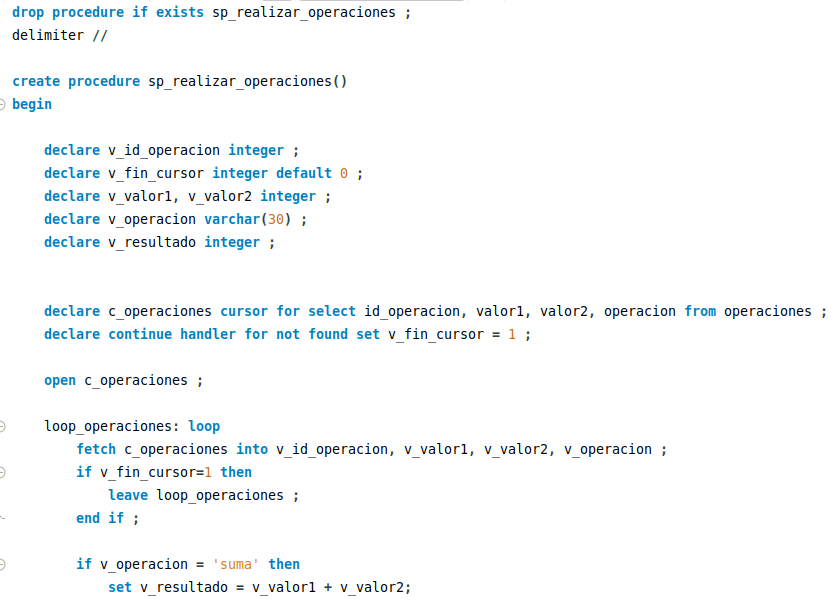
* **puntos: 3 puntos por cada partido ganado, 1 punto por cada partido empatado, 0 puntos por partido perdido**

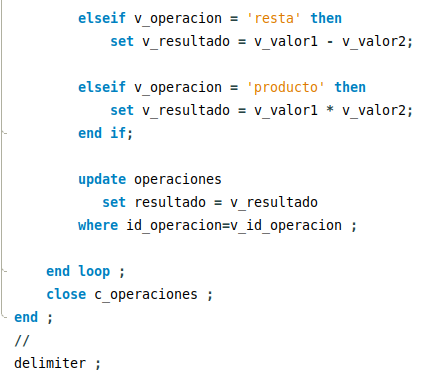


**Crea un procedimiento llamado: sp\_realizar\_operaciones que, mediante un cursor, recorra todas las operaciones de la tabla y realice lo siguiente:**

* **Si operación="suma", hace que resultado=valor1 + valor2**
* **Si operación="resta", hace que resultado=valor1 - valor2**
* **Si operación="producto", hace que resultado=valor1 \* valor2**

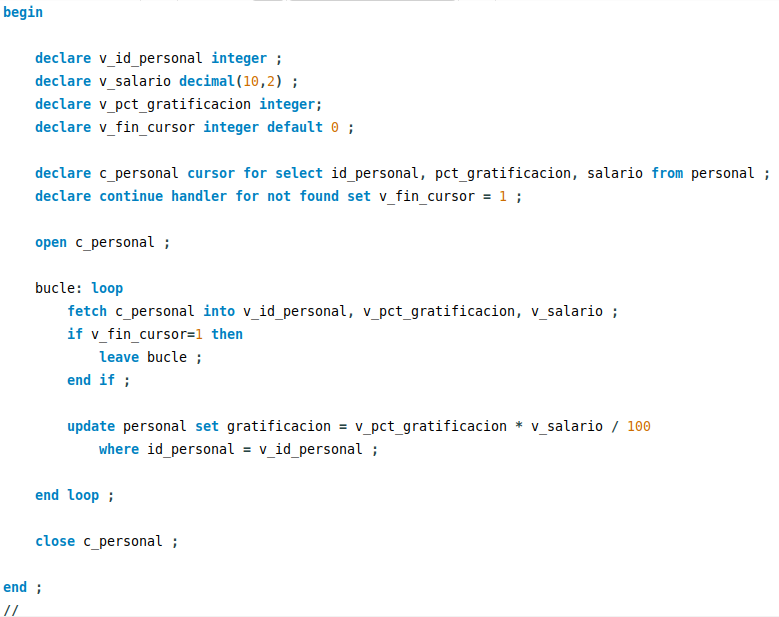
**OPERACION**





**Se crea una tabla de nombre personal con las columnas: id\_personal, nombre, salario, pct\_gratificacion, gratificacion**

**Mediante un cursor, rellena la columna gratificacion usando el pct\_gratificacion sobre el salario (porcentaje de gratificación sobre el salario)**

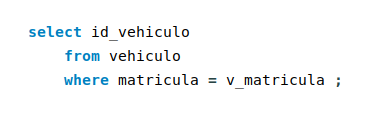


PORCENTAJE DE SALARIO

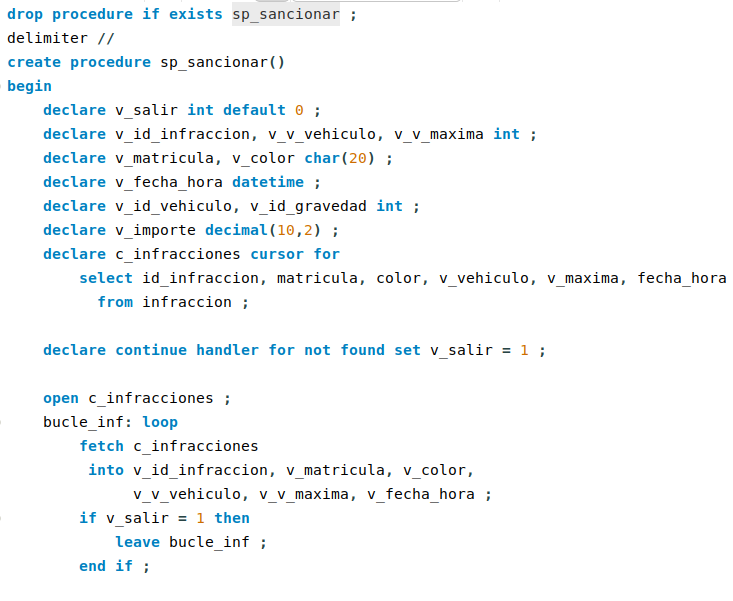
**Finalmente, debes crear un procedimiento que realice lo siguiente:**

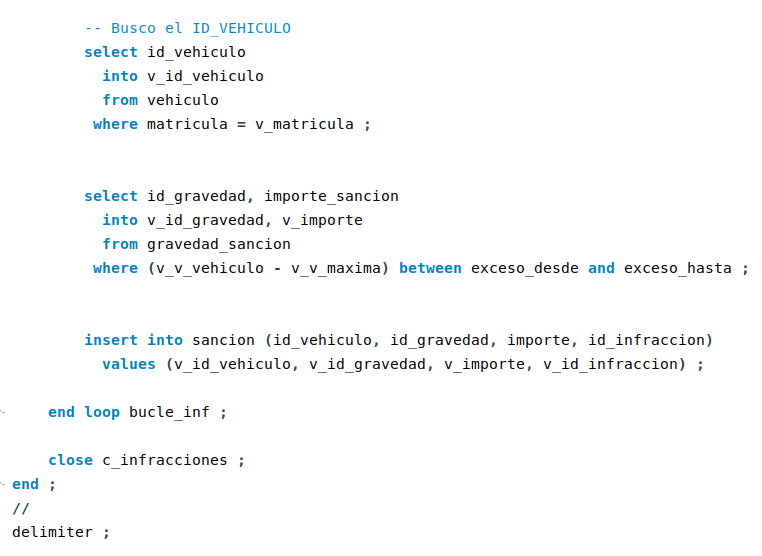
**Mediante un cursor, recorra todas las infracciones**

* **Localiza el id\_vehiculo, usando la matrícula**



* **Calcula el exceso de velocidad del vehículo (v\_vehiculo - v\_maxima) y busca la gravedad de la infracción y el importe de la sanción**
* **Inserta una SANCIÓN con los datos encontrados**
* **Actualiza la información del PROPIETARIO, añadiendo uno al NUMERO\_SANCIONES y la cantidad adecuada al IMPORTE\_TOTAL\_SANCIONES**





**El procedimiento a crear se denomina sp\_generar\_recibos() y realiza lo siguiente:**

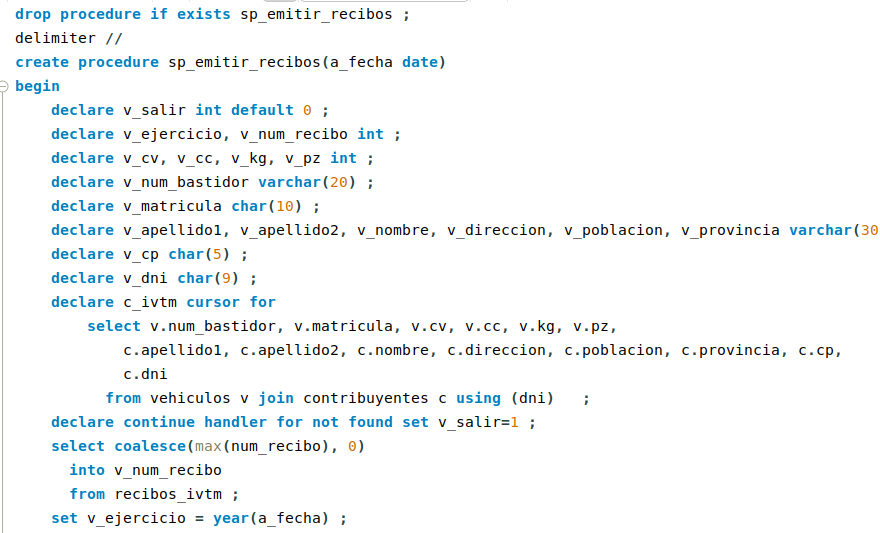
**Mediante un cursor, selecciona todos los clientes Inserta una fila en la tabla RECIBOS, teniendo en cuenta que:**

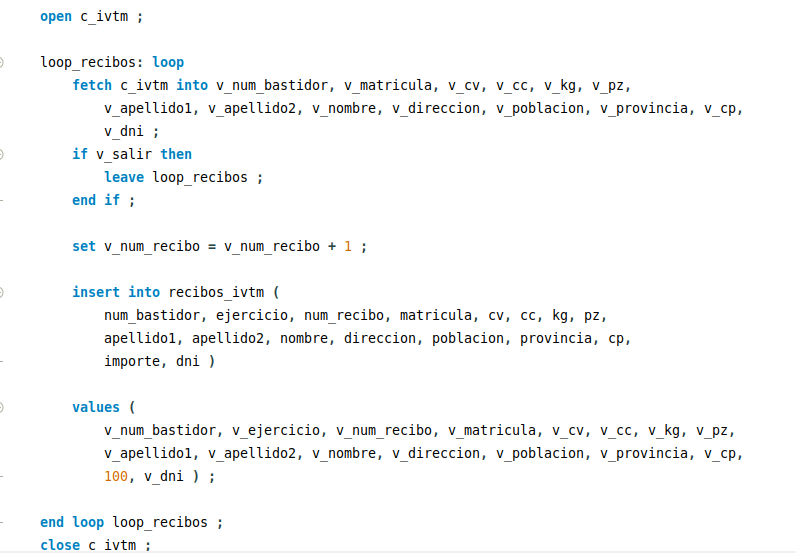
* **El importe\_PLAN es el correspondiente al PLAN del cliente  
  Si el cliente tiene algún PCT\_DESCUENTO, se aplica ese porcentaje de descuento. Se rellena en RECBO el PCT\_DESCUENTO y IMPORTE\_DESCUENTO**
* **EL Importe\_RECIBO = IMPORTE\_PLAN - IMPORTE\_DESCUENTO**

ULTIMO NUMERO

**Crear un procedimiento llamado sp\_emitir\_recibos(a\_fecha date) que recibe un argumento de tipo fecha y que realiza los siguientes pasos:**

1. **Busca el último número de recibo\_ivtm emitido**
2. **Mediante un cursor, recorre todos los vehículos e inserta en la tabla recibos\_ivtm, numerando los recibos a partir del último emitido. EL importe, será igual para todos: 100**

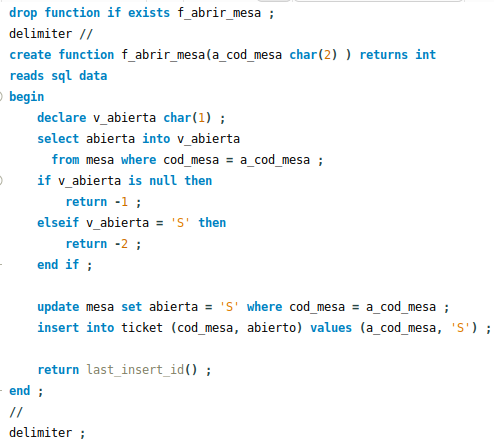




#### **Función f\_abrir\_mesa(a\_cod\_mesa char(2)) return int**

**Esta función tiene como objeto abrir la mesa. Para ello, realiza las siguientes acciones:**

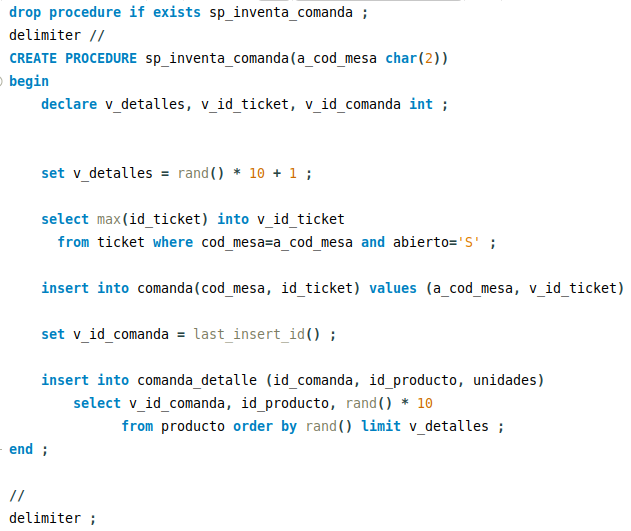
* **Comprueba que la mesa indicada como argumento exista. Si no existe, la función devuelve -1**
* **Comprueba que la mesa no esté abierta (abierta='N'). Si estuviera abierta, devuelve -2**
* **Actualiza el registro de la mesa indicado, poniendo abierta='S'**
* **Inserta un TICKET, indicando el cod\_mesa, y abierto='S'**
* **Tras insertar el TICKET, devuelve el valor de la función de sistema last\_insert\_id() que permite obtener el ID\_TICKET del nuevo ticket insertado**

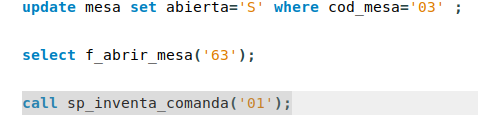


ABRIR MESA

#### **Inventar comandas**

**Cuando tengas creada la BD y también hayas creado la función f\_abrir\_mesa, utiliza este fichero, que contiene un procedimiento llamado sp\_invernta\_comanda(a\_cod\_mesa):**

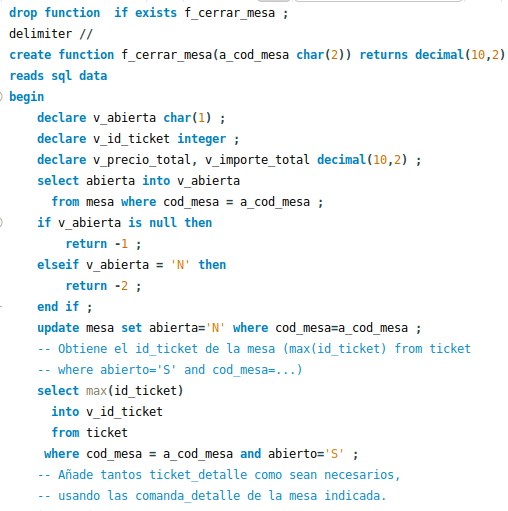


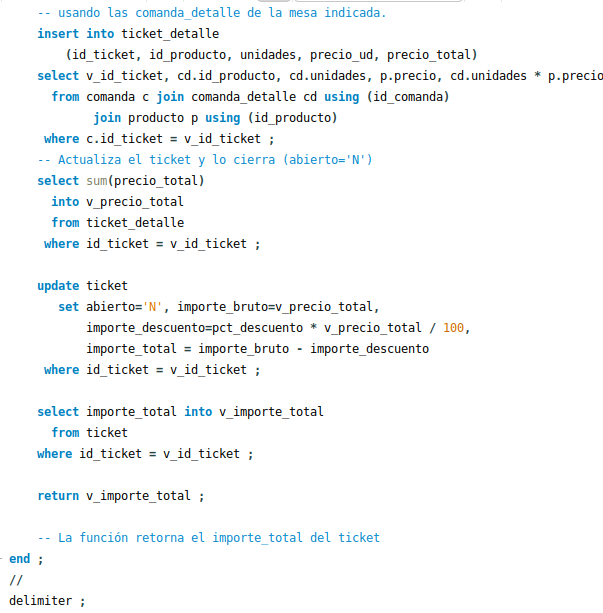


#### **Función f\_cerar\_mesa(a\_cod\_mesa char(2)) return decimal(10,2)**

**Esta función tiene como objeto cerrar la mesa. Para ello, realiza las siguientes acciones:**

* **Comprueba que la mesa indicada como argumento exista. Si no existe, la función devuelve -1**
* **Comprueba que la mesa esté abierta (abierta='S'). Si estuviera cerrada, devuelve -2**
* **Actualiza el registro de la mesa indicado, poniendo abierta='N'**
* **Obtiene el id\_ticket de la mesa (max(id\_ticket) from ticket where abierto='S' and cod\_mesa=...)**
* **Añade tantos ticket\_detalle como sean necesarios, usando las comanda\_detalle de la mesa indicada.**
* **Ten en cuenta que todos los campos de ticket\_detalle deben estar completamente rellenos, así como los importes totales del ticket.**
* **Actualiza el ticket y lo cierra (abierto='N')**
* **La función retorna el importe\_total del ticket**



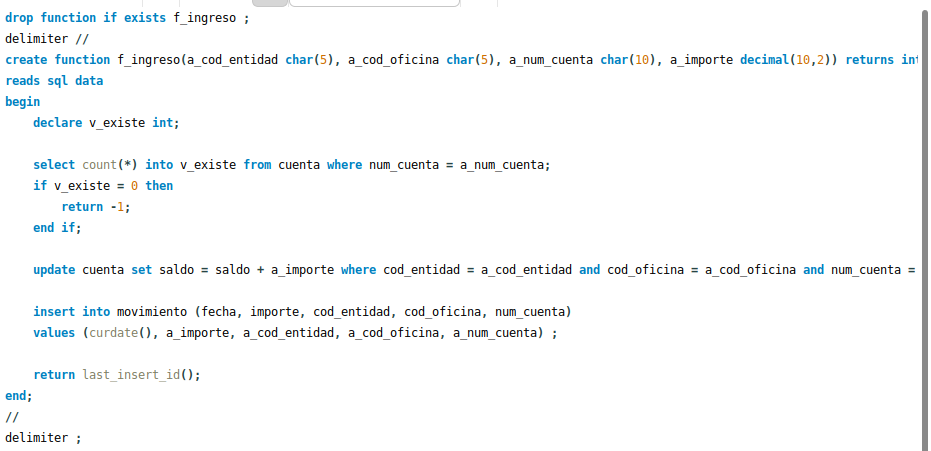


#### **Función f\_ingreso(a\_cod\_entidad, a\_cod\_oficina, a\_num\_cuenta, a\_importe)**

**La función devuelve INTEGER.**

**La función realiza las siguientes operaciones:**

* **Comprueba que existe la cuenta indicada. Si no existe, la función retorna -1**
* **Se actualiza el SALDO de la cuenta, incrementándolo en a\_importe.**
* **La función inserta una fila en la tabla MOVIMIENTO, usando la fecha actual curdate()**
* **La función retorna el ID\_MOVIMIENTO (que se obtiene usando LAST\_INSERT\_ID())**



#### 

#### 

#### 

#### 

#### 

#### 

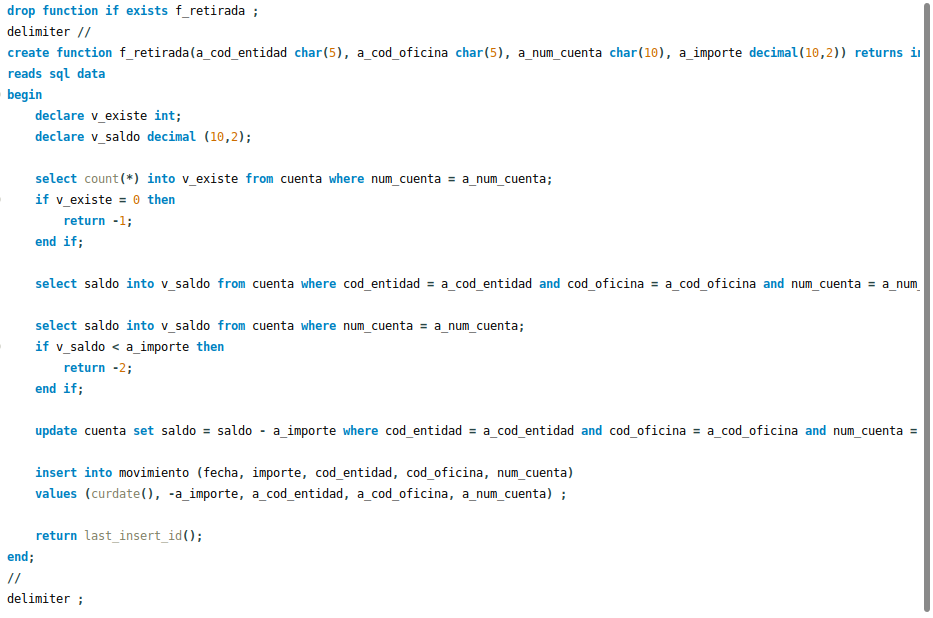
#### 

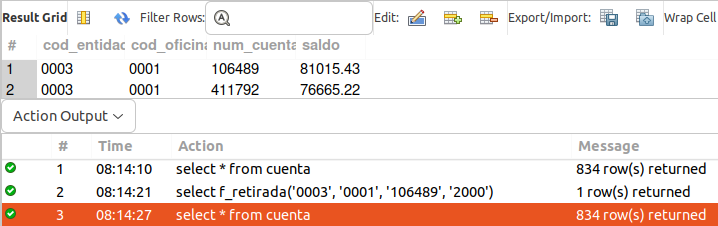
#### **Función f\_retirada(a\_cod\_entidad, a\_cod\_oficina, a\_num\_cuenta, a\_importe)**

**La función devuelve INTEGER.**

**La función realiza las siguientes operaciones:**

* **Comprueba que existe la cuenta indicada. Si no existe, la función retorna -1**
* **Comprueba que el saldo sea suficiente saldo >= a\_importe. Si no es suficiente, la función retorna -2**
* **Se actualiza el SALDO de la cuenta, disminuyendo en a\_importe.**
* **La función inserta una fila en la tabla MOVIMIENTO, usando la fecha actual curdate() y apuntando el IMPORTE que se retira en negativo.**
* **La función retorna el ID\_MOVIMIENTO (que se obtiene usando LAST\_INSERT\_ID())**

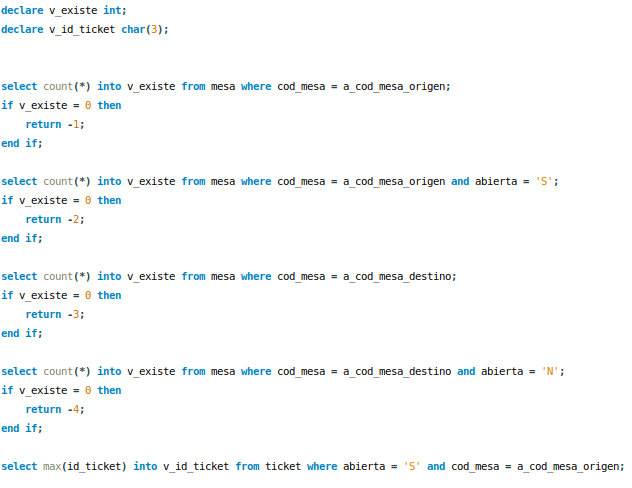




Podemos ver como le resta 2000 a su salario

#### **Función f\_cambiar\_mesa(a\_cod\_mesa\_origen, a\_cod\_mesa\_destino char(2)) return int**

* **Comprueba que la mesa\_origen indicada como argumento exista. Si no existe, la función devuelve -1**
* **Comprueba que la mesa\_origen esté abierta (abierta='S'). Si estuviera cerrada, devuelve -2**
* **Comprueba que la mesa\_destino indicada como argumento exista. Si no existe, la función devuelve -3**
* **Comprueba que la mesa\_destino no esté abierta (abierta='N'). Si estuviera abierta, devuelve -4**
* **Obtiene el id\_ticket de la mesa\_origen (max(id\_ticket) from ticket where abierto='S' and cod\_mesa=...)**
* **Cambia el cod\_mesa de las comandas de dicho ticket desde a\_cod\_mesa\_origen a a\_cod\_mesa\_destino**
* **Cambia cod\_mesa de dicho ticket desde a\_cod\_mesa\_origen a a\_cod\_mesa\_destino**
* **Cierra la mesa\_origen, cambiando abierta='S' a abierta='N'**
* **devuelve el ID\_TICKET de la mesa**



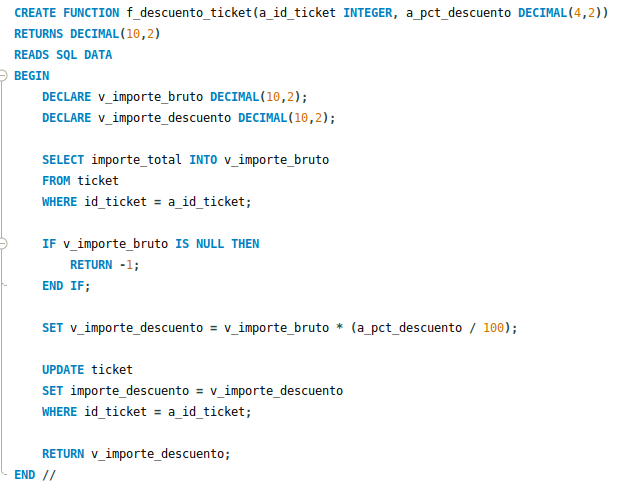
#### 

#### **Función f\_descuento\_ticket(a\_id\_ticket integer, a\_pct\_descuento decimal(4,2)) return decimal(10,2)**

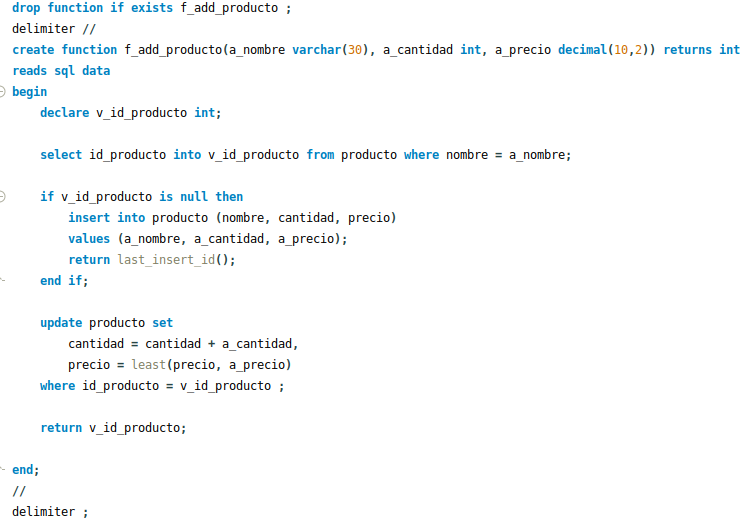
**Esta función aplica el porcentaje de descuento indicado al ticket indicado. Para ello, calcula importe\_descuento como resultado de aplicar el pct al importe bruto.**

**Comprueba que a\_id\_ticket exista, Si no existe, la función devuelve -1**

**La función devuelve el importe\_descuento de dicho ticket.**

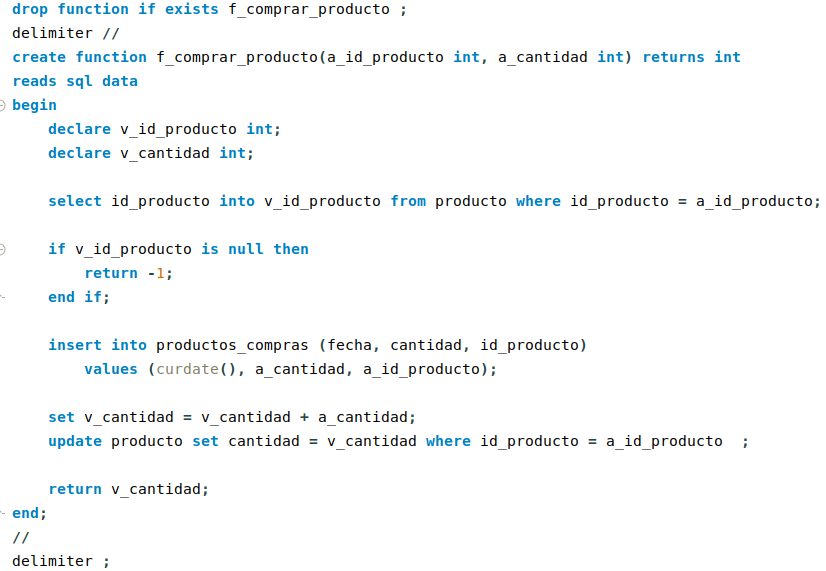


* **función f\_add\_producto(a\_nombre, a\_cantidad, a\_precio) returns int**
  + **La función comprueba si ya existe un producto con el a\_nombre indicado.**
    - **Si existe, actualiza la cantidad sumando a\_cantidad y el precio con el menor del precio existente o el indicado como argumento**
    - **SI no existe, se inserta el producto con los datos proporcionados como argumento**
    - **En ambos casos, devuelve el id\_producto (el existente o el last\_insert\_id, según corresponda)**



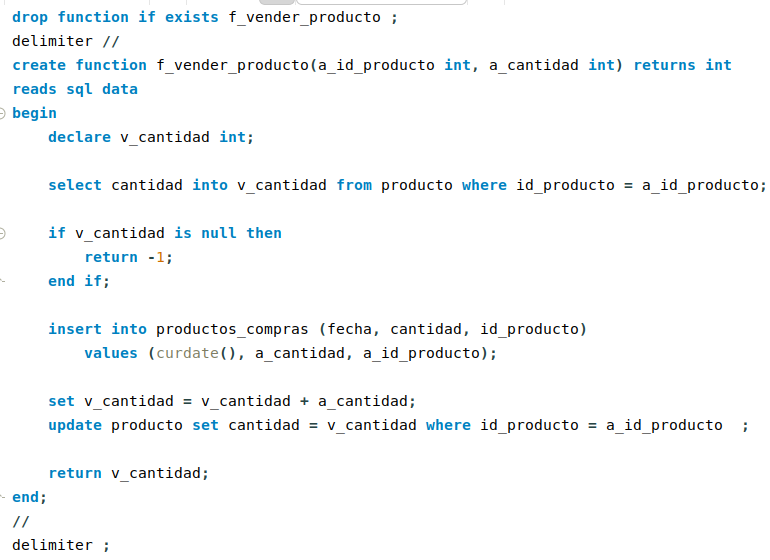


* **función f\_comprar\_producto(a\_id\_producto, a\_cantidad) returns int**
  + **La función comprueba que exista el producto indicado como argumento. Si no existe, devuelve -1**
  + **Se añade una fila en la tabla producto\_compras, usando curdate() para la fecha**
  + **Se actualiza la cantidad en producto, sumándole a\_cantidad**
  + **La función devuelve la nueva cantidad de producto**





* **función f\_vender\_producto(a\_id\_producto, a\_cantidad) returns int**
  + **La función comprueba que exista el producto indicado como argumento. Si no existe, devuelve -1**
  + **La función comprueba que la cantidad de producto indicado sea superior o igual a la indicada como argumento. Si no lo fuera, devuelve -2**
  + **Se añade una fila en la tabla producto\_ventas, usando curdate() para la fecha**
  + **Se actualiza la cantidad en producto, restándole a\_cantidad**
  + **La función devuelve la nueva cantidad de producto**

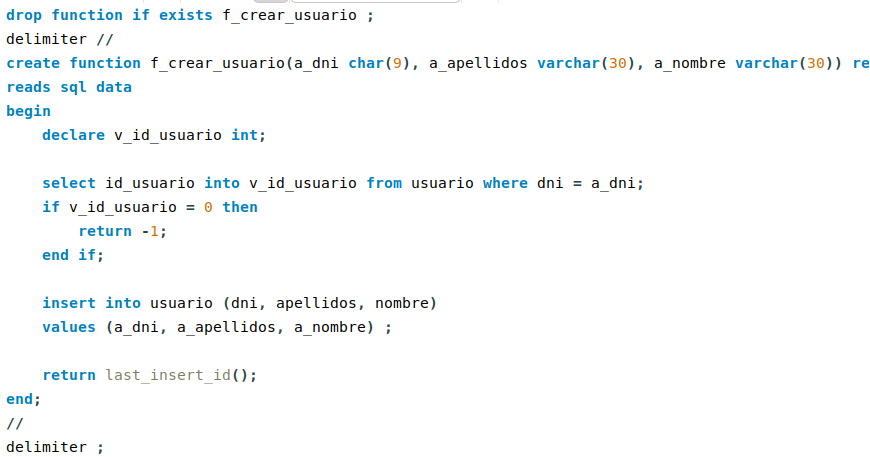


##### **BIBLIOTECA**

##### 

##### **Función f\_crear\_usuario(a\_dni, a\_apellidos, a\_nombre) returns int**

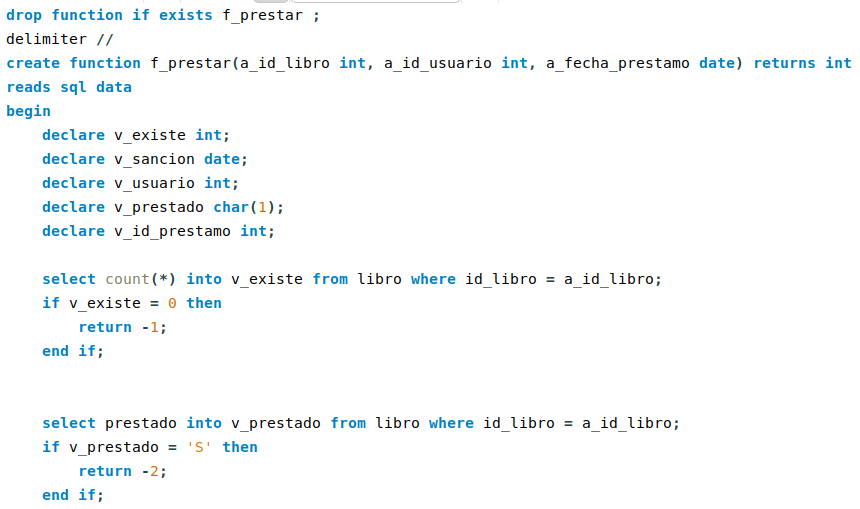
* **Comprueba si existe algún usuario con el mismo DNI. Si es así, la función devuelve -1**
* **Inserta una fila en la tabla USUARIO**
* **Devuelve el nuevo id\_usuario**

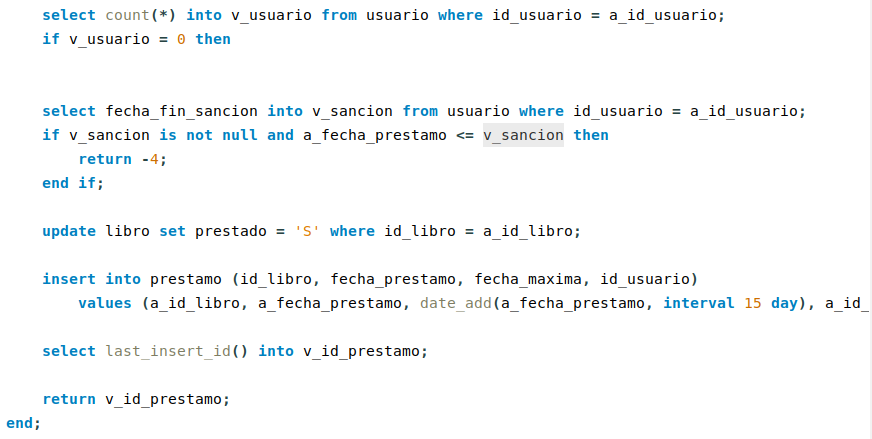


##### 

##### **Función f\_prestar(a\_id\_libro, a\_id\_usuario, a\_fecha\_prestamo) returns int**

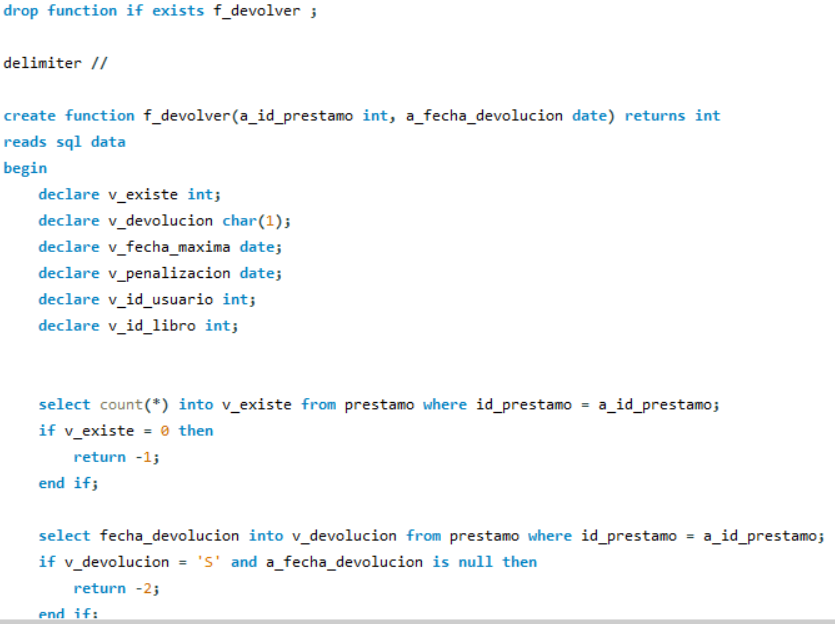
* **Comprueba que exista el libro indicado. Si no es así, devuelve -1**
* **Comprueba que el libro no está prestado (prestado='N'). Si estuviera prestado, devuelve -2**
* **Comprueba que exista el usuario indicado. Si no es así, devuelve -3**
* **Comprueba que el usuario no esté en periodo de sanción: si a\_fecha\_prestamo <= fecha\_fin\_sancion, devuelve -4**
* **Cambia el libro, poniendo prestado='S'.**
* **Inserta una fila en PRESTAMO, con fecha\_maxima = fecha\_prestamo + 15 días**
* **Devuelve id\_prestamo**

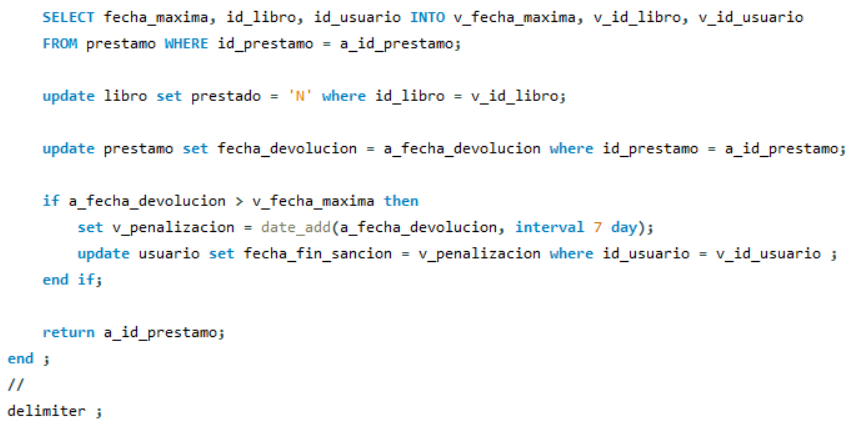




##### **Función f\_devolver(a\_id\_prestamo, a\_fecha\_devolucion) returns int**

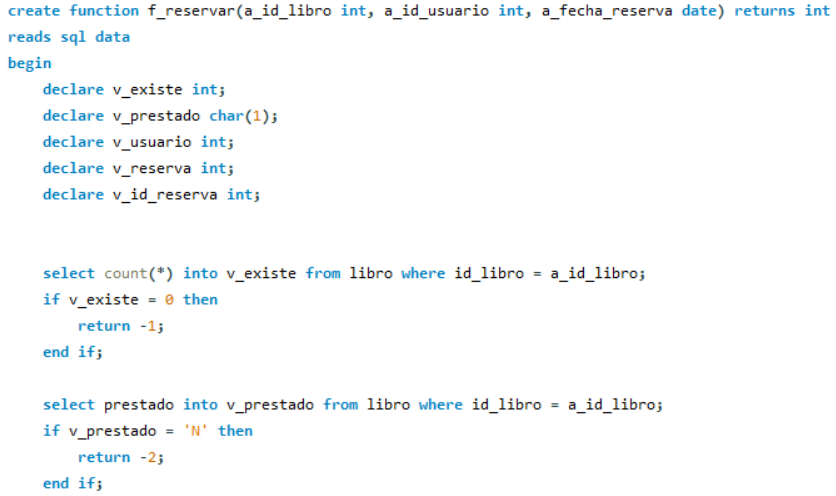
* **Comprueba que exista el préstamo indicado. Si no es así, devuelve -1**
* **Comprueba que el préstamo no esté devuelto (fecha\_devolucion is null). Si estuviera devuelto, retorna -2**
* **Actualiza el libro a prestado='N'**
* **Actualiza el préstamo con la fecha\_devolución**
* **Si fecha\_devolucion>fecha\_maxima, penaliza al usuario 1 semana sin poder coger libros (fecha\_fin\_sancion=fecha\_devolución + 7 días)**
* **Devuelve id\_prestamo**

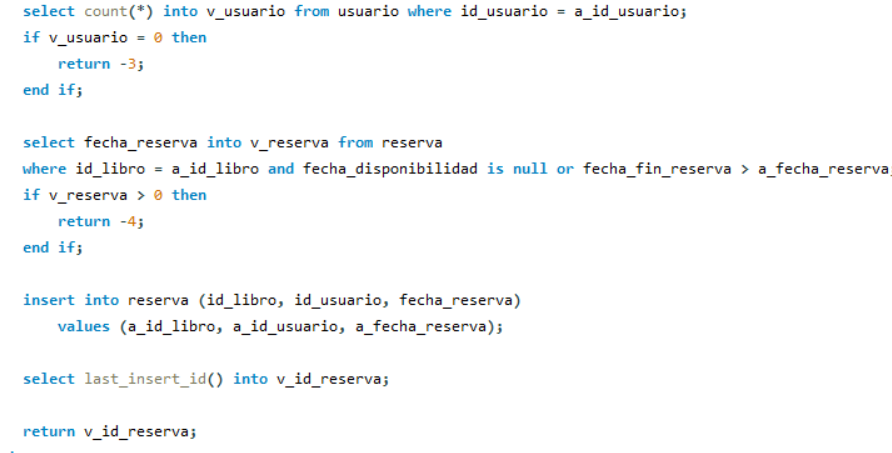




##### **Función f\_reservar(a\_id\_libro, a\_id\_usuario, a\_fecha\_reserva) returns int**

* **Comprueba que exista el libro indicado. Si no es así, devuelve -1**
* **Comprueba que el libro esté en préstamo (prestado='S'). Si no estuviera prestado, devuelve -2**
* **Comprueba que exista el usuario indicado. Si no es así, devuelve -3**
* **Comprueba que el usuario no tenga una reserva activa sobre el libro (fecha\_disponibilidad is null or fecha\_fin\_reserva > a\_fecha\_reserva). Si tuviera, se devuelve -4**
* **Inserta una fila en RESERVA, con fecha\_reserva = a\_fecha\_reserva**
* **Devuelve id\_reserva**





##### **Modificaciones a la función f\_devolver**

**Se realizan las siguientes modificaciones a la función, después de asignar la fecha de sanción (en su caso) y antes de devolver el id\_prestamo:**

* **Se comprueba si existe una reserva para el libro que se está devolviendo. Como puede haber varias, se elige la más antigua (min(id\_reserva) con fecha\_reserva <= fecha\_devolucion and fecha\_disponibilidad is null and avisado='N')**
* **Si no existe reserva, se sale de la función, devolviendo id\_reserva.**
* **Si existe reserva, se modifica: se asigna la fecha\_devolución a la fecha\_disponibilidad y 7 días más a la fecha\_fin\_reserva. Se cambia avisado a 'S'**
* **Se devuelve id\_reserva**

