Ejercicio 1

- Supongamos que tenemos una tabla de distancias de diferentes rutas y que queremos guardar estas distancias en kilómetros y en millas.
  - Distancias (ruta#, distancia_k, distancia_m).
- Crear disparadores para conseguir que cuando se introduzca (o se modifique) una distancia en kilómetros, automáticamente se introduzca también en millas y viceversa. (1 Km=0.621371 millas y 1 Milla=1.609344 Km)

Ejercicio 2

- Supongamos el siguiente esquema de base de datos:
  - Productos (CodProducto, Nombre, LineaProducto,PrecioUnitario, Stock)
  - Empleados (CodEmpleado, Salario, Comision, Nombre, CodJefe, Departamento)
  - Ventas (CodVenta, CodProducto, CodEmpleado, FechaVenta, UnidadesVendidas);

Ejercicio 2

- Escribir un disparador que limite la suma total de los salarios de los empleados a 50,000.
- Escribir un disparador que verifique que el salario de un empleado sea inferior al de su jefe.
- Escriba un procedimiento para Insertar una nueva Venta. Recuerde realizar las comprobaciones necesarias para mantener la integridad de la base de datos.
- Ahora, intente insertar una nueva venta sin el uso del procedimiento. ¿Se mantiene la integridad de la base de datos? Solucione este problema.

Ejercicio 3

- Supongamos que tenemos las tablas:
  - Cuenta (cuenta#, saldo)
  - Prestamo (prestamo#, cuenta#, capital, capital_pendiente, interes, importe_letra, num_letras, letras_pendientes)
  - letras_prestamo (prestamo#, letra#, capital_amortizado)
- Construir un disparador que, cada vez que insertemos o actualicemos una tupla en *letras_prestamo*, controle:

Ejercicio 3

- Si hay saldo suficiente en la cuenta asociada. En caso contrario, suspenda la inserción y genere una excepción que de un mensaje de salida indicando lo sucedido.
- Si hay saldo, debe descontar la cantidad de la letra de la cuenta correspondiente y actualizar la tabla préstamo para que refleje el capital pendiente, de acuerdo con el capital que amortiza la letra y el número de letras pendientes