1. Se necesita llevar un histórico de los cambios realizados en las condiciones contractuales de los empleados que están registrados en la tabla emple.
Para ello primero tienes que crear una tabla de nombre "auditaemple" con tres campos, uno de nombre "id_cambio" number (5) que será la clave primaria, otro de nombre "descripcion_cambio" de tipo VARCHAR2 (100) y otro de nombre "fecha_cambio" de tipo date.
Crea un trigger que se diga "auditasueldo" y sirva para que cada vez que se cambie el salario de un empleado se insertan en la tabla "auditaemple" un registro con el mensaje "El salario del empleado codiempleado antes era de sueldoantiguo y ahora será sueldonuevo ", la fecha actual y de id_cambio el más alto que hubiese en la tabla incrementado en 1.

```
create table auditaemple (id_canvi number(5), descripcio_canvi
varchar(100), data_canvi date, primary key(id_canvi) );
```

```
create or replace trigger auditasueldo
after update of salario on emp
for each row
declare
c number(5);
begin
  select (max(id_canvi)+1) into c from auditaemple;
  insert into auditaemple values (c, 'empleat
  '||:old.emp_no||'antiguo salario '||:old.salario||' nuevo
salario'||:new.salario,sysdate);
end;
```

2. Haz otro trigger de nombre "auditaemple" que sea una modificación de la anterior para añadir que si se da de alta un nuevo empleado se registre en la tabla "auditaemple" el mensaje "Nuevo empleado con código emp_no" con fecha de hoy.

```
create or replace trigger auditaemple after
            insert on emp
            for each row
            declare
                  i number(5);
            begin
                  select (max(id_canvi) + 1) into i from
auditaemple;
            if inserting then
                  insert into auditaemple values(i,'Nuevo empleado
con codigo' ||:new.emp_no, sysdate);
            end if;
end;
```

3. Haz otro trigger de nombre "auditaemple2" que sea una modificación de la anterior y sólo se dispare en el caso de que se intente cambiar un salario si la modificación consiste en subir el salario al empleado más de un 10%.

```
create or replace trigger auditaemple2
            before update of salario on emp
            for each row
            when(new.salario>1.1*old.salario)
            begin
            raise_application_error(-20500, 'Salario Erróneo');
end;
```

4. Crear un trigger de nombre "verificaunidades" que sirva para que en ninguna línea de detalle se puedan pedir más de 999 unidades. La forma de que un trigger pueda evitar que se lleve a cabo la operación de disparo es levantar una excepción y no tratarla.

```
create or replace trigger verificaunidades
            before insert or update on detalle
            for each row
            when(new.cantidad>999)
            begin
                  raise_application_error(-20500, 'no se puede
pedir más de 999 unidades ');
end;
```

1. Crear un trigger que se diga "verif_crea_compte" que sirva para no permitir crear una nueva cuenta con fecha de alta posterior a la fecha actual ni con "saldo Debe" positivo ni con "saldo haber" negativo o cero (es decir, para crear una cuenta deben tener dinero).

```
create or replace trigger verif_crea_compte
before insert on cuentas
for each row

declare
no_pasta EXCEPTION;

begin
if (:new.fecha_alta>sysdate or :new.saldo_debe>0 or
:new.saldo_haber<0) then
      RAISE no_pasta;
end if;
exception
when no_pasta then
raise_application_error(-20003,'no puedes abrir la cuenta');
end;
```

2. Implementa lo que sea necesario para garantizar que el tipo de movimiento de la tabla "Movimientos" sólo pueda ser "I" (ingreso) o "R" (reintegro), ambas en mayúsculas.

Tienes que plantearte qué se podría hacer en el caso de que en los datos actuales que ya están en la tabla hubiera algún registro que ya no cumpla esta condición.
También tienes que pensar si se permitirá que el usuario meta estas letras en minúsculas o no.

```
create or replace trigger verif_crea_compte1
before insert or update of tipo_mov on movimientos
for each row
declare
error_I_O EXCEPTION;
begin
if (:new.tipo_mov!='I' and :new.tipo_mov!='R' ) then
      RAISE error_I_O;
end if;
exception
when
error_I_O then
raise_application_error(-20003, 'no puedes abrir la cuenta');
end;
```

3. Implementa el trigger o triggers necesarios para garantizar que:
- Si se da de alta un nuevo movimiento la fecha no pueda ser anterior a la de hoy.

- Si se da de alta un nuevo movimiento o se modifica el tipo de movimiento, o el importe del movimiento o se elimina un movimiento el valor del "saldo Debe" y el valor del "saldo haber" sean alterados en consecuencia.
- No se pueda cambiar un movimiento de una cuenta a otra.

El valor del "saldo Debe" de la cuenta debe ser siempre la suma de todos los reintegros que se han hecho de esta cuenta y el valor del "saldo haber" debe ser la suma de todos los ingresos.

No se deben recalcular los saldos en cada operación a partir de todos los movimientos de la cuenta, sólo deben modificarse consecuentemente a la operación realizada sobre los movimientos asumiendo que el valor de partida de los saldos eran los correctos.

```
create or replace trigger verif_crea_compte2
before insert or update or delete on movimientos
for each row
declare
dia_incorrecte EXCEPTION;
insertar EXCEPTION;
begin
if inserting then
      if (:new.fecha_mov<sysdate) then
      RAISE dia_incorrecte;
      end if;
end if;
exception
when dia_incorrecte then
raise_application_error(-20003,'dia incorrecto');
end;
```

```
create or replace trigger verif_crea_compte2
before insert or update or delete on movimientos
for each row
declare
dia_incorrecte EXCEPTION;
begin
if inserting then
      if (:new.fecha_mov<sysdate) then
      RAISE dia_incorrecte;
      end if;
      if (:new.TIPO_MOV='R') then
      UPDATE  CUENTAS C SET SALDO_DEBE  = :new.IMPORTE+SALDO_DEBE ;
      end if;
      if (:new.TIPO_MOV='I') then
      UPDATE  CUENTAS C SET SALDO_HABER  = :new.IMPORTE+SALDO_HABER
;
      end if;
end if;
if   deleting then
      if (:old.TIPO_MOV='R') then
      UPDATE  CUENTAS C SET SALDO_DEBE  = :old.IMPORTE-SALDO_DEBE ;
      end if;
      if (:old.TIPO_MOV='I') then
      UPDATE  CUENTAS C SET SALDO_HABER  = :old.IMPORTE-SALDO_HABER
;
      end if;
end if;
if  updating then
UPDATE  CUENTAS C SET SALDO_DEBE  = (SELECT SUM(IMPORTE) FROM
MOVIMIENTOS
WHERE TIPO_MOV='R' AND C.COD_BANCO= COD_BANCO AND
C.COD_SUCUR=COD_SUCUR
AND C.NUM_CTA =NUM_CTA);
UPDATE  CUENTAS C SET SALDO_HABER  = (SELECT SUM(IMPORTE) FROM
MOVIMIENTOS
WHERE TIPO_MOV='I' AND C.COD_BANCO= COD_BANCO AND
C.COD_SUCUR=COD_SUCUR
```

```
AND C.NUM_CTA =NUM_CTA);
end if;
exception
when dia_incorrecte then
raise_application_error(-20003,'dia incorrecto');
end;
```