/\*1. (0.5 puntos) Crea una tabla de antiguos profesores, con los mismos datos que

ya existen en la tabla de profesores. (Recuerda el prefijo ciclos).\*/

CREATE TABLE ciclos\_antiguos\_profesores (

dni\_profesor number(10) PRIMARY KEY,

nombre\_profesor VARCHAR2(30) NOT NULL,

email\_profesor VARCHAR2(80),

telefono\_profesor NUMBER(10),

fecha\_nacimiento\_profesor DATE,

departamento NUMBER(10),

antig NUMBER(2)

);

/\*2. (1 punto) Crea un trigger que, al eliminar un profesor de la tabla de profesores,

lo inserte en la tabla antiguos profesores.\*/

CREATE OR REPLACE TRIGGER eliminarProfesorYGuardar

BEFORE DELETE ON ciclos\_profesor

FOR EACH ROW

BEGIN

INSERT INTO ciclos\_antiguos\_profesores(dni\_profesor, nombre\_profesor, email\_profesor, telefono\_profesor,fecha\_nacimiento\_profesor, departamento)

VALUES (:old.dni\_profesor, :old.nombre\_profesor, :old.email\_profesor, :old.telefono\_profesor, :old.fecha\_nacimiento\_profesor, :old.departamento);

mostrar('Profesor eliminado y añadido a la la tabla antiguos profesores: ' || :old.dni\_profesor);

END;

BEGIN

DELETE FROM CICLOS\_impartir WHERE dni\_profesor = 11111140;

DELETE FROM CICLOS\_profesor WHERE dni\_profesor = 11111140;

END;

/\*3. (0.5 puntos) Crea una tabla llamada conce\_audit\_cliente que contenga:

a. El código de auditoría que será la clave primaria.

b. El código del cliente

c. Una descripción

d. Una fecha de actualización\*/

CREATE TABLE conce\_audit\_cliente (

cod\_aud NUMBER(8) PRIMARY KEY,

cod\_cli NUMBER(8),

descripcion VARCHAR(255),

fecha\_de\_actualizacion DATE

);

/\*4. (1 punto) Crea un procedimiento insertar\_audit\_cliente2 que inserte los datos en

la tabla anterior y donde la fecha es opcional (si no ponemos nada insertará la

fecha del sistema).\*/

CREATE OR REPLACE PROCEDURE insertar\_audit\_cliente2(

p\_cod\_cli NUMBER,

p\_descripcion VARCHAR2,

p\_fecha\_de\_actualizacion DATE DEFAULT SYSDATE

)

IS

new\_cod\_aud NUMBER;

BEGIN

-- Asi obtengo el próximo codigo para que no hayan errores con la PK

-- Uso NVL para que en caso de no haya aún un codigo se asigne el 0.

SELECT NVL(MAX(cod\_aud), 0) + 1 INTO new\_cod\_aud FROM conce\_audit\_cliente;

INSERT INTO

conce\_audit\_cliente (cod\_aud, cod\_cli, descripcion, fecha\_de\_actualizacion)

VALUES (new\_cod\_aud, p\_cod\_cli, p\_descripcion, p\_fecha\_de\_actualizacion);

END;

--Prueba de inserción

BEGIN

insertar\_audit\_cliente2(11223344, 'SiTuTeVas' );

END;

/\*5. (1 punto) Crea un disparador que se ejecute antes del borrado en la tabla de

clientes y que guarde en la tabla audit\_cliente el código del cliente que se ha

borrado, la fecha actual y la siguiente descripción: ‘Borrado del cliente

{cod} : {nombre apellidos} de la ciudad :{ciudad}’.\*/

CREATE OR REPLACE TRIGGER antesDeBorrarClienteGuardar

BEFORE DELETE ON conce\_cliente

FOR EACH ROW

DECLARE

new\_cod\_aud NUMBER;

BEGIN

-- lo mismo que antes, para obtener la siguiente PK de la tabla

SELECT NVL(MAX(cod\_aud), 0) + 1 INTO new\_cod\_aud FROM conce\_audit\_cliente;

INSERT INTO conce\_audit\_cliente (cod\_aud, cod\_cli, descripcion, fecha\_de\_actualizacion)

VALUES (new\_cod\_aud, :old.IDCLI,

'Borrado del cliente ' || :old.idcli || ' : ' || :old.nombre || ' ' || :old.apellido || ' de la ciudad: ' || :old.ciudad,

SYSDATE);

END;

BEGIN

DELETE FROM conce\_ventas WHERE cliente = 44512333;

DELETE FROM conce\_cliente WHERE idcli = 44512333;

END;

/\*6. (1 punto) Crea un disparador que se ejecute después de la inserción en la tabla

de clientes y que guarde en la tabla audit\_cliente el código del cliente insertado,

la fecha y hora actual y la siguiente descripción: ‘Inserción del cliente{cod}:

{nombre apellido} de la ciudad {ciudad}’.\*/

CREATE OR REPLACE TRIGGER despuesDeInsercion

AFTER INSERT ON conce\_cliente

FOR EACH ROW

DECLARE

new\_cod\_aud NUMBER;

BEGIN

SELECT NVL(MAX(cod\_aud), 0) + 1 INTO new\_cod\_aud FROM conce\_audit\_cliente;

INSERT INTO conce\_audit\_cliente (cod\_aud, cod\_cli, descripcion, fecha\_de\_actualizacion)

VALUES (new\_cod\_aud, :new.idcli, 'Inserción del cliente ' || :new.idcli || ': ' ||

:new.nombre || ' ' || :new.apellido || ' de la ciudad ' || :new.ciudad, SYSDATE);

END;

-- Prueba del disparador al insertar un cliente

BEGIN

INSERT INTO conce\_cliente (idcli, nombre, apellido, ciudad)

VALUES (9, 'Juan', 'Pina', 'Petrer');

END;

/\*7. (1.5 puntos) Crea un disparador que se ejecute después de la modificación de la

tabla clientes, concretamente después de modificar la ciudad del cliente, guarde

en la tabla audit\_cliente la siguiente información: el código del cliente, la fecha y

hora de la modificación y la siguiente descripción: ‘Modificación de la ciudad del

cliente {cod-nombre}. Ciudad anterior: {ciudad ant}, ciudad actual: {nuevo

ciudad}’\*/

CREATE OR REPLACE TRIGGER trasModificarClientes

AFTER UPDATE ON conce\_cliente

FOR EACH ROW

DECLARE

new\_cod\_aud NUMBER;

BEGIN

IF :old.ciudad != :new.ciudad THEN -- Para que sea al modificar la ciudad del cliente

SELECT NVL(MAX(cod\_aud), 0) + 1 INTO new\_cod\_aud FROM conce\_audit\_cliente;

INSERT INTO conce\_audit\_cliente (cod\_aud, cod\_cli, descripcion, fecha\_de\_actualizacion)

VALUES (new\_cod\_aud, :new.idcli,

'Modificación de la ciudad del cliente ' || :new.nombre || '. Ciudad anterior: ' || :old.ciudad || ', ciudad actual: ' || :new.ciudad,

SYSDATE);

END IF;

END;

-- Prubea con el cliente que inserté antes.

BEGIN

UPDATE conce\_cliente SET ciudad = 'Barcelona' WHERE idcli = 17;

END;

/\*8. (1.5 punto) Ahora borra los tres disparadores anteriores para compactarlos en

uno que contenga tanto insert como update y delete. La información a guardar

será la misma que en los disparadores de los ejercicios 5,6 y 7.\*/

DROP TRIGGER antesDeBorrarClienteGuardar;

DROP TRIGGER despuesDeInsercion;

DROP TRIGGER trasModificarClientes;

CREATE OR REPLACE TRIGGER gestionAuditCliente

AFTER INSERT OR UPDATE OR DELETE ON conce\_cliente

FOR EACH ROW

DECLARE

new\_cod\_aud NUMBER;

BEGIN

SELECT NVL(MAX(cod\_aud), 0) + 1 INTO new\_cod\_aud FROM conce\_audit\_cliente;

-- Ejercicio 5, la inserción

IF INSERTING THEN

INSERT INTO conce\_audit\_cliente (cod\_aud, cod\_cli, descripcion, fecha\_de\_actualizacion)

VALUES (new\_cod\_aud, :new.idcli,

'Inserción del cliente ' || :new.idcli || ': ' || :new.nombre || ' ' || :new.apellido || ' de la ciudad ' || :new.ciudad,

SYSDATE);

-- Ejercicio 6, el borrado

ELSIF DELETING THEN

INSERT INTO conce\_audit\_cliente (cod\_aud, cod\_cli, descripcion, fecha\_de\_actualizacion)

VALUES (new\_cod\_aud, :old.idcli,

'Borrado del cliente ' || :old.idcli || ': ' || :old.nombre || ' ' || :old.apellido || ' de la ciudad: ' || :old.ciudad,

SYSDATE);

-- Ejercicio 7, la actualización

ELSIF UPDATING THEN

IF :old.ciudad != :new.ciudad THEN

INSERT INTO conce\_audit\_cliente (cod\_aud, cod\_cli, descripcion, fecha\_de\_actualizacion)

VALUES (new\_cod\_aud, :new.idcli,

'Modificación de la ciudad del cliente ' || :new.idcli || ': ' || :new.nombre || ' ' || :new.apellido || '. Ciudad anterior: ' || :old.ciudad || ', ciudad actual: ' || :new.ciudad,

SYSDATE);

END IF;

END IF;

END;

/\*9. (0.5 puntos) Crea un nuevo campo en la tabla de listas\_artista, que se llame

tipo\_artista y será de tipo carácter y tendrá uno de los siguientes valores:

'INDEPENDIENTE' O 'GRUPO'. Actualiza el campo que acabas de crear usando

sentencias update (como mucho 2 sentencias).\*/

ALTER TABLE listas\_artista ADD COLUMN tipo\_artista VARCHAR(12);

UPDATE listas\_artista SET tipo\_artista = 'INDEPENDIENTE' WHERE condition\_for\_independent\_artists;

UPDATE listas\_artista SET tipo\_artista = 'GRUPO' WHERE condition\_for\_group\_artists;

/\*10. (1.5 puntos) Crea un disparador que se ejecute al insertar datos en las tablas de

listas\_artistaind y listas\_grupo. El disparador debe guardar en la tabla

listas\_artista, ‘GRUPO’ si estamos insertando en listas\_grupo o

'INDEPENDIENTE' si estamos insertando en la tabla listas\_artistasind. Usa el

procedimiento creado en la anterior práctica.\*/

CREATE TRIGGER trg\_after\_insert\_listas\_artistaind

AFTER INSERT ON listas\_artistaind

FOR EACH ROW

BEGIN

CALL insertar\_lista\_artista(NEW.\*, 'INDEPENDIENTE');

END;

CREATE TRIGGER trg\_after\_insert\_listas\_grupo

AFTER INSERT ON listas\_grupo

FOR EACH ROW

BEGIN

CALL insertar\_lista\_artista(NEW.\*, 'GRUPO');

END;