BD TAREA 6

```
USE empresa;

-------------------------------------------------------------------------------------------------
---------------------------------  PROCEDIMIENTOS ---------------------------------------------
-------------------------------------------------------------------------------------------------

-- Procedimiento para añadir los datos de una nueva Oficina.
DELIMITER //
DROP PROCEDURE IF EXISTS agregarOficina //
CREATE PROCEDURE agregarOficina (id INT, nom VARCHAR(40), dom VARCHAR(40), loc VARCHAR(20), cod
VARCHAR(5))
BEGIN
	INSERT INTO oficinas (identificador, nombre, domicilio, localidad, codigo_postal)
	VALUES (id, nom , dom, loc, cod);
END;
//DELIMITER ;

-- Procedimiento para añadir los datos a todas las oficinas en función del procedimiento creado previamente.
DELIMITER //
DROP PROCEDURE IF EXISTS agregarOficinas //
CREATE PROCEDURE agregarOficinas ()
	BEGIN
		CALL agregarOficina( 1, 'Madrid', 'Gran vía, 37', 'Madrid', '28000' );
	END;
// DELIMITER ;

-- Procedimiento para añadir los datos de una nueva Familia.
DELIMITER //
DROP PROCEDURE IF EXISTS agregarFamilia //
CREATE PROCEDURE agregarFamilia (id INT, nom VARCHAR(40), fam INT, ofi INT)
	BEGIN
		INSERT INTO familias (identificador, nombre, familia, oficina) VALUES (id, nom, fam, ofi);
	END;
// DELIMITER ;

-- Procedimiento para añadir los datos todas las familias en función del procedimiento creado previamente.
DELIMITER //
DROP PROCEDURE IF EXISTS agregarFamilias //
CREATE PROCEDURE agregarFamilias()
	BEGIN
		CALL agregarFamilia( 11, 'Madrid-1', NULL, 1 );
	END;
// DELIMITER ;

-- Procedimiento sencillo cuya finalidad es mostrar todos los datos de las oficinas.
DELIMITER //
DROP PROCEDURE mostrarOficinas //
CREATE PROCEDURE mostrarOficinas()
	BEGIN
		SELECT * FROM oficinas;
	END;
// DELIMITER ;

CALL mostrarOficinas(); -- Llamamos al procedimiento

-- Procedimiento sencillo cuya finalidad es mostrar todos los datos de las familias
DELIMITER //
DROP PROCEDURE IF EXISTS mostrarFamilias //
CREATE PROCEDURE mostrarFamilias()
	BEGIN
		SELECT * FROM familias;
	END;
// DELIMITER ;

CALL mostrarFamilias();
```

```
-- Procedimiento sencillo cuya finalidad es mostrar todos los datos de los agentes
DELIMITER //
DROP PROCEDURE IF EXISTS mostrarAgentes //
CREATE PROCEDURE mostrarAgentes()
	BEGIN
		SELECT * FROM agentes;
	END;
// DELIMITER ;

CALL mostrarAgentes();

-- Procedimiento sencillo cuya finalidad es invocar a los tres procedimientos anteriores para mostrar los datos de las Oficinas, Familias y Agentes. Pero de forma independiente
DELIMITER //
DROP PROCEDURE IF EXISTS mostrarDatos //
CREATE PROCEDURE mostrarDatos()
	BEGIN
		CALL mostrarOficinas();
		CALL mostrarFamilias();
		CALL mostrarAgentes();
	END;
// DELIMITER ;

CALL mostrarDatos();

-- Creamos un procedimiento cuya finalidad es borrar todas las tuplas de las tablas agentes, familias y oficinas (son tres sentencias básicas de sql).
DELIMITER //
DROP PROCEDURE IF EXISTS borrarDatos //
CREATE PROCEDURE borrarDatos()
		BEGIN
			DELETE FROM agentes;
			DELETE FROM familias;
			DELETE FROM oficinas;
		END;
// DELIMITER ;

CALL borrarDatos();

-- Creamos un procedimiento cuya finalidad es restaurar las tuplas de las tablas agentes, familias y oficinas (son llamadas a los tres procedimientos de restaurar creados previamente).
DELIMITER //
DROP PROCEDURE IF EXISTS restaurarDatos //
CREATE PROCEDURE restaurarDatos()
	BEGIN
		CALL restaurarAgentes();
		CALL restaurarFamilias();
		CALL restaurarOficinas();
	END;
// DELIMITER ;

CALL restaurar Datos();
```

---------- FUNCIONES ----------

```
-----------------------------------------------------------------------------------------------------

-- Función que va a devolver el número de miembros de la familia cuyo nombre se pase como parámetro.
DELIMITER //
DROP FUNCTION IF EXISTS contarMiembrosFamilia //
CREATE FUNCTION contarMiembrosFamilia(nom VARCHAR(60)) RETURNS INT
	DETERMINISTIC -- Obligatorio en funciones.
	BEGIN
		DECLARE total INT UNSIGNED;
		DECLARE idFamilia INT DEFAULT 0;
		DECLARE CONTINUE HANDLER FOR NOT FOUND SET total = 0;
		-- Comprobamos que el nombre de familia introducido existe.
		SELECT identificador INTO idFamilia FROM familias WHERE nombre = nom;
		IF idFamilia = 0 THEN
			SET total = 0;
		ELSE
			SELECT COUNT(*) INTO total FROM agentes WHERE familia = idFamilia;
		END IF;
		RETURN total;
	END;
// DELIMITER ;

SELECT contarMiembrosFamilia('Madrid-1.2'); -- Llamamos a la función.

-- Funcion que va a devolver el nombre de la familia cuyo identificador se pasa como parametro.
DELIMITER //
DROP FUNCTION IF EXISTS buscarNombreFamilia // -- Si existe añguna función con este nombre, la borramos
CREATE FUNCTION buscarNombreFamilia (id INT) RETURNS VARCHAR(60)
	DETERMINISTIC -- Obligatorio en funciones (DETERMINISTIC | NO SQL | READS SQL DATA)
	BEGIN
		DECLARE resultado VARCHAR(60) DEFAULT 'No existe este identificador';
		SELECT nombre INTO resultado FROM familias WHERE identificador = id;
		RETURN resultado;
	END;
// DELIMITER ;

SELECT buscarNombreFamilia(31); -- Llamamos a la función

-----------------------------------------------------------------------------------------------------
----------------------------------  DISPARADORES  ---------------------------------------
-----------------------------------------------------------------------------------------------------

-- Código que se ejecuta al agregar una tupla en la tabla Oficinas, cuya finalidad es volcar esos datos sobre la tabla oficinasCopia
DELIMITER //
DROP TRIGGER IF EXISTS TRAS_AGREGAR_OFICINA //
CREATE TRIGGER TRAS_AGREGAR_OFICINA
AFTER UPDATE ON oficinas FOR EACH ROW
	BEGIN
		DECLARE FIN INT DEFAULT FALSE; -- Variable asociada al bucle, para pararlo.
		DECLARE id INT; -- Declaramos las variables en las que guardaremos los campos del cursor.
		DECLARE nom VARCHAR(40);
		DECLARE dom VARCHAR(40);
		DECLARE loc VARCHAR(20);
		DECLARE cp VARCHAR(5);
		DECLARE cursorOficinas CURSOR FOR SELECT * FROM oficinas;
		-- Declaramos manejo de error cuando no existan mas campos en el cursor, que será condición para
		finalización de bucle.
		DECLARE CONTINUE HANDLER FOR NOT FOUND SET FIN = TRUE;
		OPEN cursorOficinas; -- Abrimos cursor;
		leerDatosOficinas: LOOP
			FETCH cursorOficinas INTO id, nom, dom, loc, cp; -- Leemos cada fila del cursor y guardamos
			las columnas en variables.

			IF FIN THEN
				LEAVE leerDatosOficinas; -- Si no existen mas filas, se para el bucle (aunque no se sale
			de él).
			END IF;
```

```
                INSERT INTO oficinascopia VALUES (id, nom, dom, loc, cp); -- Guardamos los nuevos valores en la tabla
                oficinascopia tupla a tupla.
                END LOOP;
                CLOSE cursorOficinas; -- Cerramos el cursor
        END;
// DELIMITER ;

-- Código que se ejecuta al agregar una tupla en la tabla Familias, cuya finalidad es volcar esos datos sobre la tabla familiascopia.
DELIMITER //
DROP TRIGGER IF EXISTS TRAS_AGREGAR_FAMILIA //
CREATE TRIGGER TRAS_AGREGAR_FAMILIA
AFTER INSERT ON familias FOR EACH ROW
        BEGIN
                DECLARE FIN INT DEFAULT FALSE; -- Variable asociada al bucle, para pararlo.
                DECLARE id INT; -- Declaramos las variables en las que guardaremos los campos del cursor.
                DECLARE nom VARCHAR(40);
                DECLARE fam INT;
                DECLARE ofi INT;
                DECLARE cursorFamilias CURSOR FOR SELECT * FROM familias;
                DECLARE CONTINUE HANDLER FOR NOT FOUND SET FIN = TRUE; -- Declaramos manejo de error cuando
                no existan mas campos en el cursor, que será condición para finalización de bucle.
                OPEN cursorFamilias; -- Abrimos cursor;
                leerDatosFamilias: LOOP
                        FETCH cursorFamilias INTO id, nom, fam, ofi; -- Leemos cada fila del cursor y guardamos las columnas
                        en variables
                        IF FIN THEN
                                LEAVE leerDatosFamilias; -- Si no existen mas filas, se para el bucle (aunque no se sale de él).
                        END IF;
                        INSERT INTO familiascopia VALUES (id, nom, fam, ofi); -- Guardamos los nuevos valores en la tabla
                        oficinascopia tupla a tupla.
                END LOOP;
                CLOSE cursorFamilias; -- Cerramos el cursor
        END;
// DELIMITER ;

-- Código que se ejecuta al agregar una tupla en la tabla Agentes, cuya finalidad es volcar esos datos sobre la tabla agentescopia.
DELIMITER //
DROP TRIGGER IF EXISTS TRAS_AGREGAR_AGENTE //
CREATE TRIGGER TRAS_AGREGAR_AGENTE
AFTER INSERT ON agentes FOR EACH ROW
        BEGIN
                DECLARE FIN INT DEFAULT FALSE; -- Variable asociada al bucle, para pararlo.
                DECLARE id INT; -- Declaramos las variables en las que guardaremos los campos del cursor.
                DECLARE nom VARCHAR(60);
                DECLARE usu VARCHAR(20);
                DECLARE cla VARCHAR(20);
                DECLARE hab INT;
                DECLARE cat INT;
                DECLARE fam INT;
                DECLARE ofi INT;
                DECLARE cursorAgentes CURSOR FOR SELECT * FROM agentes;
                DECLARE CONTINUE HANDLER FOR NOT FOUND SET FIN = TRUE; -- Declaramos manejo de error cuando
                no existan mas campos en el cursor, que será condición para finalización de bucle.
                OPEN cursorAgentes; -- Abrimos cursor;
                leerDatosAgentes: LOOP
                        FETCH cursorAgentes INTO id, nom, usu, cla, hab, cat, fam, ofi; -- Leemos cada fila del cursor y
                        guardamos las columnas en variables
                        IF FIN THEN
                                LEAVE leerDatosAgentes; -- Si no existen mas filas, se para el bucle (aunque no se sale de él).
                        END IF;
                        INSERT agentescopia VALUES (id, nom, usu, cla, hab, cat, fam, ofi); -- Guardamos los nuevos valores en
                        la tabla oficinascopia tupla a tupla.
                        END LOOP;
                 CLOSE cursorAgentes; -- Cerramos el cursor
      END;
// DELIMITER ;
```

```
-- Código que se ejecuta al modificar datos de la tabla Agentes, cuya finalidad es volcar esos datos sobre la tabla agentescopia. En la tabla oficinasOLD se volcarán los datos que van a ser modificados.
-- Los trigger trabajan con dos tipos de variable de tipo %ROWTYPE: OLD (copia del registro antes de la acción) y NEW (después de la acción).
DELIMITER //
DROP TRIGGER IF EXISTS TRAS_MODIFICAR_OFICINA //
CREATE TRIGGER TRAS_MODIFICAR_OFICINA
AFTER UPDATE ON oficinas FOR EACH ROW
	BEGIN
		-- Guardamos los nuevos valores en la tabla oficinascopia.
		INSERT oficinascopia (identificador, nombre, domicilio, localidad, codigo_postal) VALUES (NEW.identificador,
		NEW.nombre, NEW.domicilio, NEW.localidad, NEW.codigo_postal);
		-- agregamos los datos que van a ser modificados en la tabla oficinasOLD
		INSERT oficinasold (identificador, nombre, domicilio, localidad, codigo_postal) VALUES (OLD.identificador,
		OLD.nombre, OLD.domicilio, OLD.localidad, OLD.codigo_postal);
	END;
// DELIMITER ;

-- Código que se ejecuta al modificar datos de la tabla Familias, cuya finalidad es volcar esos datos sobre la tabla familiasCopia. En la tabla familiasold se volcarán los datos que van a ser modificados.
DELIMITER //
DROP TRIGGER IF EXISTS TRAS_MODIFICAR_FAMILIA //
CREATE TRIGGER TRAS_MODIFICAR_FAMILIA
AFTER UPDATE ON familias FOR EACH ROW
	BEGIN
		-- Guardamos los nuevos valores en la tabla familiascopia.
		INSERT familiascopia (identificador, nombre, familia, oficina) VALUES (NEW.identificador, NEW.nombre,
		NEW.familia, NEW.oficina);
		-- agregamos los datos que van a ser modificados en la tabla familiasold
		INSERT familiasold (identificador, nombre, familia, oficina) VALUES (OLD.identificador, OLD.nombre,
		OLD.familia, OLD.oficina);
	END;
// DELIMITER ;

-- Código que se ejecuta al modificar datos de la tabla Agentes, cuya finalidad es volcar esos datos sobre la tabla agentesCopia. En la tabla agentesOLD se volcarán los datos que van a ser modificados.
DELIMITER //
DROP TRIGGER IF EXISTS TRAS_MODIFICAR_AGENTE //
CREATE TRIGGER TRAS_MODIFICAR_AGENTE
AFTER UPDATE ON agentes FOR EACH ROW
	BEGIN
		-- Guardamos los nuevos valores en la tabla agentescopia.
		INSERT agentescopia (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina) VALUES
		(NEW.identificador, NEW.nombre, NEW.usuario, NEW.clave, NEW.habilidad, NEW.categoria, NEW.familia,
		NEW.oficina);
		-- agregamos los datos que van a ser modificados en la tabla agenteold
		INSERT agentesold (identificador, nombre, familia, oficina) VALUES (OLD.identificador, OLD.nombre,
		OLD.usuario, OLD.clave, OLD.habilidad, OLD.categoria,OLD.familia, OLD.oficina);
	END;
// DELIMITER ;

-- Código que se ejecuta al borrar datos de la tabla Oficinas, cuya finalidad es volcar esos datos sobre la tabla oficinascopia. En la tabla oficinasold se volcarán los datos que van a ser eliminados.
DELIMITER //
DROP TRIGGER IF EXISTS TRAS_BORRAR_OFICINA //
CREATE TRIGGER TRAS_BORRAR_OFICINA
AFTER DELETE ON oficinas FOR EACH ROW
	BEGIN
		-- Borramos en la tabla oficinasCopia
		DELETE FROM oficinascopia WHERE identificador = OLD.identificador;
		-- Agragamos los datos que van a ser borrados en la tabla oficinasold.
		INSERT oficinasold (identificador, nombre, domicilio, localidad, codigo_postal) VALUES (OLD.identificador,
		OLD.nombre, OLD.domicilio, OLD.localidad, OLD.codigo_postal);
	END;
// DELIMITER ;

-- Código que se ejecuta al borrar datos de la tabla Familias, cuya finalidad es volcar esos datos sobre la tabla familiascopia. En la
```

tabla familiasold se volcarán los datos que van a ser eliminados.

```
DELIMITER //
DROP TRIGGER IF EXISTS TRAS_BORRAR_FAMILIA //
CREATE TRIGGER TRAS_BORRAR_FAMILIA
AFTER DELETE ON familias FOR EACH ROW
	BEGIN
		-- Borramos en la tabla familiasCopia
		DELETE FROM familiascopia WHERE identificador = OLD.identificador;
		-- Agragamos los datos que van a ser borrados en la tabla familiasold
		INSERT familiasold (identificador, nombre, familia, oficina) VALUES (OLD.identificador, OLD.nombre,
		OLD.familia, OLD.oficina);
	END;
// DELIMITER ;
```

-- Código que se ejecuta al borrar datos de la tabla Agentes, cuya finalidad es volcar esos datos sobre la tabla agentesCopia. En la tabla agentesOLD se volcarán los datos que van a ser eliminados.

```
DELIMITER //
DROP TRIGGER IF EXISTS TRAS_BORRAR_AGENTE //
CREATE TRIGGER TRAS_BORRAR_AGENTE
AFTER DELETE ON agentes FOR EACH ROW
	BEGIN
		-- Borramos en la tabla agentesCopia
		DELETE FROM agentescopia WHERE identificador = OLD.identificador;
		-- Agragamos los datos que van a ser borrados en la tabla agentesold
		INSERT agentesold (identificador, nombre, usuario, clave, habilidad, categoria, familia, oficina) VALUES
		(OLD.identificador, OLD.nombre, OLD.usuario, OLD.clave, OLD.habilidad, OLD.categoria, OLD.familia,
		OLD.oficina);
	END;
// DELIMITER ;
```

```
-------------------------------------------------------------------------------------------------
----------------------------------  CURSORES  -----------------------------------------------
-------------------------------------------------------------------------------------------------
```

-- Creamos un cursor cuya finalidad será restaurar la tabla oficinas tras un borrado a partir de las tuplas que hay en la tabla oficinasold

```
DELIMITER //
DROP PROCEDURE IF EXISTS restaurarOficinas //
CREATE PROCEDURE restaurarOficinas()
	BEGIN
		DECLARE FIN		INT DEFAULT FALSE;	-- Variable asociada al bucle, para finalización del mismo.
		DECLARE var_id		INT;
		DECLARE var_nom		VARCHAR( 40 );
		DECLARE var_dom		VARCHAR( 40 );
		DECLARE var_loc		VARCHAR( 20 );
		DECLARE var_cp		VARCHAR(  5 );
		DECLARE cursorOficinasOld CURSOR FOR SELECT * FROM oficinasold;  -- Variable cursor que leerá cada
		tupla de la tabla oficinasold.
		DECLARE CONTINUE HANDLER FOR NOT FOUND SET FIN = TRUE;	-- Manejo de error cuando no se
		encuentren más tuplas que leer, por el que la variable asociada al bucle cambiará de valor.
		OPEN cursorOficinasOld;-- Abrimos el cursor
		leerOficinasOld:  LOOP
			FETCH cursorOficinasOld INTO var_id, var_nom, var_dom, var_loc, var_cp;  -- Se leen los valores de los
			atributos asociados a cada tupla, uno a uno.
			IF FIN THEN
				LEAVE leerOficinasOld;  -- Si no hay más tuplas que leer salimos de este bucle.
			END IF;
			IF EXISTS (SELECT * FROM oficinas WHERE identificador = var_id ) THEN  -- Si ya existe ese registro
			en la tabla, lo midificamos.
					UPDATE oficinas SET nombre = var_nom, domicilio = var_dom, localidad = var_loc,
					codigo_postal = var_cp WHERE identificador = var_id;
			ELSE	-- Si NO existe, lo añadimos.
				INSERT oficinas VALUES ( var_id, var_nom, var_dom, var_loc, var_cp );
			END IF;
		END LOOP; -- Cerramos el bucle.
		CLOSE cursorOficinasOld;-- Cerramos el cursor.
	END;
// DELIMITER ;
```

```
-- Creamos un cursor cuya finalidad será restaurar la tabla familias tras un borrado a partir de las tuplas que hay en la tabla familiasold
DELIMITER //
DROP PROCEDURE IF EXISTS restaurarFamilias //
CREATE PROCEDURE restaurarFamilias()
	BEGIN
		DECLARE FIN		INT DEFAULT FALSE;-- Variable asociada al bucle, para finalización del mismo.
		DECLARE var_id		INT;
		DECLARE var_nom		VARCHAR( 40 );
		DECLARE var_fam		INT;
		DECLARE var_ofi		INT;
		DECLARE cursorFamiliasOld CURSOR FOR SELECT * FROM familiasold;  -- Variable cursor que leerá cada tupla de la tabla familiasold.
		DECLARE CONTINUE HANDLER FOR NOT FOUND SET FIN = TRUE;    -- Manejo de error cuando no se encuentren más tuplas que leer, por el que la variable asociada al bucle cambiará de valor.
		OPEN cursorFamiliasOld;-- Abrimos el cursor
		leerFamiliasOld:  LOOP
			FETCH cursorFamiliasOld INTO var_id, var_nom, var_fam, var_ofi;  -- Se leen los valores de los atributos asociados a cada tupla, uno a uno.
			IF FIN THEN
				LEAVE leerFamiliasOld;  -- Si no hay más tuplas que leer salimos de este bucle.
			END IF;
			IF EXISTS (SELECT * FROM familias WHERE identificador = var_id ) THEN -- Si ya existe ese registro en la tabla, lo midificamos.
				UPDATE familias SET nombre = var_nom, familia = var_fam, oficina = var_ofi WHERE identificador = var_id;
			ELSE -- Si NO existe, lo añadimos.
				INSERT familias VALUES (var_id, var_nom, var_fam, var_ofi);
			END IF;
		END LOOP; -- Cerramos el bucle.
		CLOSE cursorFamiliasOld; -- Cerramos el cursor.
	END;
// DELIMITER ;

-- Creamos un cursor cuya finalidad será restaurar la tabla agentes tras un borrado a partir de las tuplas que hay en la tabla agentesold
DELIMITER //
DROP PROCEDURE IF EXISTS restaurarAgentes //
CREATE PROCEDURE restaurarAgentes()
	BEGIN
		DECLARE FIN		INT  DEFAULT  FALSE;  -- Variable asociada al bucle, para finalización del mismo.
		DECLARE var_id		INT;
		DECLARE var_nom		VARCHAR( 60 );
		DECLARE var_usu		VARCHAR( 20 );
		DECLARE var_cla		VARCHAR( 20 );
		DECLARE var_hab		INT;
		DECLARE var_cat		INT;
		DECLARE var_fam		INT;
		DECLARE var_ofi		INT;
		DECLARE cursorAgentesOld CURSOR FOR SELECT * FROM Agentesold; -- Variable cursor que leerá cada tupla de la tabla familiasold.
		DECLARE CONTINUE HANDLER FOR NOT FOUND SET FIN = TRUE; -- Manejo de error cuando no se encuentren más tuplas que leer, por el que la variable asociada al bucle cambiará de valor.
		OPEN cursorAgentesOld; -- Abrimos el cursor
		leerAgentesOld:  LOOP
			FETCH cursorAgentesOld INTO var_id, var_nom, var_usu, var_cla, var_hab, var_fam, var_ofi;  -- Se leen los valores de los atributos asociados a cada tupla, uno a uno.
			IF FIN THEN
				LEAVE leerAgentesOld;   -- Si no hay más tuplas que leer salimos de este bucle.
			END IF;
			IF EXISTS (SELECT * FROM agentes WHERE identificador = var_id ) THEN -- Si ya existe ese registro en la tabla, lo midificamos.
				UPDATE agentes SET nombre = var_nom, usuario = var_usu, clave = var_cla, habilidad = var_hab, categoria = var_cat, familia = var_fam, oficina = var_ofi WHERE identificador = var_id;
			ELSE    -- Si NO existe, lo añadimos.
				INSERT agentes VALUES (var_id, var_nom, var_usu, var_cla, var_hab, var_fam, var_ofi);
			END IF;
		END LOOP;                                                -- Cerramos el bucle.
```

```
		CLOSE cursorAgentesOld;					-- Cerramos el cursor.
	END;
// DELIMITER ;

-- Creamos un procedimiento cuya finalidad es incrementar la categoría de los agentes en una unidad. El procedimiento se realizará a través del cursor
DELIMITER //
DROP PROCEDURE IF EXISTS aumentarCategoriaAgentes //
CREATE PROCEDURE aumentarCategoriaAgentes()
	BEGIN
		DECLARE FIN		INT DEFAULT FALSE;	-- Variable asociada al bucle, para finalización del mismo.
		DECLARE var_id		INT;			-- variable en el que se guardará el identificador de cada tupla.
		DECLARE var_cat		INT;			-- variable en el que se guardará la categoría de cada tupla.
		DECLARE cursorAgentes CURSOR FOR SELECT identificador, categoria FROM agentes; -- Variable cursor que leerá cada tupla (dos atributos únicamente) de la tabla oficinasold.
		DECLARE CONTINUE HANDLER FOR NOT FOUND SET FIN = TRUE; -- Variable de manejo de error asociada al bucle, que cambiará de valor cuando ya no haya ninguna tupla que leer.
		OPEN cursorAgentes;					-- Se abre el cursor.
		leerAgentes: LOOP
			FETCH cursorAgentes INTO var_id, var_cat; -- Se leen los valores de los dos atributos asociados a cada tupla.
			IF FIN THEN
				LEAVE leerAgentes; -- Si no hay más tuplas que leer se para el bucle (no salimos de él).
			END IF;
			UPDATE agentes SET categoria = categoria + 1 WHERE identificador = var_id; -- Se incrementa la categoría de cada agente, uno a uno.
		END LOOP;
		CLOSE cursorAgentes; -- Se cierra el cursor.
	END;
// DELIMITER ;

-- Creamos un procedimiento cuya finalidad es decrementar la categoría de los agentes en una unidad. El procedimiento se realizará a través del cursor
DELIMITER //
DROP PROCEDURE IF EXISTS disminuirCategoriaAgentes //
CREATE PROCEDURE disminurCategoriaAgentes()
	BEGIN
		DECLARE FIN		INT DEFAULT FALSE; -- Variable asociada al bucle, para finalización del mismo.
		DECLARE var_id		INT; -- variable en el que se guardará el identificador de cada tupla.
		DECLARE var_cat		INT; -- variable en el que se guardará la categoría de cada tupla.
		DECLARE cursorAgentes CURSOR FOR SELECT identificador, categoria FROM agentes; -- Variable cursor que leerá cada tupla (dos atributos únicamente) de la tabla oficinasold.
		DECLARE CONTINUE HANDLER FOR NOT FOUND SET FIN = TRUE; -- Variable de manejo de error asociada al bucle, que cambiará de valor cuando ya no haya ninguna tupla que leer.
		OPEN cursorAgentes;			-- Se abre el cursor.
		leerAgentes: LOOP
			FETCH cursorAgentes INTO var_id, var_cat; -- Se leen los valores de los dos atributos asociados a cada tupla.
			IF FIN THEN
				LEAVE leerAgentes; -- Si no hay más tuplas que leer se para el bucle (no salimos de él).
			END IF;
			UPDATE agentes SET categoria = categoria -1 WHERE identificador = var_id; -- Se incrementa la categoría de cada agente, uno a uno.
		END LOOP;
		CLOSE cursorAgentes; -- Se cierra el cursor.
		END;
// DELIMITER ;
```