Rodríguez López, Alejandro. UO281827

Cuestiones:

1. **Crear un trigger que cada vez que se inserte una fila en la tabla 'estudiante_grado_modulo' se presente el mensaje 'Se ha realizado una inserción'.**

```
create or replace function presenta_insercion() returns trigger as $$
begin
    if tg_op = 'INSERT' then
        raise notice 'Se ha realizado una insercion';
        return new;
    end if;
    return null;
end;
$$ language plpgsql;

create trigger tg_presenta_insercion after
insert on estudiante_grado_modulo
for each row execute procedure presenta_insercion();
```

2. **Crear un trigger llamado 'presenta_InsertDeleteUpdateOperacion', donde cada vez que se realiza una operación de inserción ('INSERT'), un borrado ('DELETE') o una actualización ('UPDATE') se presente un mensaje. (REALIZAR POR LOS ESTUDIANTES)**

```
create or replace function presenta_InsertDeleteUpdateOperacion()
returns trigger as $$
begin
    if tg_op = 'INSERT' then
        raise notice 'Se ha realizado una insercion';
        return new;
    elsif tg_op = 'DELETE' then
        raise notice 'Se ha realizado un borrado';
        return new;
    elsif tg_op = 'UPDATE' then
        raise notice 'Se ha realizado una actualizacion';
        return new;
    end if;
    return null;
end;
$$ language plpgsql;

create trigger tg_presenta_InsertDeleteUpdateOperacion after
insert or delete or update
on estudiante_grado_modulo
for each row execute procedure presenta_InsertDeleteUpdateOperacion();
```

3. **Añadir dos columnas a la table 'estudiante_grado_modulo' llamadas 'nota' (int) y 'grado' (varchar(10)), con un valor por defecto de 0 e 'indefinido' respectivamente. (REALIZAR POR LOS ESTUDIANTES)**

```
ALTER TABLE estudiante_grado_modulo ADD COLUMN nota int default 0;
ALTER TABLE estudiante_grado_modulo ADD COLUMN grado varchar(10) default
'indefinido';
```

4. **Cada vez que se realiza una inserción, actualización o borrado en la tabla 'estudiante_grado_modulo' se necesita registrar que dicha operación ha sido ejecutada. Para tal fin, se creará una tabla denominada 'operacion_notas_log' que deberá contener los siguientes campos: (REALIZAR POR LOS ESTUDIANTES)**

Tabla: operacion_notas_log	
Atributo	**Tipo**
operacion	**char(1)**
hora	**timestamp**
estudiantetid	**int**
moduloid	**int**
nota	**int**

```
CREATE TABLE operacion_notas_log (
operacion char(1) not NULL,
hora timestamp not null,
estudianteId int not null,
moduloId int not null,
nota int);
```

5. **Crea un trigger para que cada vez que se actualice una nota en la tabla 'estudiante_grado_modulo' se añada un registro en la tabla 'operacion_notas_log' con la siguiente información: ('U', hora actual, estudiante id, modulo id, nota).**

```
create or replace function operacion_log() returns trigger as $$
begin
if tg_op = 'UPDATE' then
insert into operacion_notas_log(operacion, hora, estudianteid, moduloid,
nota) values ('U', now(), new.estudiante_id, new.modulo_id, new.nota);
return new;
end if;
return null;
end;
$$ language plpgsql;

create trigger tg_operacion_log after
update on estudiante_grado_modulo
for each row execute procedure operacion_log();
```

5.1. Realizar otras modificaciones y comprueba los cambios en la tabla 'operacion_notas_log'. (REALIZAR POR LOS ESTUDIANTES)

```
update estudiante_grado_modulo
set nota = 5
where estudiante_id=1 and modulo_id=4;

INSERT INTO estudiante_grado_modulo
VALUES (2, 4, 7);

UPDATE estudiante_grado_modulo
SET nota=6
WHERE estudiante_id = 2 AND modulo_id = 4;
```

6. Eliminar operacion_notas_log y su trigger.

```
DROP TABLE operacion_notas_log;
DROP TRIGGER tg_operacion_log ON estudiante_grado_modulo;
```

7. Crear un trigger similar al anterior pero que también registre las operaciones de inserción ('I') y borrado ('D'). Usa la estructura del condicional anidada. Escribe al menos un ejemplo para comprobar el funcionamiento de cada uno de los dos tipos de operaciones añadidas. (REALIZAR POR LOS ESTUDIANTES)

```
create or replace function operacion_log() returns trigger as $$
begin
if tg_op = 'UPDATE' then
insert into operacion_notas_log(operacion, hora, estudianteid, moduloid,
nota) values ('U', now(), new.estudiante_id, new.modulo_id, new.nota);
return new;
elsif tg_op = 'INSERT' THEN
insert into operacion_notas_log(operacion, hora, estudianteid, moduloid,
nota) values ('I', now(), new.estudiante_id, new.modulo_id, new.nota);
elsif tg_op = 'DELETE' THEN
insert into operacion_notas_log(operacion, hora, estudianteid, moduloid,
nota) values ('D', now(), old.estudiante_id, old.modulo_id, old.nota);
return old;
end if;
return null;
end;
$$ language plpgsql;
```

```
create trigger tg_operacion_log after
update or insert or delete
on estudiante_grado_modulo
for each row execute procedure operacion_log();

INSERT INTO estudiante_grado_modulo
VALUES (1, 1, 1);

UPDATE estudiante_grado_modulo
SET nota = 3
WHERE estudiante_id=1 AND modulo_id=1 AND grado_id=1;

DELETE
FROM estudiante_grado_modulo
WHERE estudiante_id=1 AND grado_id=1 AND modulo_id=1;
```

8. **Borrar la tabla 'operacion_notas_log'. También será necesario borrar el trigger tg_operacion_log que insertaba en la tabla borrada. (REALIZAR POR LOS ESTUDIANTES)**

```
DROP TABLE operacion_notas_log;
DROP TRIGGER tg_operacion_log ON estudiante_grado_modulo;
```

9. **Crear una tabla denominada 'operacion_calificacion_log' similar a la del apartado 4, pero añadiendo el atributo 'calificacion'. (REALIZAR POR LOS ESTUDIANTES)**

Tabla: operacion_calificacion_log	
Atributo	**Tipo**
operacion	char(1)
stamp	timestamp
estudiantetid	int
moduloid	int
nota	int
calificacion	varchar(10)

```
CREATE TABLE operacion_calificacion_log(
operacion char(1) not NULL,
hora timestamp not null,
estudianteId int not null,
moduloId int not null,
nota int, calificacion varchar(10));
```

10. **Crear un trigger denominado 'calificacion_log' similar al del punto 5, pero antes de que la nota sea insertada o actualizada en la tabla 'estudiante_grado_modulo' se calculará una calificación no numérica de acuerdo al baremo indicado en la siguiente tabla:**

nota	calificación
< 4	'Pobre'
4 <= x < 5	'No buena'
5 <= x < 7	'Buena'
7 <= x < 9	'Muy buena'
9 <= x < = 10	'Excelente'

```
create or replace function calificacion_log() returns trigger as $$
declare
    calif varchar(10);
begin
-- calif
if new.nota < 4 THEN
    calif = 'Pobre';
elsif new.nota < 5 THEN
    calif = 'No buena';
elsif new.nota < 7 THEN
    calif = 'Buena';
elsif new.nota < 9 THEN
    calif = 'Muy buena';
else
    calif = 'Excelente';
end if;

-- insert
if tg_op = 'UPDATE' then
insert into operacion_calificacion_log values ('U', now(), new.estudiante_id,
new.modulo_id, new.nota, calif);
return new;
elsif tg_op = 'INSERT' THEN
insert into operacion_calificacion_log values ('I', now(), new.estudiante_id,
new.modulo_id, new.nota, calif);
elsif tg_op = 'DELETE' THEN
insert into operacion_calificacion_log values ('D', now(), old.estudiante_id,
old.modulo_id, old.nota, calif);
return old;
end if;
return null;
end;
$$ language plpgsql;
```

```
create trigger tg_calificacion_log after
update
or
insert
or
delete on estudiante_grado_modulo
for each row execute procedure calificacion_log();
```