



DEFENSA HITO 4 BASE DE DATOS II

ESTUDIANTE



**JHONATAN DAVID
ALANOCÁ BLANCO**

A photograph of a person sitting on a concrete bench, looking down at their smartphone. They are wearing a light-colored long-sleeved shirt and blue jeans. The background shows a brick wall and some foliage.

MANEJO DE CONCEPTOS

A large, semi-transparent watermark or graphic element in the foreground. It features the word "DATA" in a bold, white, sans-serif font. The letters are partially obscured by a collage of various digital icons, including a globe, a circuit board, a bar chart, a stethoscope, an envelope, a mail icon, a triangle, a heart rate monitor, a network node, a cloud, a keyboard, and a document.

DATA

1. DEFINA QUE ES LENGUAJE PROCEDURAL EN MYSQL.

Permite escribir programas que contienen instrucciones de control de flujo y estructuras de programación, como bucles, condicionales, funciones y procedimientos almacenados. En otras palabras, se trata de una extensión del lenguaje SQL que permite crear código más complejo y sofisticado para realizar operaciones más avanzadas en la base de datos.

2. DEFINA QUE ES UNA FUNCIÓN EN MYSQL.

Una función es un subprograma que realiza una tarea específica y devuelve un valor.

3. CUÁL ES LA DIFERENCIA ENTRE FUNCIONES Y PROCEDIMIENTOS ALMACENADOS.

- Las funciones son subprogramas que devuelven un valor
- Los procedimientos almacenados son un conjunto de instrucciones que pueden realizar modificaciones a la base de datos y no devuelven un valor en específico, también un proceso almacenado se guarda en el servidor de la base de datos y esta puede tener funciones dentro de ella.

4. CÓMO SE EJECUTA UNA FUNCIÓN Y UN PROCEDIMIENTO ALMACENADO.

para ejecutar una función se hace el uso de SELECT:

SELECT nombre_funcion(PARAMETRO1, PARAMETRO2, ...);

```
SELECT INSERTA_DATOS(#INGERESE DATOS);
```

Para ejecutar una función se hace el uso de CALL O SELECT

```
INSERTA_DATOS( FECHA: NOW(), USUARIO: USER(), HOSTNAME: @@HOSTNAME, ACCION: 'UPDATE', ANTES: ANTES, DESPUES: DESPUES);
```

```
SELECT INSERTA_DATOS(#INGERESE DATOS);
```

5. DEFINA QUE ES UNA TRIGGER EN MYSQL.

Es un objeto de base de datos que se define para ejecutar automáticamente un conjunto de instrucciones SQL cuando ocurre un evento específico en una tabla. Los triggers se utilizan para realizar acciones automatizadas, como validar o modificar datos, cuando se inserta, actualiza o elimina registros en una tabla determinada.

```
#ESTRUCTURA DEL TRIGGER
# CREATE TRIGGER nombre_trigger
# {BEFORE | AFTER} {INSERT | UPDATE | DELETE}
# ON nombre_tabla
# FOR EACH ROW
# BEGIN
#     -- Código del trigger
# END;
```

6. EN UN TRIGGER QUE PAPEL JUEGA LAS VARIABLES OLD Y NEW

- La variable OLD contiene los valores antiguos de las columnas antes de la operación (por ejemplo, los valores de las columnas antes de una actualización o eliminación).
- La variable NEW contiene los valores nuevos o actualizados de las columnas después de la operación (por ejemplo, los valores de las columnas después de una inserción o actualización).

Estas variables se utilizan dentro del trigger para realizar comparaciones, aplicar lógica condicional o realizar acciones basadas en los valores antiguos o nuevos de las filas.

```
SET ANTES = CONCAT(OLD.id_usr, ' ', OLD.nombre_completo, ' ', OLD.fecha_nac);
SET DESPUES = CONCAT(NEW.id_usr, ' ', NEW.nombre_completo, ' ', NEW.fecha_nac);
```

7. EN UN TRIGGER QUE PAPEL JUEGA LOS CONCEPTOS (CLÁUSULAS) BEFORE O AFTER

- BEFORE: Un trigger definido con la cláusula BEFORE se ejecuta antes de que se realice la operación que lo activó. Puede ser utilizado para realizar acciones previas a la operación, como realizar validaciones o ajustar los valores de los datos antes de que se realice la operación en la tabla.
- AFTER: Un trigger definido con la cláusula AFTER se ejecuta después de que se haya realizado la operación que lo activó. Puede ser utilizado para realizar acciones posteriores a la operación, como auditar los cambios realizados, actualizar registros adicionales o enviar notificaciones.

8. A QUE SE REFIERE CUANDO SE HABLA DE EVENTOS EN TRIGGERS

se refiere a las operaciones que ocurren en una tabla y que pueden activar un trigger. Los eventos comunes son:

- INSERT: Ocurre cuando se inserta un nuevo registro en la tabla.
- UPDATE: Ocurre cuando se actualiza uno o varios registros en la tabla.
- DELETE: Ocurre cuando se elimina uno o varios registros de la tabla.

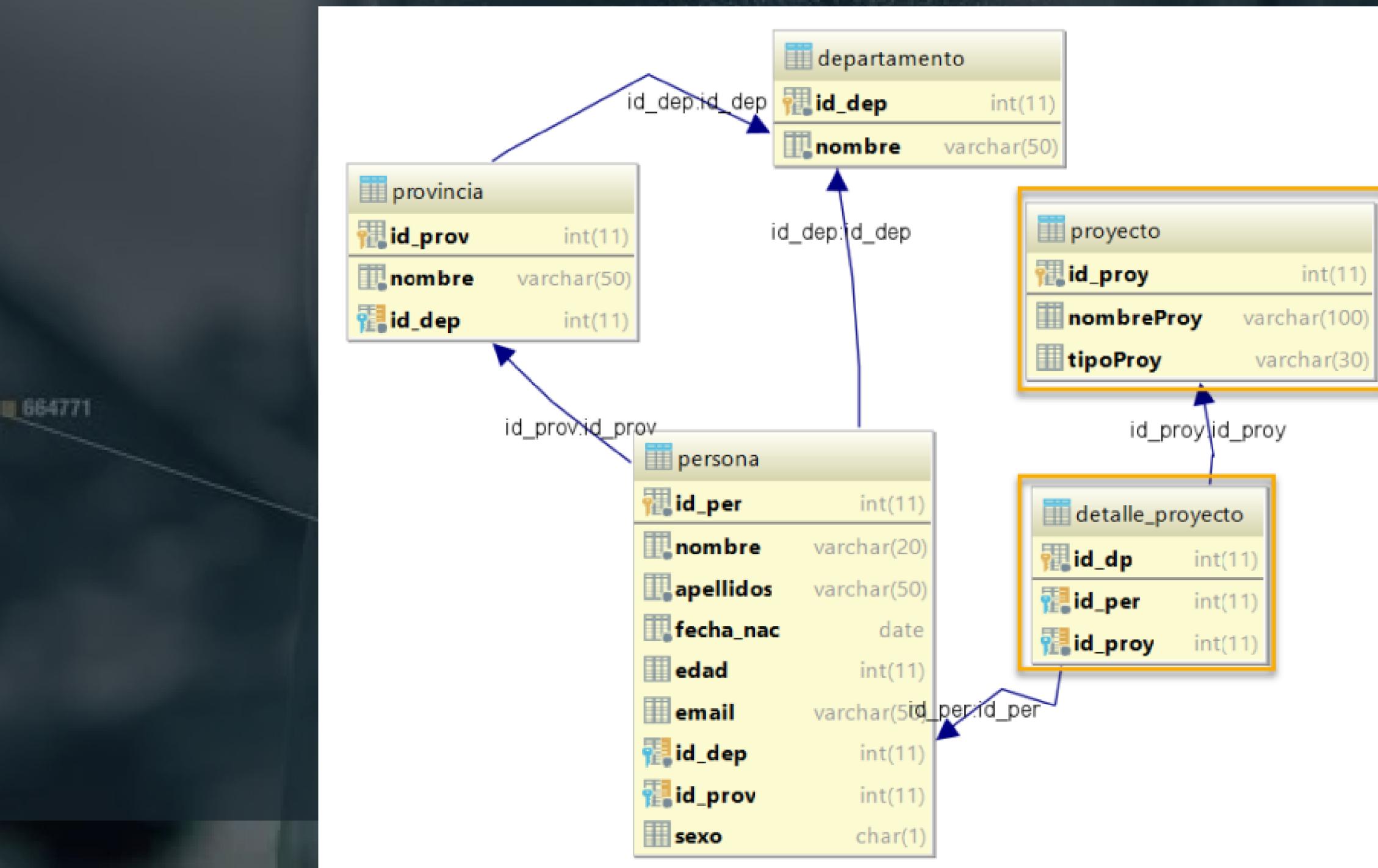
A photograph of a person sitting on a concrete bench, wearing a light-colored shirt and blue jeans. They are looking down at an open book or magazine they are holding in their lap. The background shows a brick wall and some foliage.

PARTE PRACTICA

A large, semi-transparent white text 'DATA' is centered in the foreground. It is surrounded by a variety of data-related icons in shades of purple, blue, and grey, including a globe, a bar chart, a stethoscope, a mail icon, and a line graph.

DATA

9. CREAR LA SIGUIENTE BASE DE DATOS Y SUS REGISTROS.



10. CREAR UNA FUNCIÓN QUE SUME LOS VALORES DE LA SERIE FIBONACCI.

- El objetivo es sumar todos los números de la serie fibonacci desde una cadena.
- Es decir usted tendrá solo la cadena generada con los primeros N números de la serie fibonacci y a partir de ellos deberá sumar los números de esa serie.
- Ejemplo: suma_serie_fibonacci(mi_metodo_que_retorna_la_serie(10))
 - Note que previamente deberá crear una función que retorne una cadena con la serie fibonacci hasta un cierto valor. 1. Ejemplo: 0,1,1,2,3,5,8,.....
 - Luego esta función se deberá pasar como parámetro a la función que suma todos los valores de esa serie generada.

```
'fibonacci(10)'  
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

FUNCTION QUE GENERA LA SERIE

```
'sumFibonacci(10)'  
88
```

FUNCTION QUE SUMA LA SERIE

10. CREAR UNA FUNCIÓN QUE SUME LOS VALORES DE LA SERIE FIBONACCI.

- Adjuntar el código SQL generado y una imagen de su correcto funcionamiento.

```
CREATE OR REPLACE FUNCTION FIBO(LIMITE INT)
RETURNS TEXT
BEGIN
    DECLARE RESP TEXT DEFAULT '';
    DECLARE a int default 0;
    DECLARE b int default 1;
    DECLARE c int default 0;
    DECLARE CONT INT DEFAULT 0;

    WHILE CONT < LIMITE DO
        SET RESP = CONCAT(RESP,c,',');
        SET a = b;
        SET b = c;
        SET c = a + b;
        SET CONT = CONT + 1;
    end while;
    RETURN RESP;
end;
```

```
CREATE OR REPLACE FUNCTION SUMAFIBO(SERIE TEXT)
RETURNS INT
BEGIN
    DECLARE SUMA INT DEFAULT 0;
    DECLARE FIBO TEXT;
    DECLARE COMAAUX INT;

    WHILE SERIE != '' DO
        SET COMAAUX = LOCATE(',',SERIE);
        IF COMAAUX = 0 THEN
            SET FIBO = SERIE;
            SET SERIE = '';
        ELSE
            SET FIBO = SUBSTRING(SERIE,1,COMAAUX-1);
            SET SERIE = SUBSTRING(SERIE,COMAAUX+1);
        end if;
        SET SUMA = SUMA + CAST(FIBO AS INT);
    end while;
    RETURN SUMA;
end;
```

The screenshot shows a database interface with two queries and their results. The first query is 'FIBO(10)', which returns the sequence of Fibonacci numbers from 0 to 34. The second query is 'SUMAFIBO(FIBO(10))', which returns the sum of these values, 88. The interface includes tabs for Output, FIBO(10):text, and SUMAFIBO(HB).

```
1 0,1,1,2,3,5,8,13,21,34,
1 88
```

11. MANEJO DE VISTAS.

- Crear una consulta SQL para lo siguiente.
- La consulta de la vista debe reflejar como campos:
 1. nombres y apellidos concatenados
 2. la edad
 3. fecha de nacimiento.
- 4. Nombre del proyecto
- Obtener todas las personas del sexo femenino que hayan nacido en el departamento de El Alto en donde la fecha de nacimiento sea:
 1. fecha_nac = '2000-10-10'

LA CONSULTA GENERADA PREVIAMENTE CONVERTIR EN UNA VISTA

11. MANEJO DE VISTAS.

```
CREATE OR REPLACE VIEW DATOS AS  
SELECT CONCAT(per.nombre, ' ', per.apellido) as fullname,  
per.edad as edad,  
per.fecha_nac as fecha_de_nacimiento,  
P.nombre_proy AS nombre_del_proyecto  
FROM PERSONA AS per  
inner join DETALLE_DE_PROYECTO AS DDP on per.id_per = DDP.id_per  
INNER JOIN PROYECTO AS P on DDP.id_proy = P.id_proy  
INNER JOIN DEPARTAMENTO AS D on per.id_dep = D.id_dep  
where per.sexo = 'F' and D.nombre = 'La paz' and '2000-10-10';##
```

WHERE	ORDER BY
1 Maria Lopez	fullname, edad, fecha_de_nacimiento, nombre_del_proyecto

22 2000-10-04 Proyecto de educación digitales

12. MANEJO DE TRIGGERS I.

- Crear TRIGGERS Before or After para INSERT y UPDATE aplicado a la tabla PROYECTO
 - Deberá de crear 2 triggers minimamente.
- Agregar un nuevo campo a la tabla PROYECTO.
 - El campo debe llamarse ESTADO 6
- Actualmente solo se tiene habilitados ciertos tipos de proyectos.
 - EDUCACION, FORESTACION y CULTURA
- Si al hacer insert o update en el campo tipoProy llega los valores EDUCACION, FORESTACIÓN o CULTURA, en el campo ESTADO colocar el valor ACTIVO. Sin embargo si se llega a un tipo de proyecto distinto colocar INACTIVO
- Adjuntar el código SQL generado y una imagen de su correcto funcionamiento.

12. MANEJO DE TRIGGERS I.

```
CREATE or replace TABLE AUDITORIA(
    id_proy int,
    nombre_proy_anteriores varchar(100),
    nombre_proy_despues varchar(100),
    tipo_proy_anteriores varchar(30),
    tipo_proy_despues varchar(30),
    fecha date
);

CREATE OR REPLACE TRIGGER TRIGGER1
before insert
on PROYECTO
for each row
begin
    insert AUDITORIA(id_proy, nombre_proy_anteriores, nombre_proy_despues, tipo_proy_anteriores, tipo_proy_despues, fecha)
    values(new.id_proy,'no hay anteriores datos',new.nombre_proy,'no hay anteriores datos',new.tipo_proy,now());
end;
```

#	id_proy	nombre_proy_anteriores	nombre_proy_despues	tipo_proy_anteriores	tipo_proy_despues	fecha
1	1	Carretera	Carreteras	Infraestructura	Infraestructura	2023-06-07
2	6	no hay anteriores datos	Festival Cultural	no hay anteriores datos	Educacion	2023-06-07

12. MANEJO DE TRIGGERS I.

```
CREATE OR REPLACE TRIGGER TRIGGER2
before update
on PROYECTO
for each row
begin
    insert AUDITORIA(id_proy, nombre_proy_anteriores, nombre_proy_despues, tipo_proy_anteriores, tipo_proy_despues, fecha)
    values(new.id_proy,new.nombre_proy,old.nombre_proy,new.tipo_proy,old.tipo_proy,now());
end;
```



id_proy	nombre_proy_anteriores	nombre_proy_despues	tipo_proy_anteriores	tipo_proy_despues	fecha
1	Carretera	Carreteras	Infraestructura	Infraestructura	2023-06-07
2	no hay anteriores datos	Festival Cultural	no hay anteriores datos	Educacion	2023-06-07

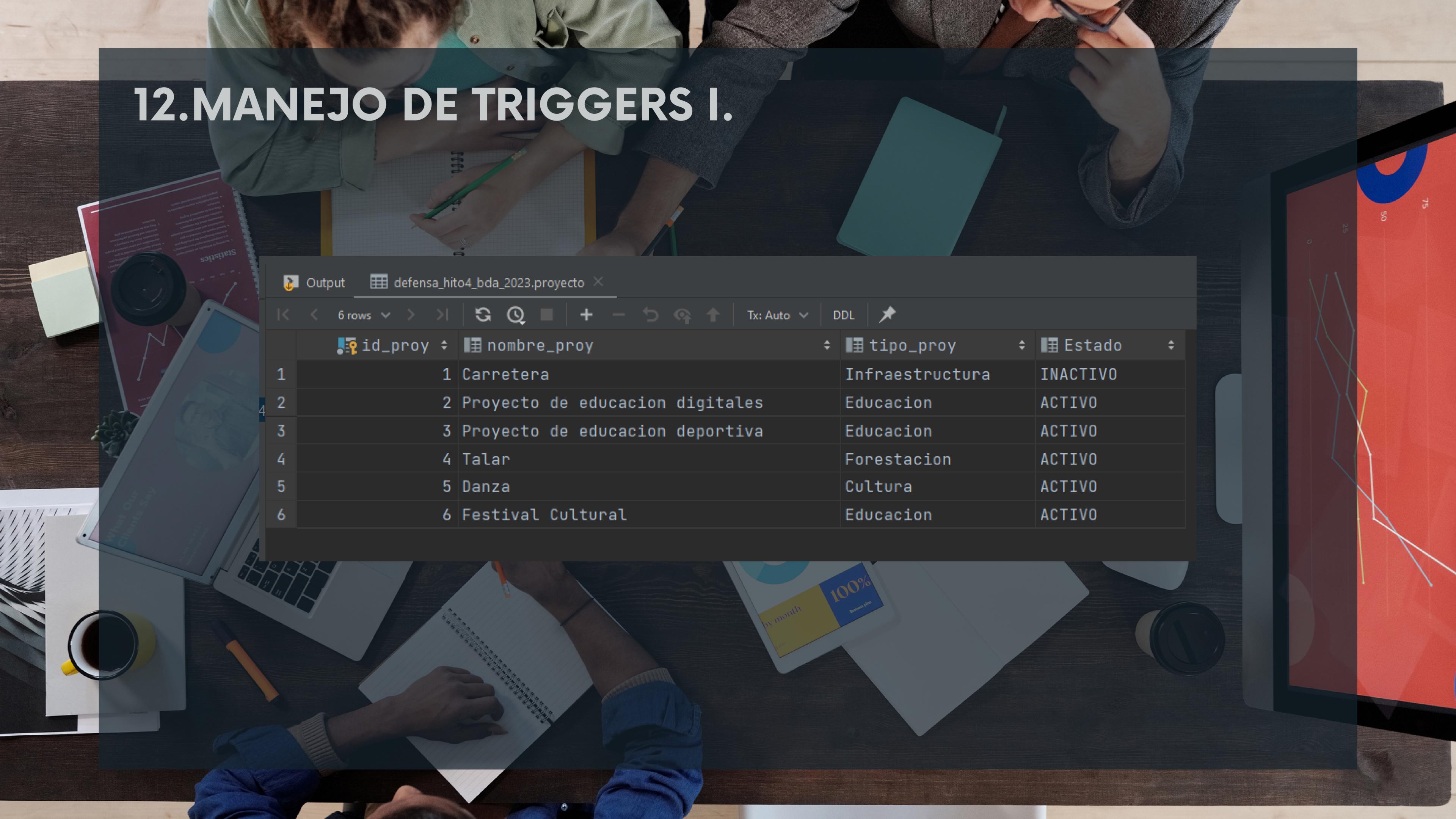
12. MANEJO DE TRIGGERS I.

```
alter table PROYECTO
add column Estado varchar(10);

# creando trigger

CREATE OR REPLACE TRIGGER ACTIVO
BEFORE insert
ON PROYECTO
FOR EACH ROW
BEGIN
    IF NEW.tipo_proy = 'Educacion' or NEW.tipo_proy = 'Forestacion' or NEW.tipo_proy = 'Cultura' then
        set new.Estado = 'ACTIVO';
    ELSE
        SET NEW.Estado = 'INACTIVO';
    end if;
end;
```

12. MANEJO DE TRIGGERS I.



A screenshot of a MySQL Workbench interface showing a table named "defensa_hito4_bda_2023.proyecto". The table has four columns: "id_proy", "nombre_proy", "tipo_proy", and "Estado". The data consists of six rows:

	id_proy	nombre_proy	tipo_proy	Estado
1	1	Carretera	Infraestructura	INACTIVO
2	2	Proyecto de educacion digitales	Educacion	ACTIVO
3	3	Proyecto de educacion deportiva	Educacion	ACTIVO
4	4	Talar	Forestacion	ACTIVO
5	5	Danza	Cultura	ACTIVO
6	6	Festival Cultural	Educacion	ACTIVO

13. MANEJO DE TRIGGERS II

- El trigger debe de llamarse calculaEdad.
- El evento debe de ejecutarse en un BEFORE INSERT.
- Cada vez que se inserta un registro en la tabla PERSONA, el trigger debe de calcular la edad en función a la fecha de nacimiento.
- Adjuntar el código SQL generado y una imagen de su correcto funcionamiento.

13. MANEJO DE TRIGGERS II

```
CREATE OR REPLACE TRIGGER EDAD
BEFORE INSERT
ON PERSONA
FOR EACH ROW
BEGIN
    SET NEW.edad = TIMESTAMPDIFF(YEAR,NEW.fecha_nac,CURDATE());
end;
```

	id_per	nombre	apellido	fecha_nac	edad	email	id_dep	id_prov	sexo
1	1	Juan	Perez	1996-05-03	27	Juanperez@gmial.com	2	2	M
2	2	Maria	Lopez	2000-10-04	22	Maria@gmial.com	1	1	F
3	3	Juan	Alarcon	2002-06-02	21	Juan@gmail.com	1	1	M
4	4	Jhoana	Mamani	2004-09-04	18	JHoana@gmail.com	2	2	F

14. MANEJO DE TRIGGERS III.

- Crear otra tabla con los mismos campos de la tabla persona(Excepto el primary key id_per).
 - No es necesario que tenga PRIMARY KEY.
- Cada vez que se haga un INSERT a la tabla persona estos mismos valores deben insertarse a la tabla copia.
- Para resolver esto deberá de crear un trigger before insert para la tabla PERSONA.
- Adjuntar el código SQL generado y una imagen de su correcto funcionamiento.

14. MANEJO DE TRIGGERS III.

```
CREATE TABLE NUEVOS(
    nombre varchar(20),
    apellido varchar(50),
    fecha_nac date,
    edad int,
    email varchar(50),
    id_dep int,
    id_prov int,
    sexo char(1)
);

#CREAR TRIGGER
CREATE OR REPLACE TRIGGER PERSONANUEVOS
BEFORE INSERT
ON PERSONA
FOR EACH ROW
BEGIN
    INSERT INTO NUEVOS(NOMBRE, APELLIDO, FECHA_NAC, EDAD, EMAIL, ID_DEP, ID_PROV, SEXO)
        VALUES(new.nombre,new.apellido,new.fecha_nac,new.edad,new.email,new.id_dep,new.id_prov,new.sexo);
end;
```

	nombre	apellido	fecha_nac	edad	email	id_dep	id_prov	sexo
1	Jhoana	Mamani	2010-09-16	12	JHoana@gmail.com	2	2	F

15. CREAR UNA CONSULTA SQL QUE HAGA USO DE TODAS LAS TABLAS

- La consulta generada convertirlo a VISTA

```
CREATE OR REPLACE VIEW DATOS_GENERALESV1 AS
    SELECT CONCAT(PER.nombre, ' ', PER.apellido) AS FULLNAME,
           PER.edad AS EDAD,
           CONCAT(DEP.nombre, ': ', PV.nombre) AS UBICACION,
           PRO.nombre_proy AS PROYECTO
      FROM PERSONA AS PER
     inner join DETALLE_DE_PROYECTO AS DE on per.id_per = DE.id_per
    INNER JOIN PROYECTO AS PRO on DE.id_proy = PRO.id_proy
    INNER JOIN DEPARTAMENTO AS DEP on per.id_dep = DEP.id_dep
    INNER JOIN PROVINCIA AS PV on DEP.id_dep = PV.id_dep
 WHERE PER.fecha_nac > '2000-01-01';
```