Búffalos Adventure

José Ortega Gutiérrez
Curso 2023-2024

Índice

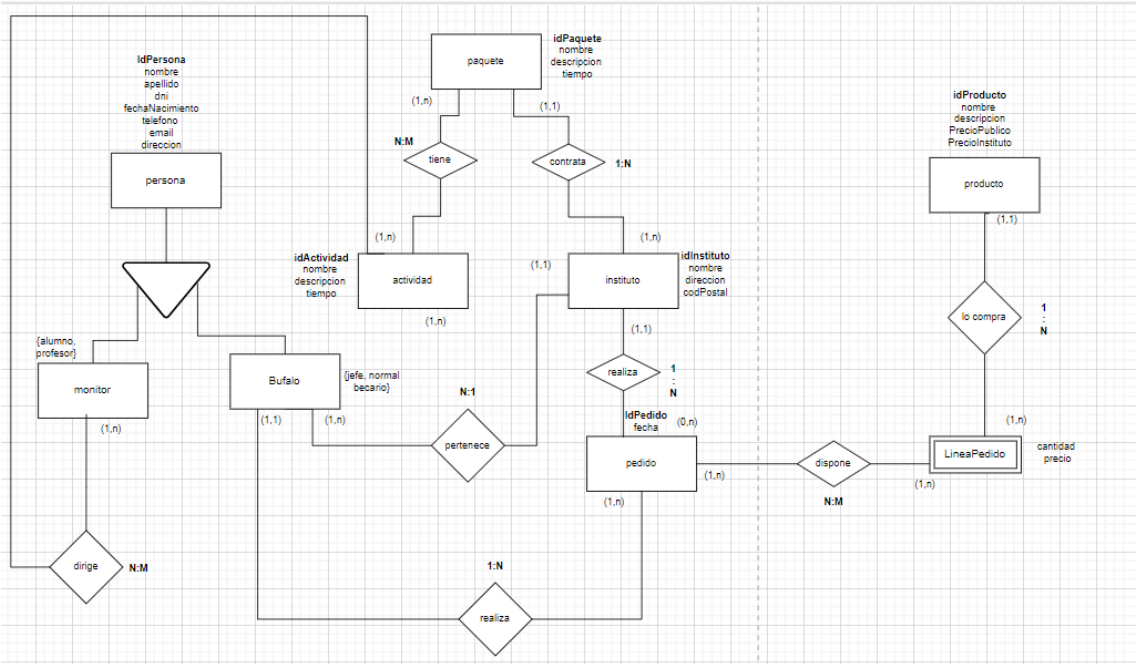
1.INTRODUCIÓN

Búffalos Adventure es una empresa que organiza campamentos multiaventura. El campamento se ofrece a institutos y colegios para viajes de fin de curso o de navidad. Está situada en los Picos de Aroche, en la sierra de Aracena. El campamento permite que puedan ir personas de todas las edades, pero está más enfocado a un público infantil.
El negocio hoy en día se encuentra funcionando perfectamente, pero les cuesta trabajo relacionarse con los institutos y proporcionar información debido a que no tienen un sistema que se lo facilite.
La única información que pueden proporcionar es yendo al mismo instituto con folletos o papeles, y eso hoy en día se queda muy obsoleto. Los niños pueden ver las actividades que van a realizar, pero no de una manera muy clara.
Búffalos nos comenta que hoy en día en internet se encuentra un paso por detrás de sus competidores. No tienen una página web a la altura de lo que su empresa hoy en día ya es y no pueden ofrecer toda la información que ellos desean. Quieren ponerse al menos a la altura de los competidores y ofrecer una información vía internet completa para que los institutos que contraten el campamento puedan ver la información de este en todo momento.

2.DESCRIPCIÓN

Como dueño del campamento quiero conocer los siguientes datos de mis **monitores**: nombre, apellidos, dirección, teléfono, DNI, fecha de nacimiento, email y tipo de monitor (para saber quién trabaja en mi campamento).
Como monitor quiero saber los siguientes datos de los **niños** que van a venir al campamento: nombre, apellidos, dirección, teléfono, DNI, Fecha de nacimiento, email y tipo de búfalo (para saber a quién voy a monitorizar).
Como dueño del campamento quiero saber los siguientes datos de los **institutos** que vienen al campamento: nombre, dirección y código postal (para sacar valiosa información sobre la aceptación del campamento como en qué lugares se apuntan más institutos a nuestro campamento).
Como instituto y búfalo quiero saber los datos de los **productos** que venden en la tienda búfalo: nombre, precio para persona individual, precio para instituto y descripción: para comprar cosas de recuerdo o para llevarla al campamento.
Como instituto y búfalo quiero saber los datos de las **actividades** que se realizan en el campamento: nombre, descripción y duración:(para decidir si quiero ir o no).
Como instituto y búfalo quiero saber la información de los distintos **paquetes** que hay y lo que ofrecen como: nombre, descripción y precio (para contratar uno u otro).
Como instituto y búfalo quiero tener un resumen del **pedido** que realizo cuando compro para tener una garantía: id de pedido y fecha.
Como **monitor jefe** quiero que haya exactamente 2 monitores por actividad, para repartir mejor el trabajo y tambien quiero que no pueda haber 2 monitores becarios dirigiendo la misma actividad (para que haya algo de experiencia a la hora de dirigir una actividad). Tambien quiero que no haya dos actividades para el mismo instituto que se hagan a la vez para evitar posibles errores.
Como **dueño del campamento** quiero que un mismo instituto no pueda contratar 2 paquetes diferentes a la vez para evitar equivocaciones y tambien quiero que no se elimine ningún instituto hasta que no haya pagado todo lo que debe para que no perdamos dinero y tengamos un recuento perfecto de todo nuestro dinero

3.MODELO ENTIDAD RELACIÓN



4.MODELO RELACIONAL EXTENDIDO

-Monitores :(idPersona, tipoMonitor)
(PK):idPersona
(FK): idPersona

-Buffalos:(IdPersona, tipoBuffalo, IdInstituto)
(PK):idPersona
(FK):idPersona REFERENCIA persona(IdPersona)
(FK):IdInstituto REFERENCIA Instituto(IdInstituto)

-Personas: (IdPersona, nombre, apellidos, dni, fechaNacimiento, teléfono, email, dirección)
(PK):idPersona

-Paquetes:(IdPaquete, nombre, descripción, precio)
(PK):IdPaquete

-Actividades:(IdActividad, nombre, descripcion, tiempo)
(PK):IdActividad

-ActividadesPaquetes:(IdPaquete, IdActividad, IdActividadPaquete)
(PK): IdPaquete, IdActividad
(FK): IdPaquete REFERENCIA Paquete(IdPaquete)
(FK): IdActividad REFERENCIA Actividad (IdActividad)

-Institutos:(IdInstituto, nombre, dirección,codPostal,IdPaquete)
(PK):IdInstituto
(FK):IdPaquete REFERENCIA Paquete(IdPaquete)

-Productos:(IdProducto, nombre, descripción, precioPVP, precioInstitutos)
(PK):IdProducto

-Pedidos:(IdPedido, fecha, IdBuffalo, IdInstituto)
(PK):IdPedido
(FK): IdBuffalo REFERENCIA Buffalo(IdBuffalo)
(FK): IdInstituto REFERENCIA Instituto(IdInstituto)

-LineaDePedidos:(IdProductosPedidos, IdProducto, IdPedido,cantidad, precio)
(PK): IdProducto,IdPedido

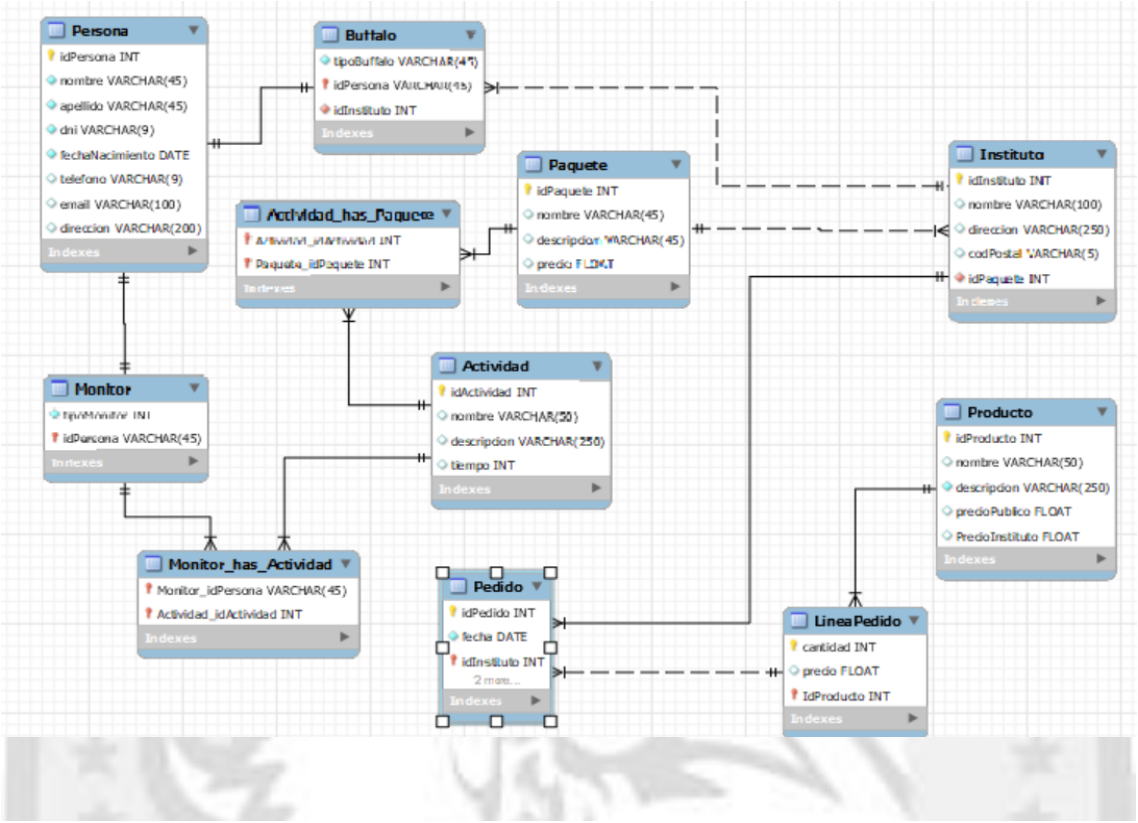
(FK): IdProducto REFERENCIA Producto (IdProducto)
(FK): IdPedido REFERENCIA Pedido(IdPedido)

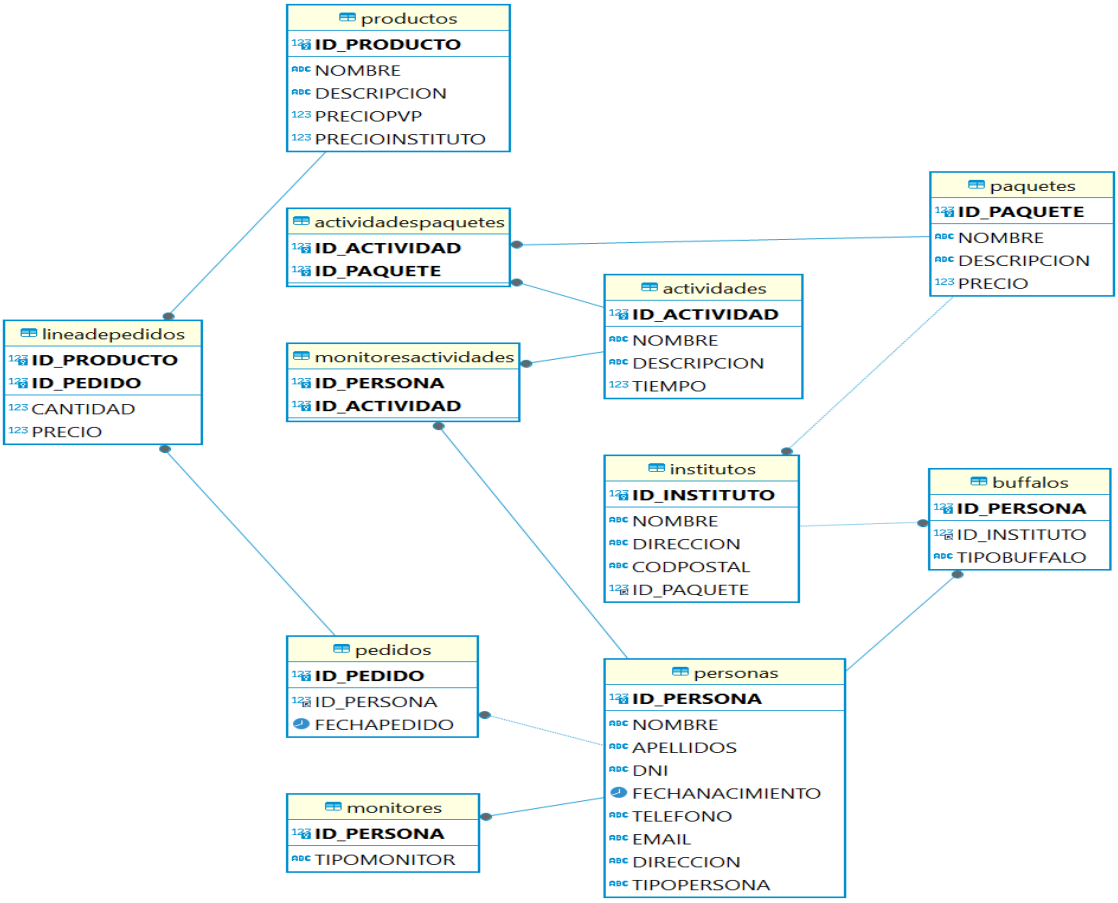
-MonitoresActividades:(IdMonitorActividad, IdPersona, IdActividad)
(PK): IdMonitor,IdActividad

(FK): IdMonitor REFERENCIA Monitor (IdMonitor)

(FK): IdActividad REFERENCIA Actividad(IdActividad)

5.MODELO RELACIONAL





5.SCRIPT CÓDIGO SQL

A partir del modelo relacional hecho en el anterior punto, realizamos el paso a tabla gracias al programa **mysql workbench** y dentro de este enlace se encuentra las tablas en mysql hechas.

6.CARGA MASIVA DE DATOS

Para realizar la carga masiva de datos, la hicimos en nuestra plataforma moddle los csv de cada tabla buscando datos de varias webs como chatgpt, el centro de estadísticas e imaginándote productos y relacionarlos en condiciones. Luego de tener cada csv lo importamos a nuestro programa Dbeaver para cargarla en la tabla relacionada con sus datos. Dentro de este enlace se encuentra la carga de datos que metimos en nuestro proyecto

7.CONSULTAS MULTITABLAS/SUBCONSULTA

1.Obtener la cantidad total de productos vendidos por cada instituto, mostrando el nombre del instituto y la suma de las cantidades de productos vendidos.

```
SELECT I.NOMBRE AS INSTITUTO, SUM(LP.CANTIDAD) AS TOTAL_PRODUCTOS_VENDIDOS
FROM INSTITUTOS I
INNER JOIN BUFFALOS B ON I.ID_INSTITUTO = B.ID_INSTITUTO
INNER JOIN PEDIDOS PE ON B.ID_PERSONA = PE.ID_PERSONA
INNER JOIN LINEADEPEDIDOS LP ON PE.ID_PEDIDO = LP.ID_PEDIDO
GROUP BY I.NOMBRE;
```

	INSTITUTO	TOTAL_PRODUCTOS_VENDIDOS
1	San Fernando	5
2	Ciudad de Almería	2
3	Cardenal Cisneros	16
4	Las Marinas	3
5	Antonio Relaño	6
6	Abdera	1
7	Velázquez	10
8	Juan XXIII	23
9	Gaviota	34
10	Virgen del Saliente	8

2. Contar la cantidad de monitores por cada tipo de actividad que realizan, mostrando el nombre de la actividad y la cantidad de monitores.

```
SELECT A.NOMBRE AS ACTIVIDAD, COUNT(MA.ID_PERSONA) AS CANTIDAD_MONITORES
FROM ACTIVIDADES A
LEFT JOIN MONITORESACTIVIDADES MA ON A.ID_ACTIVIDAD = MA.ID_ACTIVIDAD
GROUP BY A.NOMBRE;
```

	ACTIVIDAD	CANTIDAD_MONITORES
1	Escalada	4
2	Karting	3
3	Piscina	3
4	Fubol Indoor	3
5	Tenis	3
6	Tiro con Arco	3
7	Padel	3
8	Desayuno	3
9	Almuerzo	3
10	Cena	3
11	Noche de Terror	3

3. Encontrar el instituto que tiene el precio promedio más alto de los paquetes que ofrece, mostrando el nombre del instituto y el precio promedio.

```
SELECT I.NOMBRE AS INSTITUTO, AVG(P.PRECIO) AS PRECIO_PROMEDIO_PAQUETE
FROM INSTITUTOS I
INNER JOIN PAQUETES P ON I.ID_PAQUETE = P.ID_PAQUETE
GROUP BY I.NOMBRE
ORDER BY PRECIO_PROMEDIO_PAQUETE DESC
LIMIT 1;
```

	INSTITUTO	PRECIO_PROMEDIO_PAQUETE
1	Antonio Relaño	85

4. Obtener los nombres de los monitores que han adquirido productos con un precio superior al precio promedio de todos los productos, mostrando el nombre completo del monitor

```
SELECT DISTINCT P.NOMBRE, P.APELLIDOS
FROM PERSONAS P
INNER JOIN MONITORES M ON P.ID_PERSONA = M.ID_PERSONA
INNER JOIN PEDIDOS PE ON P.ID_PERSONA = PE.ID_PERSONA
INNER JOIN LINEADEPEDIDOS LP ON PE.ID_PEDIDO = LP.ID_PEDIDO
INNER JOIN PRODUCTOS PR ON LP.ID_PRODUCTO = PR.ID_PRODUCTO
WHERE PR.PRECIOPVP > (SELECT AVG(PRECIOPVP) FROM PRODUCTOS);
```

	NOMBRE	APELLIDOS
1	Jorge	Guzmán Pérez
2	Mario	Domínguez Serna
3	Elena	Pérez Gutiérrez
4	Carlos	Ruiz Montana
5	Adrián	López Romero
6	Raúl	Puerta López
7	Santiago	Carmona Ruíz

5. Identificar el instituto que ha realizado la mayor cantidad de pedidos en un mes específico, mostrando el nombre del instituto y la cantidad de pedidos realizados.

```
SELECT I.NOMBRE AS INSTITUTO, COUNT(PE.ID_PEDIDO) AS CANTIDAD_PEDIDOS
FROM INSTITUTOS I
INNER JOIN BUFFALOS B ON I.ID_INSTITUTO = B.ID_INSTITUTO
INNER JOIN PEDIDOS PE ON B.ID_PERSONA = PE.ID_PERSONA
WHERE MONTH(PE.FECHAPEDIDO) = 3  -- Cambiar el número del mes según
corresponda
GROUP BY I.NOMBRE
ORDER BY CANTIDAD_PEDIDOS DESC
LIMIT 1;
```

	INSTITUTO	CANTIDAD_PEDIDOS
1	Velázquez	1

8.VISTAS

1.Vista **cantidad_monitores** a través de Contar la cantidad de monitores por cada tipo de actividad que realizan, mostrando el nombre de la actividad y la cantidad de monitores.

```
create view cantidad_monitores as
SELECT A.NOMBRE AS ACTIVIDAD, COUNT(MA.ID_PERSONA) AS CANTIDAD_MONITORES
FROM ACTIVIDADES A
LEFT JOIN MONITORESACTIVIDADES MA ON A.ID_ACTIVIDAD = MA.ID_ACTIVIDAD
GROUP BY A.NOMBRE;
```

	ACTIVIDAD	CANTIDAD_MONITORES
1	Escalada	4
2	Karting	3
3	Piscina	3
4	Fubol Indoor	3
5	Tenis	3
6	Tiro con Arco	3
7	Padel	3
8	Desayuno	3
9	Almuerzo	3
10	Cena	3
11	Noche de Terror	3

2.vista de **cantidad_pedidos** para identificar el instituto que ha realizado la mayor cantidad de pedidos en un mes específico, mostrando el nombre del instituto y la cantidad de pedidos realizados.

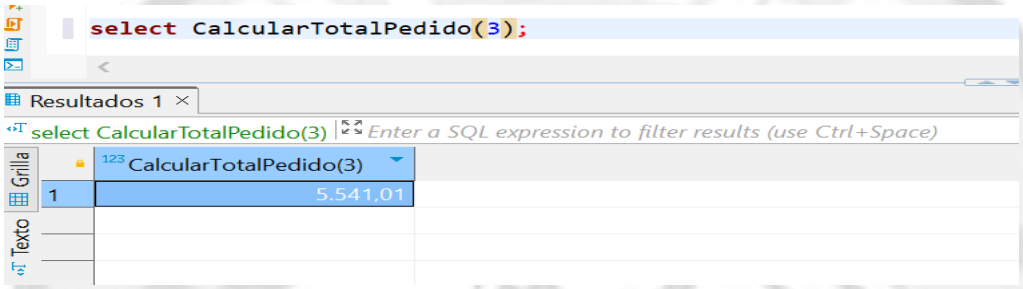
```
create view cantidad_pedidos as
SELECT I.NOMBRE AS INSTITUTO, COUNT(PE.ID_PEDIDO) AS CANTIDAD_PEDIDOS
FROM INSTITUTOS I
INNER JOIN BUFFALOS B ON I.ID_INSTITUTO = B.ID_INSTITUTO
INNER JOIN PEDIDOS PE ON B.ID_PERSONA = PE.ID_PERSONA
WHERE MONTH(PE.FECHAPEDIDO) = 3  -- Cambiar el número del mes según
corresponda
GROUP BY I.NOMBRE
ORDER BY CANTIDAD_PEDIDOS DESC
LIMIT 1;
```

	INSTITUTO	CANTIDAD_PEDIDOS
1	Velázquez	1

9.FUNCIONES Y PROCEDIMIENTOS

1.Funcion que calcule el precio total de un pedido

```
delimiter $$
CREATE FUNCTION CalcularTotalPedido(ID_PEDIDO INT)
RETURNS FLOAT
deterministic
BEGIN
    DECLARE Total FLOAT;
    SELECT SUM(CANTIDAD * PRECIO) INTO Total
    FROM LINEADEPEDIDOS
    WHERE ID_PEDIDO = ID_PEDIDO;
    RETURN Total;
END $$
delimiter ;
```



2. Función calculará el descuento aplicado a un pedido basado en el total de la compra y un porcentaje de descuento proporcionado.

```
delimiter $$
CREATE FUNCTION CalcularTotalConDescuento(
    ID_PEDIDO INT,
    PorcentajeDescuento FLOAT
)
RETURNS FLOAT
deterministic
BEGIN
    DECLARE Total FLOAT;
    DECLARE TotalConDescuento FLOAT;

    -- Calcular el total del pedido usando la función existente
CalcularTotalPedido
      select CalcularTotalPedido(ID_PEDIDO) into Total;

    -- Aplicar el descuento
    SET TotalConDescuento = Total - (Total * PorcentajeDescuento / 100);

    RETURN TotalConDescuento;
end$$
delimiter ;
```

```
select CalcularTotalConDescuento(1,6.3);
```

Resultados 1

elect CalcularTotalConDescuento(1,6.3) | Enter a SQL expression to filter results (use C

	123 CalcularTotalConDescuento(1,6.3)
1	5.241,68

3.Procedimiento para listar todos los pedidos y sus totales usando la función "calcularTotalPedido"

```
DELIMITER $$
drop procedure if exists ListarPedidosYTotales$$
CREATE PROCEDURE ListarPedidosYTotales()
BEGIN
    DECLARE TotalPedidos DECIMAL(10,2);

    -- Select all orders and their totals
    SELECT P.ID_PEDIDO, P.ID_PERSONA, P.FECHAPEDIDO,
CalcularTotalPedido(P.ID_PEDIDO) AS Total
    FROM PEDIDOS P;

    -- Calculate the overall total of all orders
    SELECT SUM(CalcularTotalPedido(P.ID_PEDIDO)) INTO TotalPedidos
    FROM PEDIDOS P;

    -- Display the overall total
    SELECT 'Total Pedidos:' AS Mensaje, TotalPedidos AS Total;
END $$

DELIMITER ;
```

```
call ListarPedidosYTotales();
```

pedidos 1 | Resultados 1 (2) | Estadísticas 1

call ListarPedidosYTotales() | Data filter is not supported

	ABC Mensaje	123 Total
1	Total Pedidos:	190.199,74

4.Procedimiento que muestra los detalles de un pedido específico. Este procedimiento seleccionará y mostrará la información de cada línea de pedido asociada con un pedido dado.

```
CREATE PROCEDURE MostrarDetallesPedido(IN pedido_id INT)
BEGIN
    -- Mostrar los detalles del pedido
    SELECT
        p.NOMBRE AS Producto,
        lp.CANTIDAD AS Cantidad,
        lp.PRECIO AS Precio,
        (lp.CANTIDAD * lp.PRECIO) AS Subtotal
    FROM
        LINEADEPEDIDOS lp
    JOIN
        PRODUCTOS p ON lp.ID_PRODUCTO = p.ID_PRODUCTO
    WHERE
        lp.ID_PEDIDO = pedido_id;
END $$

DELIMITER ;
```

```
call MostrarDetallesPedido(1);
```

productos(+) 1 | Estadísticas 1

call MostrarDetallesPedido(1) | Enter a SQL expression to filter results (use Ctrl+

	Producto	Cantidad	Precio	Subtotal
1	gafas bufalo	6	76,7	460,1999816895

5. Procedimiento que te muestre los productos con su id, nombre y descripción. (Cursor)

```
DELIMITER $$
drop procedure if exists MostrarProductos$$
CREATE PROCEDURE MostrarProductos()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE producto_id INT;
    DECLARE producto_nombre VARCHAR(50);
    DECLARE producto_descripcion VARCHAR(250);
      declare salida varchar(5000) default '';

    -- Declaro el cursor
    DECLARE productos_cursor CURSOR FOR
        SELECT ID_PRODUCTO, NOMBRE, DESCRIPCION
        FROM PRODUCTOS;
```

```
    -- Declaro el controlador
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = true;

      set salida=concat(salida, '----Procesando productos chachis del
            campamento:----\n');
      set salida=concat(salida,
            'Id','\t\t','nombre','\t\t\t\t\t','descripcion','\n');

  -- Abro el cursor
    OPEN productos_cursor;

    -- Obtengo las filas por cada cursor
    fetch_loop: LOOP
        FETCH productos_cursor INTO producto_id, producto_nombre,
            producto_descripcion;

        IF done THEN
            LEAVE fetch_loop;
        END IF;

        -- agrego los detalles del producto a la salida
        set salida= CONCAT(salida, producto_id, '\t\t', producto_nombre,
            '\t\t\t\t', producto_descripcion, '\n');
    END LOOP;

    -- Cierro el cursor
    CLOSE productos_cursor;

    -- muestro la salida
    SELECT salida;
END $$

DELIMITER ;
```

```
call MostrarProductos();
```

Resultados 1 | Estadísticas 1

```
call MostrarProductos()  Enter a SQL expression to filter results (use Ctrl+Space)
```

	salida
1	----Procesando productos chachis del campamento:----¶Id nombre descripcion¶1 Camiseta Buf

**Aquí le muestro como se vería la tabla más correctamente:

```
----Procesando productos chachis del campamento:----
Id              nombre                                  descripcion
1               Camiseta Buffalo                                Vistete de Buffalo
2               Camiseta Buffalo manga larga                            Vistete de Buffalo
3               sudadera bufalo                         Vistete de Buffalo
4               riñonera bufalo                         guarda las cosas
5               gafas bufalo                            protegete la vista
6               Camiseta Buffalo L                              Vistete de Buffalo
7               postal bufalo                           recuerdas hecha papel
8               llavero bufalo                          guarda tus llaves
9               gorra bufalo                            protegete de la cabeza
10              roca de la sierra                               recuerdo de la sierra
11              cantimplora                             llena el agua de la sierra
```

10. TRIGGERS

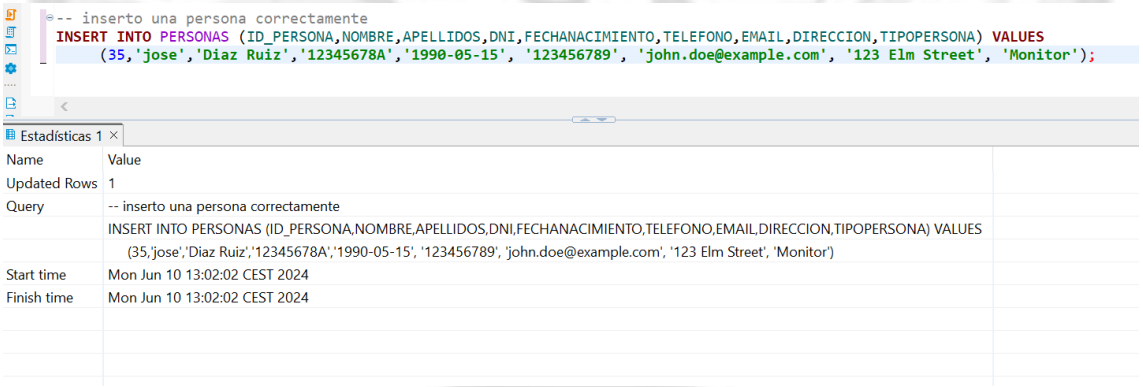
1. Trigger que se ejecutará antes de insertar un nuevo registro en la tabla PERSONAS y realiza dos validaciones:

-Comprueba que el DNI no esté duplicado.

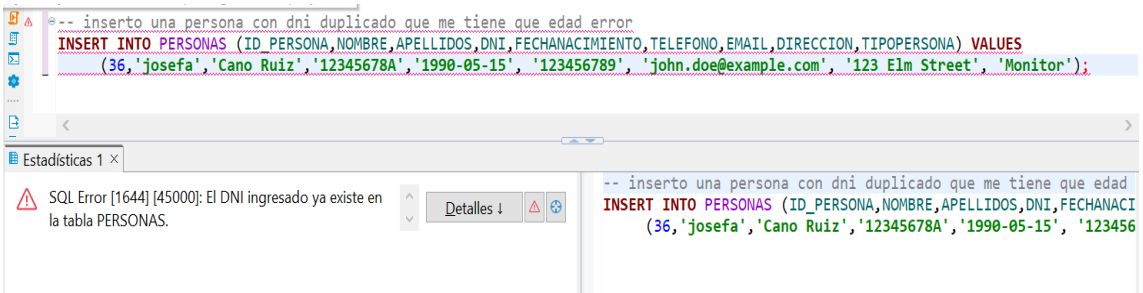
-Asegura que la fecha de nacimiento no sea anterior al 1 de enero de 1900.

```
delimiter $$
CREATE TRIGGER TR_INSERTAR_PERSONA
BEFORE INSERT ON PERSONAS
FOR EACH ROW
BEGIN
    -- Validar que el DNI no esté duplicado
    IF (SELECT COUNT(*) FROM PERSONAS WHERE DNI = NEW.DNI) > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'El DNI ingresado ya
existe en la tabla PERSONAS.';
    END IF;

    -- Validar que la fecha de nacimiento sea mayor a 1900-01-01
    IF NEW.FECHANACIMIENTO < '1900-01-01' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'La fecha de nacimiento no
puede ser anterior a 1900-01-01.';
    END IF;
end$$
delimiter ;
```

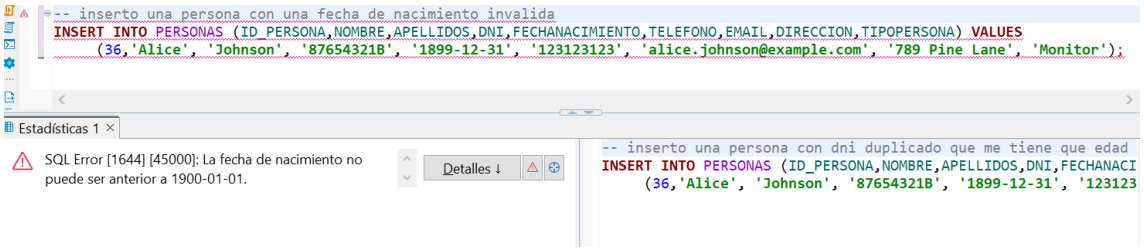
inserto una persona correctamente:



Inserto una persona con dni duplicado(me tiene que dar error):



Inserto una fecha de nacimiento invalida(me tiene que dar error):



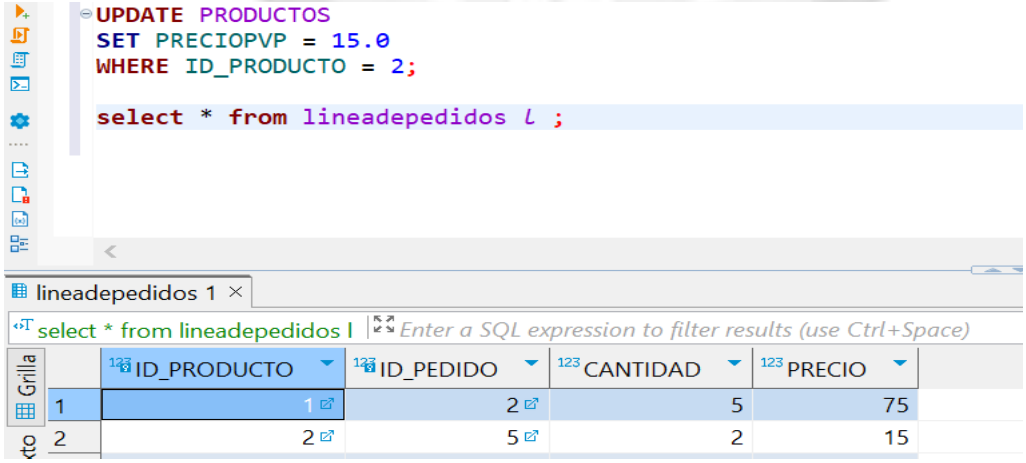
2. Trigger que actualiza el precio de los productos relacionados en la tabla LINEADEPEDIDOS cada vez que se actualiza el precio de un producto en la tabla PRODUCTOS.

```
DELIMITER $$
drop trigger if exists ActualizarPrecioProductosRelacionados$$

CREATE TRIGGER ActualizarPrecioProductosRelacionados
AFTER UPDATE ON PRODUCTOS
FOR EACH ROW
BEGIN
    IF NEW.PRECIOPVP <> OLD.PRECIOPVP THEN
        UPDATE LINEADEPEDIDOS
        SET PRECIO = NEW.PRECIOPVP
        WHERE ID_PRODUCTO = NEW.ID_PRODUCTO;
    END IF;
END $$

DELIMITER ;
```

Inserto unos productos

	ID_PRODUCTO	ID_PEDIDO	CANTIDAD	PRECIO
1	1	2	5	56,7
2	2	5	2	34,2

Compruebo que el trigger se comprueba correctamente:



11.ENLACE A GITHUB

Dentro de este enlace se encuentra los **GitHub** en 4 ficheros:

- Un fichero SQL con el esquema (miproyecto-schema.sql)
- Otro fichero SQL con los datos (miproyecto-data.sql)
- Un último fichero SQL con las consultas, vistas, funciones, procedimientos y triggers que has realizado (miproyecto-queries.sql)
- El PDF de la documentación (que será este documento).

12.VALORACIÓN PERSONAL

A lo largo de este proyecto me he encontrado con varias dificultades que, poco a poco, se han ido corrigiendo.

El primero de ellos fue encontrar una temática que se adecuara a mis objetivos a la hora de realizar el proyecto. Sin embargo, con un poco de imaginación e investigación, logré encontrar una que me gustara.

Luego, tanto el diseño del modelo entidad-relación (MER) como el paso a tablas requería un poco de creatividad, y la carga de datos demandaba bastante tiempo. Al realizar las consultas, funciones, procedimientos y triggers, pude aplicar lo que hemos aprendido en clase, añadiendo un toque de creatividad en el proceso. Esta etapa ha sido la que más he disfrutado y de la que más he aprendido, ya que estos elementos son más complejos de desarrollar.

Estoy seguro de que el proyecto puede mejorarse en varios aspectos, como en la carga de datos, desarrollando funciones y procedimientos más complejos, y quizá optimizando aún más el modelo relacional. No obstante, estoy muy contento con el resultado final del proyecto y con todo lo que he aprendido a lo largo de estos meses. Todo lo que hemos visto en clase ha sido implementado en este proyecto, lo cual me deja muy satisfecho.