ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

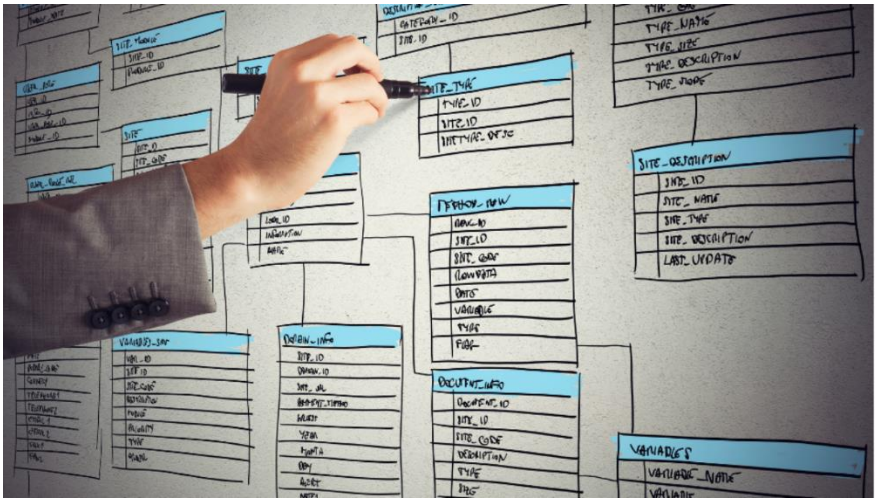
BASE DE DATOS

PROFESOR:	Ing. Yadira Franco R
PERÍODO ACADÉMICO:	2024-B

TAREA

TÍTULO:

INVESTIGACIÓN Y PRACTICA



Estudiante

Cristian Tambaco

2024-B

Link GitHub: https://github.com/CristianTambaco/PARTE_1_y_2_Tarea_Funciones_de_Usuario.git

PARTE 1 Tarea Funciones de Usuario

Tarea: Funciones de Usuario en Bases de Datos

Objetivo:

El objetivo de esta tarea es que los estudiantes aprendan a crear funciones de usuario en bases de datos

Escenario:

Vas a crear una base de datos para una tienda en línea que maneja clientes, productos, pedidos y detalles de los pedidos.

Pasos a Seguir:

1. Crear la Base de Datos y Tablas:
    - Crea una base de datos llamada `tienda_online`.

-- Crear la base de datos

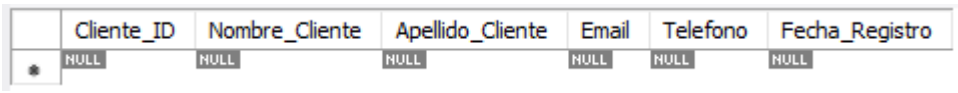
CREATE DATABASE tienda_online;

-- Seleccionar la base de datos tienda_online

USE tienda_online;

    - Dentro de la base de datos, crea las siguientes tablas:
        - Clientes: Contendrá información básica sobre los clientes (id, nombre, apellido, email, teléfono, fecha de registro).

```
-- Crear la tabla Clientes
CREATE TABLE Clientes (
    Cliente_ID INT PRIMARY KEY AUTO_INCREMENT ,
    Nombre_Cliente VARCHAR(50) NOT NULL,
    Apellido_Cliente VARCHAR(50) NOT NULL,
    Email VARCHAR(100) NOT NULL UNIQUE,
    Telefono VARCHAR(30),
    Fecha_Registro DATE NOT NULL
);
```



- Productos: Contendrá información sobre los productos (id, nombre, precio, stock, descripción).

```
-- Crear la tabla Productos
CREATE TABLE Productos (
    Producto_ID INT PRIMARY KEY AUTO_INCREMENT ,
    Nombre_Producto VARCHAR(50) NOT NULL UNIQUE,
```

```
    Precio_Producto DECIMAL(10,2) NOT NULL CHECK (Precio_Producto > 0),
    Stock INT NOT NULL CHECK (Stock >= 0),
    Descripcion TEXT
);
```

Table: productos

Columns:

Producto_ID	int AI PK
Nombre_Producto	varchar(50)
Precio_Producto	decimal(10,2)
Stock	int
Descripcion	text

	Producto_ID	Nombre_Producto	Precio_Producto	Stock	Descripcion
*	NULL	NULL	NULL	NULL	NULL

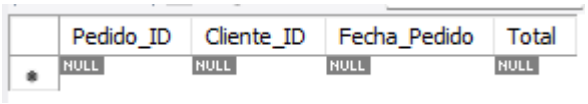
- Pedidos: Registra los pedidos realizados por los clientes (id, cliente_id, fecha del pedido, total).

```
-- Crear la tabla Pedidos
CREATE TABLE Pedidos (
    Pedido_ID INT PRIMARY KEY AUTO_INCREMENT ,
    Cliente_ID INT,
    Fecha_Pedido DATE NOT NULL,
    Total DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (Cliente_ID) REFERENCES Clientes(Cliente_ID)
);
```

Table: pedidos

Columns:

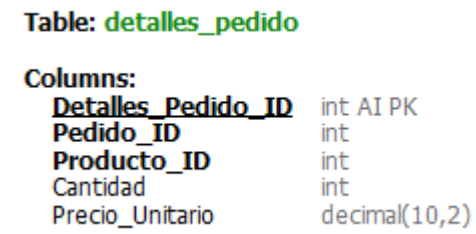
Pedido_ID	int AI PK
Cliente_ID	int
Fecha_Pedido	date
Total	decimal(10,2)



	Pedido_ID	Cliente_ID	Fecha_Pedido	Total
*	NULL	NULL	NULL	NULL

- Detalles_Pedido: Registra los detalles de cada pedido (id, pedido_id, producto_id, cantidad, precio unitario).

```
-- Crear la tabla Detalles_Pedido
CREATE TABLE Detalles_Pedido (
    Detalles_Pedido_ID INT AUTO_INCREMENT PRIMARY KEY,
    Pedido_ID INT,
    Producto_ID INT,
    Cantidad INT NOT NULL CHECK (Cantidad > 0),
    Precio_Unitario DECIMAL(10,2) NOT NULL CHECK (Precio_Unitario > 0),
    FOREIGN KEY (Pedido_ID) REFERENCES Pedidos(Pedido_ID),
    FOREIGN KEY (Producto_ID) REFERENCES Productos(Producto_ID)
);
```



	Detalles_Pedido_ID	Pedido_ID	Producto_ID	Cantidad	Precio_Unitario
*	NULL	NULL	NULL	NULL	NULL

2. Restricciones:
    - No se permiten valores nulos en campos como nombre, apellido, email, precio, y cantidad.

- Los precios deben ser positivos.
- El stock de los productos no puede ser negativo.
- Los nombres de los productos no deben repetirse.
- El email de los clientes debe ser único.

-- Insertar registros en la tabla Clientes

INSERT INTO Clientes (Nombre_Cliente, Apellido_Cliente, Email, Telefono, Fecha_Registro)

VALUES

('Josue', 'Estrada', 'josue.estrada@gmail.com', '6543246', '2022-02-14'),

('Susana', 'Cajamarca', 'susana.cajamarca@gmail.com', '5555555', '2023-03-17'),

('David', 'Mora', 'david.mora@gmail.com', '2222222', '2021-08-27');

Cliente_ID	Nombre_Cliente	Apellido_Cliente	Email	Telefono	Fecha_Registro
1	Josue	Estrada	josue.estrada@gmail.com	6543246	2022-02-14
2	Susana	Cajamarca	susana.cajamarca@gmail.com	5555555	2023-03-17
3	David	Mora	david.mora@gmail.com	2222222	2021-08-27
NULL	NULL	NULL	NULL	NULL	NULL

-- Insertar registros en la tabla Productos

INSERT INTO Productos (Nombre_Producto, Precio_Producto, Stock, Descripcion)

VALUES

('Computador', 1000.00, 65, 'Computador con procesador i9, 64GB RAM y 2048GB HDD'),

('Tablet', 500.00, 115, 'Tablet con pantalla SOLID y cámara HD de 80MP'),

('Parlantes', 170.00, 222, 'Parlantes inalámbricos Bluetooth');

Producto_ID	Nombre_Producto	Precio_Producto	Stock	Descripcion
1	Computador	1000.00	65	Computador con procesador i9, 64GB RAM y 20...
2	Tablet	500.00	115	Tablet con pantalla SOLID y cámara HD de 80MP
3	Parlantes	170.00	222	Parlantes inalámbricos Bluetooth
NULL	NULL	NULL	NULL	NULL

-- Insertar registros en la tabla Pedidos

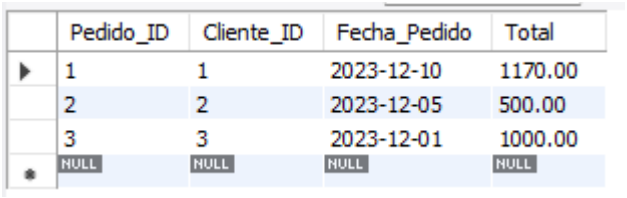
INSERT INTO Pedidos (Cliente_ID, Fecha_Pedido, Total)

VALUES

(1, '2023-12-10', 1170.00),

(2, '2023-12-05', 500.00),

(3, '2023-12-01', 1000.00);



Pedido_ID	Cliente_ID	Fecha_Pedido	Total
1	1	2023-12-10	1170.00
2	2	2023-12-05	500.00
3	3	2023-12-01	1000.00
NULL	NULL	NULL	NULL

-- Insertar registros en la tabla Detalles_Pedido

INSERT INTO Detalles_Pedido (Pedido_ID, Producto_ID, Cantidad, Precio_Unitario)

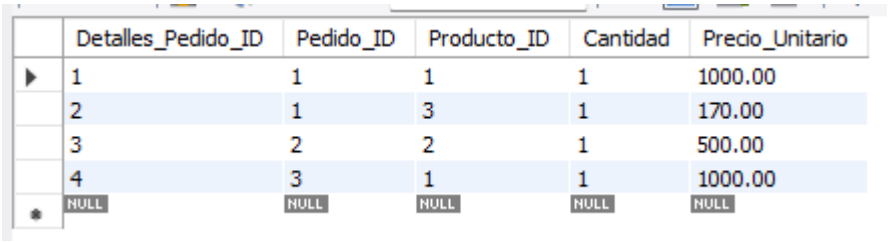
VALUES

(1, 1, 1, 1000.00),

(1, 3, 1, 170.00),

(2, 2, 1, 500.00),

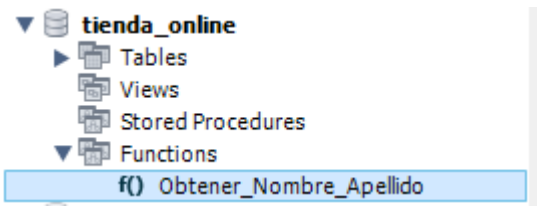
(3, 1, 1, 1000.00);



Detalles_Pedido_ID	Pedido_ID	Producto_ID	Cantidad	Precio_Unitario
1	1	1	1	1000.00
2	1	3	1	170.00
3	2	2	1	500.00
4	3	1	1	1000.00
NULL	NULL	NULL	NULL	NULL

3. Crear Funciones de Usuario
4. Función para obtener el nombre completo de un cliente:
    - Esta función debe aceptar un `cliente_id` como parámetro y devolver el nombre completo (nombre + apellido) del cliente.

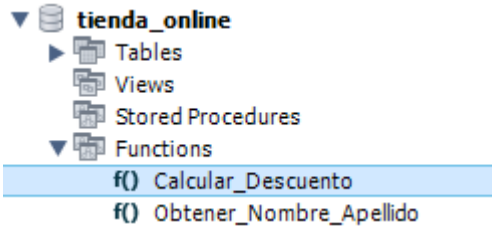
```
-- Función para obtener el nombre completo de un cliente:
-- Esta función debe aceptar un cliente_id como
-- parámetro y devolver el nombre completo (nombre + apellido) del cliente.

DELIMITER //
CREATE FUNCTION Obtener_Nombre_Apellido(p_Cliente_ID INT)
RETURNS VARCHAR(200)
DETERMINISTIC

BEGIN
    DECLARE Nombres_Apellido VARCHAR(250);

    SELECT CONCAT(Nombre_Cliente, ' ', Apellido_Cliente)
    INTO Nombres_Apellido
    FROM Clientes
    WHERE Cliente_ID = p_Cliente_ID;

    RETURN Nombres_Apellido;
END //
DELIMITER ;
```

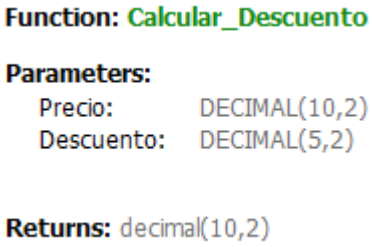


Function: Obtener_Nombre_Apellido

- Función para calcular el descuento de un producto:
  - Esta función debe aceptar el `precio` y el `descuento` como parámetros y devolver el precio con descuento.

```
-- Función para calcular el descuento de un producto:
-- Esta función debe aceptar el precio y el descuento
-- como parámetros y devolver el precio con descuento.

DELIMITER //
CREATE FUNCTION Calcular_Descuento(Precio DECIMAL(10,2), Descuento
DECIMAL(5,2))
RETURNS DECIMAL(10,2)
DETERMINISTIC

BEGIN
    DECLARE Precio_Con_Descuento DECIMAL(10,2);
    SET Precio_Con_Descuento = Precio - (Precio * Descuento / 100);
    RETURN Precio_Con_Descuento;
END //
DELIMITER ;
```

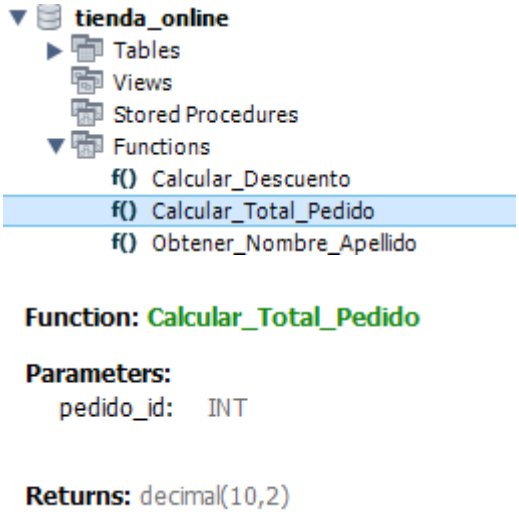




- Función para calcular el total de un pedido:
  - Esta función debe aceptar un `pedido_id` y calcular el total del pedido sumando los precios de los productos multiplicados por sus respectivas cantidades.

```
-- Función para calcular el total de un pedido:
-- Esta función debe aceptar un pedido_id y calcular
-- el total del pedido sumando los precios de los
-- productos multiplicados por sus respectivas cantidades.

DELIMITER $$
CREATE FUNCTION Calcular_Total_Pedido(pedido_id INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC

BEGIN
    DECLARE total DECIMAL(10,2) DEFAULT 0.00;
        -- Sumar el precio de cada producto * cantidad
```

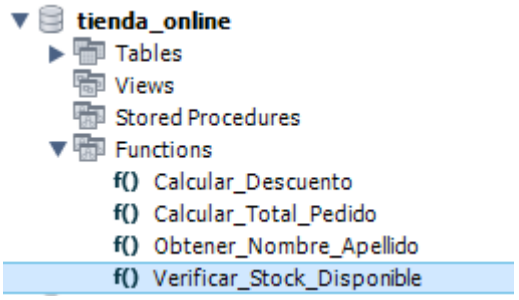
```
    SELECT SUM(dp.Cantidad * dp.Precio_Unitario)
    INTO total
    FROM Detalles_Pedido dp
    WHERE dp.Pedido_ID = pedido_id;

    RETURN total;
END $$
DELIMITER ;
```



- Función para verificar la disponibilidad de stock de un producto:
  - Esta función debe aceptar un `producto_id` y una `cantidad` como parámetros y devolver `TRUE` si el stock disponible es suficiente, de lo contrario, debe devolver `FALSE`.

```
-- Función para verificar la disponibilidad de stock de un producto:
-- Esta función debe aceptar un producto_id y una cantidad
-- como parámetros y devolver TRUE si el stock disponible
-- es suficiente, de lo contrario, debe devolver FALSE.
```

```
DELIMITER //

CREATE FUNCTION Verificar_Stock_Disponible(Producto_ID INT, Cantidad INT)

RETURNS BOOLEAN

DETERMINISTIC

BEGIN

   DECLARE Stock_Disponible INT;


   SELECT Stock INTO Stock_Disponible

   FROM Productos

   WHERE Producto_ID = Producto_ID

   LIMIT 1;


   IF Stock_Disponible >= Cantidad THEN

      RETURN TRUE;  -- 1 = Stock disponible

   ELSE

      RETURN FALSE ; -- 0 = Stock no disponible

   END IF;

END //

DELIMITER ;
```



Function: Verificar_Stock_Disponible

Parameters:

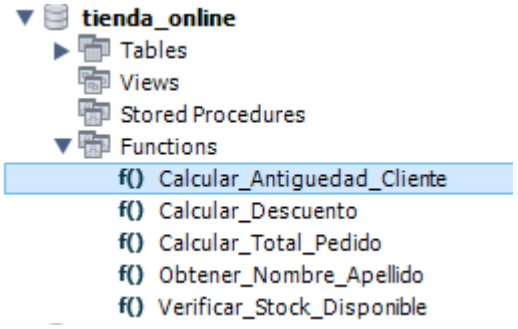
Producto_ID: INT
Cantidad: INT

Returns: tinyint(1)

- Función para calcular la antigüedad de un cliente:
  - Esta función debe aceptar un `cliente_id` y calcular la antigüedad del cliente en años a partir de la fecha de registro.

```
DELIMITER //

CREATE FUNCTION Calcular_Antiguedad_Cliente(a_cliente_id INT)

RETURNS INT

DETERMINISTIC

BEGIN

    DECLARE año_registro INT;

    DECLARE año_actual INT;

    DECLARE antiguedad INT;


    -- Obtener el año de registro del cliente

    SELECT YEAR(Fecha_Registro)

    INTO año_registro

    FROM Clientes

    WHERE Cliente_ID = a_cliente_id

    LIMIT 1;
```

```
    -- Obtener año actual
    SET año_actual = YEAR(CURDATE());

    -- Calcular antigüedad del cliente en años
    SET antiguedad = año_actual - año_registro;

    -- Retornar antigüedad en años
    RETURN antiguedad;
END //
DELIMITER ;

-- Seleccionar la función Calcular_Antiguedad_Cliente
SELECT Calcular_Antiguedad_Cliente(1) AS Años_Antigüedad;
```



Function:
Calcular_Antiguedad_Cliente

Parameters:
a_cliente_id: INT

Returns: int

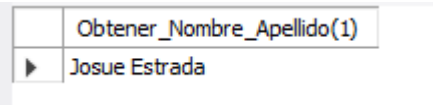
	Años_Antigüedad
▶	2

5. Consultas de Uso de Funciones:

- Consulta para obtener el nombre completo de un cliente dado su `cliente_id`.

-- Seleccionar la función Obtener_Nombre_Apellido

SELECT Obtener_Nombre_Apellido(1);

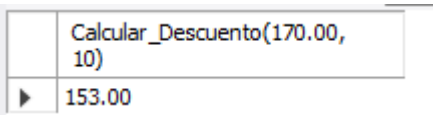


Obtener_Nombre_Apellido(1)
Josue Estrada

- Consulta para calcular el descuento de un producto dado su `precio` y un `descuento` del 10%.

-- Seleccionar la función Calcular_Descuento

SELECT Calcular_Descuento(170.00, 10);

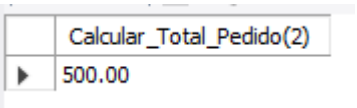


Calcular_Descuento(170.00, 10)
153.00

- Consulta para calcular el total de un pedido dado su `pedido_id`.

-- Seleccionar la función Calcular_Total_Pedido

SELECT Calcular_Total_Pedido(2);



Calcular_Total_Pedido(2)
500.00

- Consulta para verificar si un producto tiene suficiente stock para una cantidad solicitada.

-- Seleccionar la función Verificar_Stock_Disponible con ID del Producto y la cantidad
-- True=1, False=0

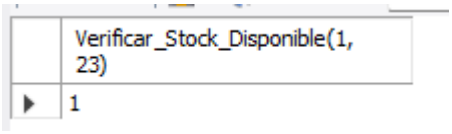
SELECT Verificar_Stock_Disponible(1, 23);

-- Seleccionar la función Verificar_Stock_Disponible con ID del Producto y la cantidad

-- True=1, False=0

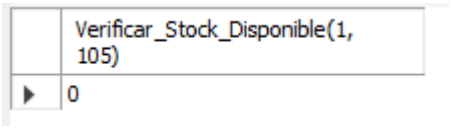
SELECT Verificar_Stock_Disponible(1, 105);

- Si es verdadero = 1



	Verificar_Stock_Disponible(1, 23)
▶	1

- Si es falso = 0



	Verificar_Stock_Disponible(1, 105)
▶	0

PARTE 2

Aprendizaje de Funciones SQL: Creación, Análisis y Ejecución

Objetivo:

El objetivo de esta actividad es aprender a crear y utilizar funciones definidas por el usuario en SQL, analizar su estructura y lógica, y practicar la creación de tablas y consultas con funciones personalizadas. También se incluirán ejemplos prácticos para mostrar cómo utilizar estas funciones en un contexto real.

Instrucciones:

1. **Transcripción y análisis del código SQL.**
2. **Creación de las tablas necesarias para almacenar los datos.**
3. **Ejecución de las funciones SQL creadas y captura de los resultados.**
4. **Explicación detallada de cada línea del código.**

SUBIR A GIT HUB EL SCRIPT Y EL PDF

EJERCICIO 1

```
CREATE FUNCTION CalcularTotalOrden(id_orden INT)
RETURNS DECIMAL(10, 2)
DETERMINISTIC
BEGIN
    DECLARE total DECIMAL(10, 2);
    DECLARE iva DECIMAL(10, 2);

    SET iva = 0.15;

    SELECT SUM(P.precio * O.cantidad) INTO total
    FROM Ordenes O
    JOIN Productos P ON O.producto_id = P.ProductoID
    WHERE O.OrdenID = id_orden;

    SET total = total + (total * iva);

    RETURN total;
END $$

DELIMITER ;
```

-- Se crea y se selecciona la base de datos.
CREATE DATABASE Parte2;
USE Parte2;

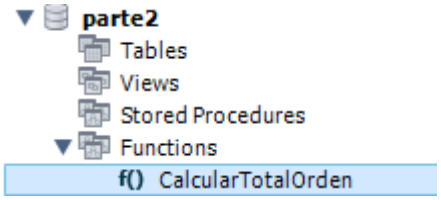
-- EJERCICIO 1
-- Se crea la función
DELIMITER $$
CREATE FUNCTION CalcularTotalOrden(id_orden INT) -- LA función CalcularTotalOrden tiene un parámetro id_orden de tipo entero.
RETURNS DECIMAL(10,2) -- La función devolverá un valor tipo DECIMAL de 10 dígitos y 2 decimales.
DETERMINISTIC
BEGIN
-- Se declaran dos variables: el total y el iva, de tipo DECIMAL.
DECLARE total DECIMAL(10, 2);
DECLARE iva DECIMAL(10, 2);

```
    SET iva = 0.15;  -- Se asigna el valor de 15% a la variable iva.

    SELECT SUM(P.precio * O.cantidad) INTO total -- Se calcular el precio total
multiplicando el precio de cada producto por la cantidad.
    FROM Ordenes O
    JOIN Productos P ON O.ProductoID = p.ProductoID
    WHERE O.OrdenID = id_orden;

    SET total = total + (total * iva);  -- Se suma el total con el iva.

    RETURN total;   -- Devuelve el valor total, calculado con el IVA.
END $$
DELIMITER ;
```

Function: CalcularTotalOrden

Parameters:
id_orden: INT

Returns: decimal(10,2)

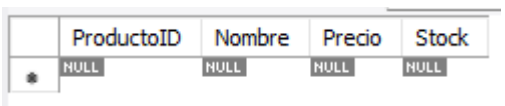



```
-- Crear la tabla Productos
CREATE TABLE Productos (
    ProductoID INT AUTO_INCREMENT PRIMARY KEY,
    Nombre VARCHAR(255) NOT NULL,
    Precio DECIMAL(10, 2) NOT NULL,
    Stock INT NOT NULL
);
```

Table: productos

Columns:

ProductoID	int AI PK
Nombre	varchar(255)
Precio	decimal(10,2)
Stock	int



	ProductoID	Nombre	Precio	Stock
*	NULL	NULL	NULL	NULL

```
-- Crear la tabla Ordenes
CREATE TABLE Ordenes (
    OrdenID INT AUTO_INCREMENT PRIMARY KEY,
    ClienteID INT NOT NULL,
    Fecha DATE NOT NULL,
    ProductoID INT,
    Cantidad INT NOT NULL,
    FOREIGN KEY (ProductoID) REFERENCES Productos(ProductoID)
);
```

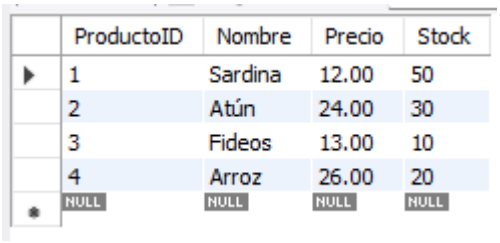
Table: ordenes

Columns:

OrdenID	int AI PK
ClienteID	int
Fecha	date
ProductoID	int
Cantidad	int

OrdenID	ClienteID	Fecha	ProductoID	Cantidad
NULL	NULL	NULL	NULL	NULL

```
-- Insertar registros en la tabla Productos
INSERT INTO Productos (Nombre, Precio, Stock) VALUES
('Sardina', 12.00, 50),
('Atún', 24.00, 30),
('Fideos', 13.00, 10),
('Arroz', 26.00, 20);
```



ProductoID	Nombre	Precio	Stock
1	Sardina	12.00	50
2	Atún	24.00	30
3	Fideos	13.00	10
4	Arroz	26.00	20
NULL	NULL	NULL	NULL

```
-- Insertar registros en la tabla Ordenes
INSERT INTO Ordenes (ClienteID, Fecha, ProductoID, Cantidad) VALUES
(1, '2024-06-01', 1, 2),
(1, '2024-06-02', 2, 1),
(2, '2024-06-03', 3, 5),
(3, '2024-06-04', 4, 3);
```

	OrdenID	ClienteID	Fecha	ProductoID	Cantidad
▶	1	1	2024-06-01	1	2
	2	1	2024-06-02	2	1
	3	2	2024-06-03	3	5
	4	3	2024-06-04	4	3
*	NULL	NULL	NULL	NULL	NULL

-- Seleccionar la función CalcularTotalOrden
Select CalcularTotalOrden(2) AS TotalOrdenConIVA;

	TotalOrdenConIVA
▶	27.60

EJERCICIO 2

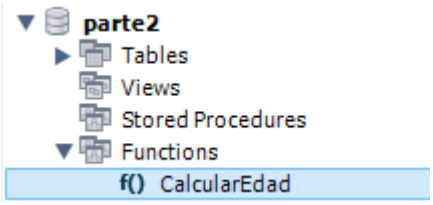
```
DELIMITER $$

CREATE FUNCTION CalcularEdad(fecha_nacimiento DATE)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE edad INT;
    SET edad = TIMESTAMPDIFF(YEAR, fecha_nacimiento, CURDATE());
    RETURN edad;
END $$

DELIMITER ;
```

-- EJERCICIO 2
-- Se crea la función
DELIMITER $$
CREATE FUNCTION CalcularEdad(fecha_nacimiento DATE) -- Función que toma un parámetro fecha tipo DATE.
RETURNS INT -- La función devuelve un valor de tipo INT.
DETERMINISTIC
BEGIN
DECLARE edad INT;
SET edad = TIMESTAMPDIFF(YEAR, fecha_nacimiento, CURDATE()); -- Se usa la función TIMESTAMPDIFF para la diferencia en años entre la fecha_nacimiento y fecha actual (CURDATE()).
RETURN edad; -- Devuelve la edad calculada.
END $$
DELIMITER ;

Function: CalcularEdad

Parameters:
fecha_nacimiento: DATE

Returns: int

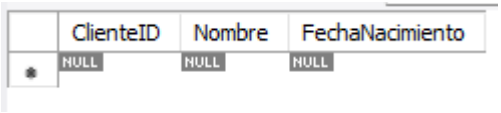



```
-- Crear la tabla Clientes
CREATE TABLE Clientes (
    ClienteID INT AUTO_INCREMENT PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    FechaNacimiento DATE NOT NULL
);
```

Table: clientes

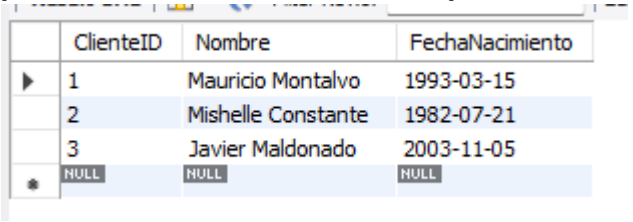
Columns:

ClienteID	int AI PK
Nombre	varchar(100)
FechaNacimiento	date



ClienteID	Nombre	FechaNacimiento
NULL	NULL	NULL

```
-- Insertar registro en la tabla Clientes
INSERT INTO Clientes (Nombre, FechaNacimiento) VALUES
('Mauricio Montalvo', '1993-03-15'),
('Mishelle Constante', '1982-07-21'),
('Javier Maldonado', '2003-11-05');
```



ClienteID	Nombre	FechaNacimiento
1	Mauricio Montalvo	1993-03-15
2	Mishelle Constante	1982-07-21
3	Javier Maldonado	2003-11-05
NULL	NULL	NULL

```
-- Seleccionar la función CalcularEdad
SELECT CalcularEdad(FechaNacimiento) AS EdadCliente
FROM Clientes
WHERE ClienteID = 3;
```

	EdadCliente
▶	21

EJERCICIO 3

```
DELIMITER $$

CREATE FUNCTION VerificarStock(producto_id INT)
RETURNS BOOLEAN
DETERMINISTIC
BEGIN
    DECLARE stock INT;
    SELECT Existencia INTO stock
    FROM Productos
    WHERE ProductoID = producto_id;

    IF stock > 0 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END $$

DELIMITER ;
```

-- EJERCICIO 3
-- Se crea la función
DELIMITER $$
CREATE FUNCTION VerificarStock(producto_id INT) -- Función que toma un parámetro producto_id de tipo INT.

RETURNS BOOLEAN -- La función devuelve un valor BOOLEAN indicando si hay o no hay stock del producto.
DETERMINISTIC
BEGIN
DECLARE stock_A INT;
SELECT stock INTO stock_A -- Realizar una consulta a la tabla Productos para obtener la cantidad de stock del producto.

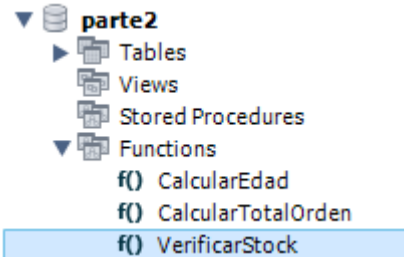
```
    FROM Productos
    WHERE ProductoID = producto_id;

    IF stock_A > 0 THEN  -- Si el stock es mayor que 0, la función devuelve TRUE = 1, stock disponible.
                RETURN TRUE;
        ELSE
                RETURN FALSE; -- Si no hay stock devuelve FALSE = 0.
        END IF;  -- Se cierra la estructura condicional.
END $$
DELIMITER ;
```

Function: VerificarStock

Parameters:

producto_id: INT

Returns: tinyint(1)




```
-- Crear la tabla Productos
CREATE TABLE Productos (
    ProductoID INT AUTO_INCREMENT PRIMARY KEY,
    Nombre VARCHAR(255) NOT NULL,
    Precio DECIMAL(10, 2) NOT NULL,
    Stock INT NOT NULL
);


-- Insertar registros en la tabla Productos
INSERT INTO Productos (Nombre, Precio, Stock) VALUES
('Sardina', 12.00, 50),
('Atún', 24.00, 30),
('Fideos', 13.00, 10),
('Arroz', 26.00, 20);
```

Table: productos

Columns:

ProductoID	int AI PK
Nombre	varchar(255)
Precio	decimal(10,2)
Stock	int

	ProductoID	Nombre	Precio	Stock
▶	1	Sardina	12.00	50
	2	Atún	24.00	30
	3	Fideos	13.00	10
	4	Arroz	26.00	20
*	NULL	NULL	NULL	NULL

-- Seleccionar la función VerificarStock
SELECT VerificarStock(3) AS StockDisponible;
-- True=1, False=0

	StockDisponible
▶	1

EJERCICIO 4

```
CREATE FUNCTION CalcularSaldo(id_cuenta INT)
RETURNS DECIMAL(10, 2)
DETERMINISTIC
BEGIN
    DECLARE saldo DECIMAL(10, 2);

    SELECT SUM(CASE
        WHEN tipo_transaccion = 'deposito' THEN monto
        WHEN tipo_transaccion = 'retiro' THEN -monto
        ELSE 0
    END) INTO saldo
    FROM Transacciones
    WHERE cuenta_id = id_cuenta;

    RETURN saldo;
END $$

DELIMITER ;
```

-- EJERCICIO 4
-- Se crea la función
DELIMITER $$
CREATE FUNCTION CalcularSaldo(id_cuenta int) -- La función toma un parámetro id_cuenta de tipo INT.

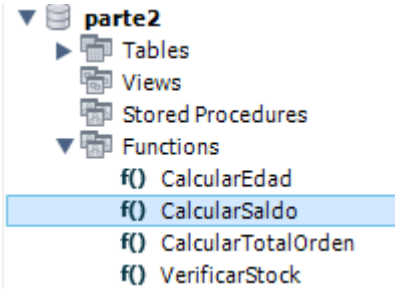
```
RETURNS DECIMAL(10, 2) -- La función devuelve un valor de tipo DECIMAL con 10 dígitos y 2 decimales.
DETERMINISTIC
BEGIN
        DECLARE saldo DECIMAL(10,2);
        -- Se utiliza una consulta SELECT para calcular el saldo.
   -- Si la transacción es un "depósito" se agrega el monto al saldo y si es un "retiro" se resta.

   SELECT SUM(CASE
                WHEN TipoTransaccion = 'deposito' THEN monto
      WHEN TipoTransaccion = 'retiro' THEN -monto
                ELSE 0
        END) INTO saldo
   FROM Transacciones  -- Se filtra las transacciones donde la cuenta_id coincide con el parámetro id_cuenta.
   WHERE CuentaID  = id_cuenta;

   RETURN saldo;  -- Devuelve el saldo de la transacción.
END $$
DELIMITER ;
```

Function: CalcularSaldo

Parameters:

id_cuenta: int

Returns: decimal(10,2)

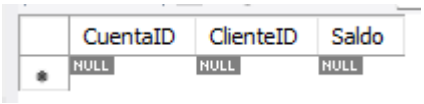



```
-- Crear la tabla Cuentas
CREATE TABLE Cuentas (
   CuentaID INT AUTO_INCREMENT PRIMARY KEY,
   ClienteID INT NOT NULL,
   Saldo DECIMAL(10, 2) NOT NULL
);
```

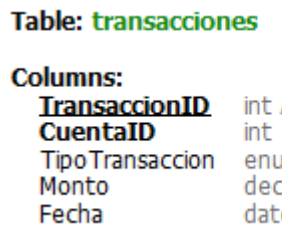
Table: cuentas

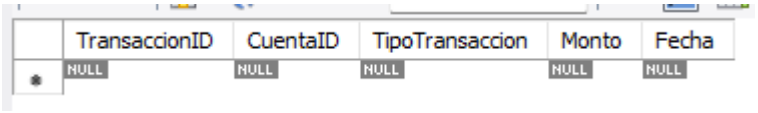
Columns:

CuentaID	int AI PK
ClienteID	int
Saldo	decimal(10,2)

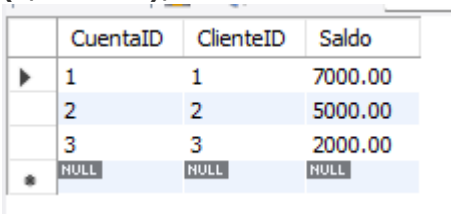



```
-- Crear la tabla Transacciones
CREATE TABLE Transacciones (
    TransaccionID INT AUTO_INCREMENT PRIMARY KEY,
    CuentaID INT NOT NULL,
    TipoTransaccion ENUM('deposito', 'retiro') NOT NULL,
    Monto DECIMAL(10, 2) NOT NULL,
    Fecha DATE NOT NULL,
    FOREIGN KEY (CuentaID) REFERENCES Cuentas(CuentaID)
);
```






```
-- Insertar registros en la tabla Cuentas
INSERT INTO Cuentas (ClienteID, Saldo) VALUES
(1, 7000.00),
(2, 5000.00),
(3, 2000.00);
```




```
-- Insertar registros en la tabla Transacciones
INSERT INTO Transacciones (CuentaID, TipoTransaccion, Monto, Fecha) VALUES
(1, 'deposito', 250.00, '2024-07-01'),
(1, 'retiro', 100.00, '2024-07-02'),
(2, 'deposito', 550.00, '2024-07-03'),
(3, 'retiro', 115.00, '2024-07-04');
```

	TransaccionID	CuentaID	TipoTransaccion	Monto	Fecha
▶	1	1	deposito	250.00	2024-07-01
	2	1	retiro	100.00	2024-07-02
	3	2	deposito	550.00	2024-07-03
	4	3	retiro	115.00	2024-07-04
*	NULL	NULL	NULL	NULL	NULL

```
-- Seleccionar la función CalcularSaldo
SELECT CalcularSaldo(2) AS MontoTransacción;
```

	MontoTransacción
▶	550.00

Link GitHub:

https://github.com/CristianTambaco/PARTE_1_y_2_Tarea_Funciones_de_Usuario.git