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.

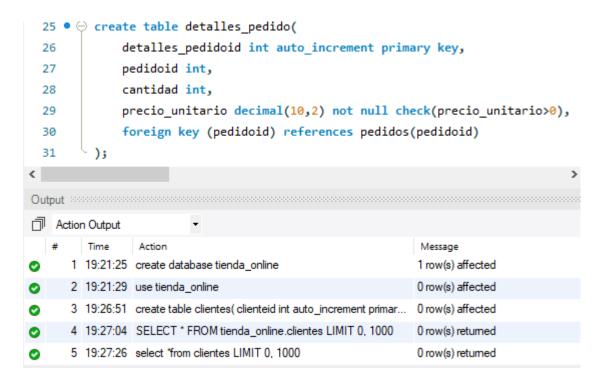
Pasos a Seguir:

- 1. Crear la Base de Datos y Tablas:
 - o Crea una base de datos llamada tienda online.
 - o 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).
 - Productos: Contendrá información sobre los productos (id, nombre, precio, stock, descripción).
 - Pedidos: Registra los pedidos realizados por los clientes (id, cliente_id, fecha del pedido, total).
 - Detalles_Pedido: Registra los detalles de cada pedido (id, pedido_id, producto_id, cantidad, precio unitario).

Creación de la base de datos y las tablas

```
1 •
       create database tienda_online;
       use tienda_online;
3 • ⊖ create table clientes(
           clienteid int auto_increment primary key,
4
           nombre varchar(100) not null,
5
           apellido varchar(100) not null,
6
           email varchar(100) unique,
7
           telefono varchar(100),
8
9
           fecha_registro date
10
       );
```

```
11 • ⊖ create table productos(
           productoid int auto_increment primary key,
12
13
           nombre varchar(100) not null,
           precio decimal(10,2) not null check(precio>0),
14
           stock int not null check(stock>=0),
15
           unique(nombre)
16
17
       );
18 • ⊖ create table pedidos(
           pedidoid int auto increment primary key,
19
           clienteid int,
20
           fecha pedido date,
21
           total int,
22
23
           foreign key (clienteid) references clientes(clienteid)
24
```



2. Restricciones:

- No se permiten valores nulos en campos como nombre, apellido, email, precio, y cantidad.
- Los precios deben ser positivos.
- o 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.

Ingreso de registros

```
insert into clientes (nombre,apellido,email,telefono,fecha_registro)
                             ("Daniel", "Guzman", "dg@gmail.com", "0984352134", "2020-01-15"),
34
       values
                             ("Carlos", "Endara", "ce@gmail.com", "0975434567", "2020-03-15"),
35
                             ("Gabriel", "Beltran", "gb@gmail.com", "0983657689", "2021-04-15");
36
37
       insert into productos (nombre,precio,stock)
38
39
       values
                             ("Ropa", 25.50,50),
                             ("Celulares", 150.75,80),
40
                             ("Laoptos",400.25,100);
41
42
       insert into pedidos (clienteid,fecha_pedido,total)
43 •
                             (2,"2024-12-15",10),
       values
44
45
                             (3,"2024-11-30",5),
                             (1,"2024-10-15",12);
46
47
48 •
       insert into detalles pedido(pedidoid,cantidad,precio unitario)
                             (1,5,25.50),
49
50
                             (2,10,150.75),
51
                             (3,6,400.25);
```

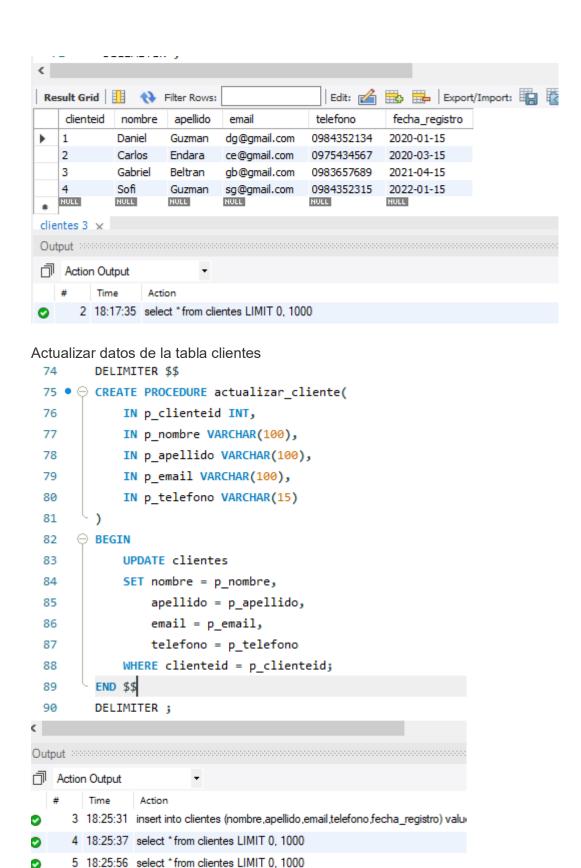
3. Crear Funciones de Usuario

Insertar un nuevo cliente

```
DELIMITER $$
 55
 56
 57 ● ○ CREATE PROCEDURE insertar cliente(
             IN p_nombre VARCHAR(100),
 58
 59
             IN p_apellido VARCHAR(100),
             IN p_email VARCHAR(255),
 60
 61
             IN p_telefono VARCHAR(15)
 62

→ BEGIN

 63
             INSERT INTO clientes (nombre, apellido, email, telefono)
 64
 65
             VALUES (p_nombre, p_apellido, p_email, p_telefono);
 66
         END $$
 67
         DELIMITER;
 68
 69
Output
Action Output
      1 18:15:15 insert into detalles_pedido(pedidoid,cantidad,precio_unitario) values (1,5,25.50), (2,10
     2 18:17:35 select *from clientes LIMIT 0, 1000
```



4. Función para obtener el nombre completo de un cliente:

6 18:28:12 CREATE PROCEDURE actualizar cliente(IN p. clienteid INT, IN

Esta función debe aceptar un cliente_id como parámetro y devolver
 el nombre completo (nombre + apellido) del cliente.

```
53
         DELIMITER $$
 54
 55 •
         CREATE FUNCTION obtener_nombre_completo(clienteid INT)
         RETURNS VARCHAR(255)
 56
         DETERMINISTIC
 57

→ BEGIN

 58
             DECLARE nombre_completo VARCHAR(255);
 59
             SELECT CONCAT(nombre, ' ', apellido) INTO nombre_completo
 60
             FROM clientes
 61
             WHERE id cliente = cliente id;
 62
             RETURN nombre_completo;
 63
 64
         END $$
         DELIMITER;
 65
 66
         -- Mostrar
 67 •
 68
         SELECT obtener_nombre_completo(3);
Output
Action Output
     6 18:48:25 insert into detalles_pedido(pedidoid,cantidad,precio_unitario) values (1,5,25.50), (2,10,15)
      7 18:48:26 insert into detalles_pedido(pedidoid,cantidad,precio_unitario) values (1,5,25.50), (2,10,15)
```

- o 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.

```
71
         DELIMITER $$
 72
          CREATE FUNCTION calcular_descuento(precio DECIMAL(10, 2), descuento DECIMAL(5, 2))
 73 •
 74
          RETURNS DECIMAL(10, 2)
 75
          DETERMINISTIC

→ BEGIN

 76
              DECLARE precio_con_descuento DECIMAL(10, 2);
 77
 78
              -- Calcular el precio con el descuento aplicado
 79
              SET precio_con_descuento = precio - (precio * descuento / 100);
 80
              -- Retornar el precio con descuento
 82
              RETURN precio_con_descuento;
 83
 84
          END $$
 85
Action Output
                                                                                          Message
         Time
                 Action
      8 18:48:26 insert into detalles_pedido(pedidoid,cantidad,precio_unitario) values (1,5,25.50), (2,10,150.75)... 3 row(s) affected
      9 18:48:26 insert into detalles_pedido(pedidoid,cantidad,precio_unitario) values (1,5,25.50), (2,10,150.75)... 3 row(s) affected
          SELECT calcular_descuento(100, 15);
 87 •
calcular_descuento(100,
   15)
  85.00
```

- 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.

```
90
         DELIMITER $$
         CREATE FUNCTION calcular_total_pedido(pedidoid INT)
         RETURNS DECIMAL(10, 2)
 92
         DETERMINISTIC
 93
 94

→ BEGIN

              DECLARE total DECIMAL(10, 2);
 95
              -- Calcular el total del pedido (precio * cantidad)
 96
              SELECT SUM(p.precio * dp.cantidad)
 97
              INTO total
 98
              FROM detalle pedido dp
 99
              JOIN productos p ON dp.productoid = p.productoid
100
101
              WHERE dp.pedido_id = pedido_id;
              -- Retornar el total calculado
102
              RETURN total;
103
        - END $$
104
105
106
         DELIMITER ;
Action Output
        Time
     15 19:07:04 SELECT calcular_descuento(100, 15) LIMIT 0, 1000

    16 19:07:34 SELECT calcular_descuento(100, 15) LIMIT 0, 1000
```

- o 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.

```
111
         DELIMITER $$
         CREATE FUNCTION verificar_stock(producto_id INT, cantidad INT)
112 •
113
         RETURNS BOOLEAN
114
         DETERMINISTIC
115

→ BEGIN

             DECLARE stock disponible INT;
116
             -- Obtener el stock disponible del producto
117
             SELECT stock INTO stock disponible
118
             FROM productos
119
             WHERE productoid = producto_id;
120
             -- Verificar si el stock disponible es suficiente
121
122
             IF stock_disponible >= cantidad THEN
                 RETURN TRUE;
123
124
             ELSE
125
                 RETURN FALSE;
             END IF;
126
127
        END $$
128
         DELIMITER ;
<
Action Output
        Time
               Action
     25 19:12:19 CREATE FUNCTION verificar_stock(producto_id INT, cantidad INT) RETURNS BOOLEAN .
 129 •
         -- Mostrar
 130
         SELECT verificar_stock(1, 10);
Ex
    verificar_stock(1,
    10)
1
```

- o 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.

```
132
        DELIMITER $$
133 •
        CREATE FUNCTION calcular_antiguedad(cliente_id INT)
134
         RETURNS INT
135
        DETERMINISTIC
136

→ BEGIN

             DECLARE fecha_registro DATE;
137
             DECLARE antiguedad INT;
138
139
140
             -- Obtener la fecha de registro del cliente
             SELECT fecha_registro INTO fecha_registro
141
             FROM clientes
142
            WHERE cliente_id = cliente_id;
143
             -- Calcular la antigüedad en años
144
             SET antiguedad = TIMESTAMPDIFF(YEAR, fecha_registro, CURDATE());
145
             -- Retornar la antigüedad calculada
146
             RETURN antiguedad;
147
148
         END $$
149
         DELIMITER ;
<
Output
Action Output
        Time
               Action
     28 19:14:55 SELECT verificar_stock(1, 10) LIMIT 0, 1000
```

5. Consultas de Uso de Funciones:

cliente id.

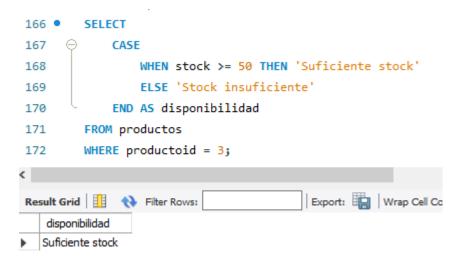
■ Consulta para obtener el nombre completo de un cliente dado su

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



■ Consulta para calcular el total de un pedido dado su pedido id.

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



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;
```

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 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 ;

```
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;
```