

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

```

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

```

```

25 • create table detalles_pedido(
26     detalles_pedidoid int auto_increment primary key,
27     pedidoid int,
28     cantidad int,
29     precio_unitario decimal(10,2) not null check(precio_unitario>0),
30     foreign key (pedidoid) references pedidos(pedidoid)
31 );

```

Output				
Action Output				
#	Time	Action	Message	
✓ 1	19:21:25	create database tienda_online	1 row(s) affected	
✓ 2	19:21:29	use tienda_online	0 row(s) affected	
✓ 3	19:26:51	create table clientes(clienteid int auto_increment primar...	0 row(s) affected	
✓ 4	19:27:04	SELECT * FROM tienda_online.clientes LIMIT 0, 1000	0 row(s) returned	
✓ 5	19:27:26	select *from clientes LIMIT 0, 1000	0 row(s) returned	

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.

Ingreso de registros

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

3. Crear Funciones de Usuario

Insertar un nuevo cliente

```
55 DELIMITER $$
56
57 • CREATE PROCEDURE insertar_cliente(
58     IN p_nombre VARCHAR(100),
59     IN p_apellido VARCHAR(100),
60     IN p_email VARCHAR(255),
61     IN p_telefono VARCHAR(15)
62 )
63 BEGIN
64     INSERT INTO clientes (nombre, apellido, email, telefono)
65     VALUES (p_nombre, p_apellido, p_email, p_telefono);
66 END $$
67
68 DELIMITER ;
69
```

Output

Action Output

#	Time	Action
✓ 1	18:15:15	insert into detalles_pedido(pedidoid,cantidad,precio_unitario) values (1,5,25.50). (2,10,150.75). (3,6,400.25);
✓ 2	18:17:35	select * from clientes LIMIT 0, 1000

Result Grid						
	clienteid	nombre	apellido	email	telefono	fecha_registro
▶	1	Daniel	Guzman	dg@gmail.com	0984352134	2020-01-15
	2	Carlos	Endara	ce@gmail.com	0975434567	2020-03-15
	3	Gabriel	Beltran	gb@gmail.com	0983657689	2021-04-15
	4	Sofi	Guzman	sg@gmail.com	0984352315	2022-01-15
*	NULL	NULL	NULL	NULL	NULL	NULL

clientes 3 x

Output

Action Output

#	Time	Action
2	18:17:35	select * from clientes LIMIT 0, 1000

Actualizar datos de la tabla clientes

```

74 DELIMITER $$
75 ● CREATE PROCEDURE actualizar_cliente(
76     IN p_clienteid INT,
77     IN p_nombre VARCHAR(100),
78     IN p_apellido VARCHAR(100),
79     IN p_email VARCHAR(100),
80     IN p_telefono VARCHAR(15)
81 )
82 BEGIN
83     UPDATE clientes
84     SET nombre = p_nombre,
85         apellido = p_apellido,
86         email = p_email,
87         telefono = p_telefono
88     WHERE clienteid = p_clienteid;
89 END $$
90 DELIMITER ;

```

Output

Action Output

#	Time	Action
3	18:25:31	insert into clientes (nombre,apellido,email,telefono,fecha_registro) valu
4	18:25:37	select * from clientes LIMIT 0, 1000
5	18:25:56	select * from clientes LIMIT 0, 1000
6	18:28:12	CREATE PROCEDURE actualizar cliente(IN p_clienteid INT, IN

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.

```
53 DELIMITER $$
54
55 • CREATE FUNCTION obtener_nombre_completo(clienteid INT)
56 RETURNS VARCHAR(255)
57 DETERMINISTIC
58 BEGIN
59     DECLARE nombre_completo VARCHAR(255);
60     SELECT CONCAT(nombre, ' ', apellido) INTO nombre_completo
61     FROM clientes
62     WHERE id_cliente = clienteid;
63     RETURN nombre_completo;
64 END $$
65 DELIMITER ;
66
67 • -- Mostrar
68 SELECT obtener_nombre_completo(3);
```

< Output

Action Output

	#	Time	Action
✓	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)

- 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 $$
72
73 • CREATE FUNCTION calcular_descuento(precio DECIMAL(10, 2), descuento DECIMAL(5, 2))
74 RETURNS DECIMAL(10, 2)
75 DETERMINISTIC
76 BEGIN
77     DECLARE precio_con_descuento DECIMAL(10, 2);
78
79     -- Calcular el precio con el descuento aplicado
80     SET precio_con_descuento = precio - (precio * descuento / 100);
81
82     -- Retornar el precio con descuento
83     RETURN precio_con_descuento;
84 END $$
85

```

Output				
Action Output				
#	Time	Action	Message	
✓ 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	

```

87 • SELECT calcular_descuento(100, 15);

```

Result Grid	
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 $$
91 • CREATE FUNCTION calcular_total_pedido(pedido_id INT)
92 RETURNS DECIMAL(10, 2)
93 DETERMINISTIC
94 BEGIN
95     DECLARE total DECIMAL(10, 2);
96     -- Calcular el total del pedido (precio * cantidad)
97     SELECT SUM(p.precio * dp.cantidad)
98     INTO total
99     FROM detalle_pedido dp
100    JOIN productos p ON dp.productoid = p.productoid
101    WHERE dp.pedido_id = pedido_id;
102     -- Retornar el total calculado
103     RETURN total;
104 END $$
105
106 DELIMITER ;
```

Output

Action Output

#	Time	Action
✓ 15	19:07:04	SELECT calcular_descuento(100, 15) LIMIT 0, 1000
✓ 16	19:07:34	SELECT calcular_descuento(100, 15) LIMIT 0, 1000

- 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 $$
112 • CREATE FUNCTION verificar_stock(producto_id INT, cantidad INT)
113 RETURNS BOOLEAN
114 DETERMINISTIC
115 BEGIN
116     DECLARE stock_disponible INT;
117     -- Obtener el stock disponible del producto
118     SELECT stock INTO stock_disponible
119     FROM productos
120     WHERE productoid = producto_id;
121     -- Verificar si el stock disponible es suficiente
122     IF stock_disponible >= cantidad THEN
123         RETURN TRUE;
124     ELSE
125         RETURN FALSE;
126     END IF;
127 END $$
128 DELIMITER ;

```

Output

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);

```

Result Grid		Filter Rows:	Ex
	verificar_stock(1, 10)		
▶	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.


```

132 DELIMITER $$
133 • CREATE FUNCTION calcular_antigüedad(cliente_id INT)
134 RETURNS INT
135 DETERMINISTIC
136 BEGIN
137     DECLARE fecha_registro DATE;
138     DECLARE antigüedad INT;
139
140     -- Obtener la fecha de registro del cliente
141     SELECT fecha_registro INTO fecha_registro
142     FROM clientes
143     WHERE cliente_id = cliente_id;
144     -- Calcular la antigüedad en años
145     SET antigüedad = TIMESTAMPDIF(YEAR, fecha_registro, CURDATE());
146     -- Retornar la antigüedad calculada
147     RETURN antigüedad;
148 END $$
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:

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

```

153 • SELECT CONCAT(nombre, ' ', apellido) AS nombre_completo
154 FROM clientes
155 WHERE clienteid = 4;

```

nombre_completo
Sofi Guzman

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

```

157 • SELECT precio, (precio - (precio * 0.10)) AS precio_con_descuento
158 FROM productos

```

<		
Result Grid	Filter Rows:	Export: Wrap Cell Content: IA
	precio	precio_con_descuento
▶	150.75	135.6750

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

```

161 • SELECT SUM(p.precio * dp.cantidad) AS total_pedido
162 FROM detalles_pedido dp
163 JOIN productos p ON dp.pedidoid = p.productoid
164 WHERE dp.pedidoid = 3;

```

<		
Result Grid	Filter Rows:	Export: Wrap Cell Cont
	total_pedido	
▶	31219.50	

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

```

166 • SELECT
167 CASE
168 WHEN stock >= 50 THEN 'Suficiente stock'
169 ELSE 'Stock insuficiente'
170 END AS disponibilidad
171 FROM productos
172 WHERE productoid = 3;

```

<		
Result Grid	Filter Rows:	Export: Wrap Cell Co
	disponibilidad	
▶	Suficiente stock	

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 ;
```

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 ;
```