

## 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).

##### 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.

### 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.

```
59 -- función para obtener el nombre de un cliente
60 delimiter //
61 • create function obtener_nombre(cliente_id int) returns varchar(101)
62 deterministic
63 begin
64     return (select nombre from clientes where id = cliente_id limit 1); -- Devuelve el nombre del cliente.
65 end //
66 delimiter ;
```

#### ○ 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.

```
69 -- función para calcular el descuento de un producto
70 delimiter //
71 • create function descuento(precio decimal(10, 2), descuento decimal(5, 2)) returns decimal(10, 2)
72 deterministic
73 begin
74     return precio - (precio * (descuento / 100)); -- Calcula y devuelve el precio con descuento.
75 end //
76 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.

```
78 -- Función para calcular el total de un pedido
79 delimiter //
80 • create function total_pedido(pedido_id int) returns decimal(10, 2)
81 deterministic
82 begin
83     declare total_pedido decimal(10, 2);
84     select precio_unitario * cantidad into total_pedido
85     from detalles_pedido
86     where detalles_pedido.pedido_id = pedido_id; -- Calcula el total del pedido.
87     return total_pedido;
88 end //
89 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`.

```
91 -- función para verificar la disponibilidad de stock de un producto
92 delimiter //
93 • create function verificar_stock(producto_id int, cantidad int) returns varchar(10)
94     deterministic
95 begin
96     declare stock_disponible int;
97     select stock into stock_disponible
98     from productos
99     where productos.id = producto_id; -- Obtiene el stock disponible del producto.
100     if stock_disponible >= cantidad then
101         return 'true'; -- Retorna 'true' si hay suficiente stock
102     else
103         return 'false'; -- Retorna 'false' si no hay suficiente stock.
104     end if;
105 end //
106 delimiter ;
```

- 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.



```
108 -- función para calcular la antigüedad de un cliente
109 delimiter //
110 • create function calcular_antigüedad(cliente_id int) returns int
111     deterministic
112 begin
113     declare antigüedad int;
114     select timestampdiff(year, fecha_registro, curdate()) into antigüedad
115     from clientes
116     where id = cliente_id; -- Calcula la diferencia en años desde la fecha de registro.
117     return antigüedad;
118 end //
119 delimiter ;
```

## 5. Consultas de Uso de Funciones:

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

```
123      -- Obtener el nombre del cliente con ID 1.
124      select obtener_nombre (1) as nombre_cliente;
125
126      -- Obtener el total del pedido con ID 3.
```

---



Result Grid |  Filter Rows:  | Export:  Wrap

	nombre_cliente
▶	Juan

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

```
126      -- Obtener el total del pedido con ID 3.
127      select descuento (25.00, 10) as precio_descuento;
128
```

---



Result Grid |  Filter Rows:  | Export:  Wrap Cell C

	precio_descuento
▶	22.50

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

```
129      -- Obtener el total del pedido con ID 3.
130      select total_pedido (3) as pedido_total;
131
```

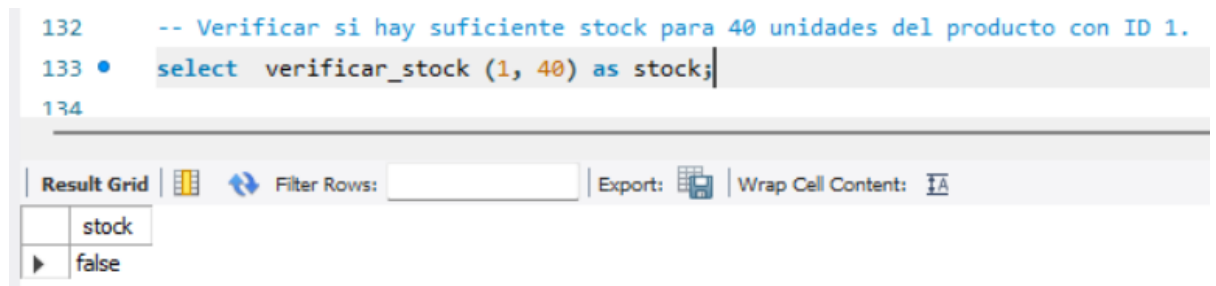
---

Result Grid |  Filter Rows:  | Export:  V

	pedido_total
▶	45.00

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

```
132 -- Verificar si hay suficiente stock para 40 unidades del producto con ID 1.
133 • select verificar_stock (1, 40) as stock;
134
```



stock
false

## 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 ;
```