

Nombre: Richard Padilla

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`.

-- Richard Padilla - PARTE 1

-- Creacion de la base de datos

`create database Tienda_online;`

`use Tienda_online;`

```
1  -- Richard Padilla - PARTE 1
2  -- Creacion de la base de datos
3  • create database Tienda_online;
4  • 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).
 - 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.

-- Creacion de las tablas

```
create table clientes (  
    id int auto_increment primary key,  
    nombre varchar(50) not null,  
    apellido varchar(50) not null,  
    email varchar(100) not null unique,  
    telefono varchar(15),  
    fecha_registro date not null  
);  
  
create table productos (  
    id int auto_increment primary key,  
    nombre varchar(100) not null unique,  
    precio decimal(10, 2) not null check (precio > 0),  
    stock int not null check (stock >= 0),  
    descripcion text  
);
```

```
create table pedidos (  
    id int auto_increment primary key,  
    cliente_id int not null,  
    fecha_pedido date not null,  
    total decimal(10, 2) not null check (total >= 0),  
    foreign key (cliente_id) references clientes(id)  
);
```

```
create table detalles_pedido (  
    id int auto_increment primary key,  
    pedido_id int not null,  
    producto_id int not null,  
    cantidad int not null check (cantidad > 0),  
    precio_unitario decimal(10, 2) not null check (precio_unitario > 0),  
    foreign key (pedido_id) references pedidos(id),  
    foreign key (producto_id) references productos(id)  
);
```

```
6 -- Creacion de las tablas
7 create table clientes (
8     id int auto_increment primary key,
9     nombre varchar(50) not null,
10    apellido varchar(50) not null,
11    email varchar(100) not null unique,
12    telefono varchar(15),
13    fecha_registro date not null
14 );
15
16 create table productos (
17     id int auto_increment primary key,
18     nombre varchar(100) not null unique,
19     precio decimal(10, 2) not null check (precio > 0),
20     stock int not null check (stock >= 0),
21     descripcion text
22 );
23
24 create table pedidos (
25     id int auto_increment primary key,
26     cliente_id int not null,
27     fecha_pedido date not null,
28     total decimal(10, 2) not null check (total >= 0),
29     foreign key (cliente_id) references clientes(id)
30 );
31
32 create table detalles_pedido (
33     id int auto_increment primary key,
34     pedido_id int not null,
35     producto_id int not null,
36     cantidad int not null check (cantidad > 0),
37     precio_unitario decimal(10, 2) not null check (precio_unitario > 0),
38     foreign key (pedido_id) references pedidos(id),
39     foreign key (producto_id) references productos(id)
40 );
41
```

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 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 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 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 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.

5. Consultas de Uso de Funciones:

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

```
-- Creacion de funciones
-- Obtener el nombre completo del cliente
delimiter //

create function obtener_nombres(cliente_id int)
returns varchar(50)
deterministic
begin
  declare nombre_completo varchar(50);
  select concat(nombre, ' ', apellido)
  into nombre_completo
  from clientes
  where id = cliente_id;
  return nombre_completo;
end;
//
```

```

delimiter ;

select obtener_nombres(1) as nombre_completo;

-- Calcular el precio con descuento
delimiter //

create function Calcular_precio_con_descuento(precio decimal(10,2), descuento
decimal(5,2))
returns decimal(10,2)
deterministic
begin
    return precio - (precio * (descuento/100));
end;

//

delimiter ;

select Calcular_precio_con_descuento(12.50,10) as Precio_Descuento;

-- Calcular el total de un pedido
delimiter //

create function Calcular_total_pedidos(pedido_id int)
returns decimal(10,2)
deterministic
begin
    declare TOTAL decimal(10,2);
    select sum(cantidad*precio_unitario)
    into TOTAL
    from detalles_pedido
    where pedido_id = pedido_id;
    return ifnull(TOTAL, 0);
end;

//

delimiter ;

select Calcular_total_pedidos(1) as Total_Pedidos;

-- Verificar la disponibilidad de stock
delimiter //

create function Disponibilidad_stock(producto_id int, cantidad int)
returns boolean
deterministic
begin
    declare stock_actual int;
    select stock

```

```

        into stock_actual
        from productos
        where id = producto_id;

        return stock_actual >= cantidad;
    end;
//

delimiter ;

select Disponibilidad_stock(2,7) as Stock_Disponible;

-- Calcular la antigüedad de un cliente
delimiter //

create function Antigüedad_cliente(cliente_id int)
returns int
deterministic
begin
    declare antiguo int;
    select timestampdiff(year, fecha_registro, curdate())
    into antiguo
    from clientes
    where id = cliente_id;
    return antiguo;
end;
//

delimiter ;

select Antigüedad_cliente(1) as Cliente_mas_antiguo;

```

```

77  -- Obtener el nombre completo del cliente
78  delimiter //
79
80 • create function obtener_nombres(cliente_id int)
81  returns varchar(50)
82  deterministic
83  begin
84      declare nombre_completo varchar(50);
85      select concat(nombre, ' ', apellido)
86      into nombre_completo
87      from clientes
88      where id = cliente_id;
89      return nombre_completo;
90  end;
91  //
92
93  delimiter ;
94

```

94

```
95 • select obtener_nombres(1) as nombre_completo;
```

96

Result Grid	Filter Rows:	Export:	Wrap Cell Content
nombre_completo			
Richard Padilla			

99

```
100 • create function Calcular_precio_con_descuento(precio decimal(10,2), descuento decimal(5,2))
```

```
101 returns decimal(10,2)
```

```
102 deterministic
```

```
103 begin
```

```
104     return precio - (precio * (descuento/100));
```

```
105 end;
```

106

```
107 //
```

108

```
109 delimiter ;
```

110

```
111 • select Calcular_precio_con_descuento(12.50,10) as Precio_Descuento;
```

112

Result Grid	Filter Rows:	Export:	Wrap Cell Content
Precio_Descuento			
11.25			

```
113 -- Calcular el total de un pedido
```

```
114 delimiter //
```

115

```
116 • create function Calcular_total_pedidos(pedido_id int)
```

```
117 returns decimal(10,2)
```

```
118 deterministic
```

```
119 begin
```

```
120     declare TOTAL decimal(10,2);
```

```
121     select sum(cantidad*precio_unitario)
```

```
122     into TOTAL
```

```
123     from detalles_pedido
```

```
124     where pedido_id = pedido_id;
```

```
125     return ifnull(TOTAL, 0);
```

```
126 end;
```

```
127 //
```

128

```
129 delimiter ;
```

130

```
131 • select Calcular_total_pedidos(1) as Total_Pedidos;
```

132

Result Grid	Filter Rows:
Total_Pedidos	
112.50	


```

133 -- Verificar la disponibilidad de stock
134 delimiter //
135
136 • create function Disponibilidad_stock(producto_id int, cantidad int)
137 returns boolean
138 deterministic
139 begin
140     declare stock_actual int;
141     select stock
142     into stock_actual
143     from productos
144     where id = producto_id;
145
146     return stock_actual >= cantidad;
147 end;
148 //
149
150 delimiter ;
151
152 • select Disponibilidad_stock(2,7) as Stock_Disponible;

```

Result Grid		Filter Rows:	Export:	Wrap Cell C
	Stock_Disponible			
▶	1			

```

154 -- Calcular la antigüedad de un cliente
155 delimiter //
156
157 • create function Antigüedad_cliente(cliente_id int)
158 returns int
159 deterministic
160 begin
161     declare antiguo int;
162     select timestampdiff(year, fecha_registro, curdate())
163     into antiguo
164     from clientes
165     where id = cliente_id;
166     return antiguo;
167 end;
168 //
169
170 delimiter ;
171
172 • select Antigüedad_cliente(1) as Cliente_mas_antiguo;
172 • select Antigüedad_cliente(1) as Cliente_mas_
173

```

Result Grid		Filter Rows:	Export:	Wrap Cell C
	Cliente_mas_antiguo			
▶	6			

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

```

1  -- Richard Padilla - PARTE 2
2  -- Aprendizaje de Funciones SQL: Creación, Análisis y Ejecución
3  • create database Funciones_sql;
4  • use Funciones_sql;
5
6  • create table Productos (
7      ProductoID int auto_increment primary key,
8      Nombre varchar(100) not null unique,
9      Precio decimal(10, 2) not null check (Precio > 0),
10     Stock int not null check (Stock >= 0),
11     Descripcion text
12 );
13
14 • create table Ordenes (
15     OrdenID int auto_increment primary key,
16     ProductoID int not null,
17     Cantidad int not null check (Cantidad > 0),
18     Fecha_Orden date not null,
19     foreign key (ProductoID) references Productos(ProductoID)
20 );

```

```

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 ;

```

63

64 • `select` `Calcular_Total_Orden` (1) `as` `TOTAL`;

65

Result Grid



Filter Rows:

Export:



Wrap Cell Content:

	TOTAL
▶	1840.00

EJERCICIO 2

```
66 -- FUNCION 2
67 • create table Personas (
68     PersonaID int auto_increment primary key,
69     Nombre varchar(50) not null,
70     Apellido varchar(50) not null,
71     Fecha_Nacimiento date not null
72 );
73
74 • insert into Personas (Nombre, Apellido, Fecha_Nacimiento)
75 values
76 ('Richard', 'Padilla', '1990-05-15'),
77 ('Ana', 'López', '1985-12-20'),
78 ('Carlos', 'Pérez', '2000-07-10'),
79 ('María', 'García', '1995-03-25');
```

DELIMITER \$\$

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

DELIMITER ;

```
96 • select Nombre, Apellido, calcularEdad(Fecha_Nacimiento) as Edad from Personas;
97
98
```

Result Grid



Filter Rows:

Export:



Wrap Cell Content:

	Nombre	Apellido	Edad
▶	Richard	Padilla	34
	Ana	López	39
	Carlos	Pérez	24
	María	García	29

EJERCICIO 3

```
99  -- FUNCION 3
100 • create table Productos1 (
101     ProductosID int auto_increment primary key,
102     Nombre varchar(100) not null,
103     Precio decimal(10, 2) not null check (Precio > 0),
104     Existencia int not null check (Existencia >= 0)
105 );
106
107 -- Insertar registros de ejemplo
108 • insert into Productos1 (Nombre, Precio, Existencia)
109 values
110 ('Café Colombiano', 10.50, 100),
111 ('Café Expreso', 12.00, 50),
112 ('Taza de Cerámica', 5.00, 0),
113 ('Filtro de Café', 2.50, 300);
```

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 ;

```
136 • select VerificarStock(1) as Stock_Disponible;
137
138
```

Stock_Disponible
1

EJERCICIO 4

```
130
139 -- FUNCION 4
140 • create table Transacciones (
141     TransaccionID int auto_increment primary key,
142     cuenta_id int not null,
143     tipo_transaccion varchar(10) not null check (tipo_transaccion in ('deposito', 'retiro')),
144     monto decimal(10, 2) not null check (monto > 0),
145     fecha_transaccion date not null
146 );
147
148 • insert into Transacciones (cuenta_id, tipo_transaccion, monto, fecha_transaccion)
149 values
150 (1, 'deposito', 1000.00, '2024-12-20'),
151 (1, 'retiro', 200.00, '2024-12-21'),
152 (1, 'deposito', 500.00, '2024-12-22'),
153 (2, 'deposito', 1500.00, '2024-12-20'),
154 (2, 'retiro', 700.00, '2024-12-21'),
155 (3, 'deposito', 300.00, '2024-12-22');
156
```

```
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 ;

178

179 • `select` `CalcularSaldo(1)` `as` `SALDO_TOTAL`;

180

181

182

Result Grid



Filter Rows:

Export:



Wrap Cell

	SALDO_TOTAL
▶	1300.00