Tarea Funciones de Usuario

Nombre: Christian Márquez

PARTE 1 Tarea Funciones de Usuario

Tarea: Funciones de Usuario en Bases de Datos

1. Crear la Base de Datos y Tablas

Query | Query History

```sql
-- Crear la base de datos
CREATE DATABASE tienda_online;

-- Crear la tabla Clientes
CREATE TABLE Clientes (
  id SERIAL 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
);

-- Crear la tabla Productos
CREATE TABLE Productos (
  id SERIAL 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
);

-- Crear la tabla Pedidos
CREATE TABLE Pedidos (
  id SERIAL PRIMARY KEY,
  cliente_id INT NOT NULL REFERENCES Clientes(id),
  fecha_pedido DATE NOT NULL,
  total DECIMAL(10, 2)
);
-- Crear la tabla Detalles_Pedido
CREATE TABLE Detalles_Pedido (
  id SERIAL PRIMARY KEY,
  pedido_id INT NOT NULL REFERENCES Pedidos(id),
  producto_id INT NOT NULL REFERENCES Productos(id),
  cantidad INT NOT NULL CHECK (cantidad > 0),
  precio_unitario DECIMAL(10, 2) NOT NULL
);

```

2. Crear Funciones de Usuario

Función para obtener el nombre completo de un cliente:

```sql
CREATE OR REPLACE FUNCTION obtener_nombre_cliente(cliente_id INT)
RETURNS VARCHAR AS $$
DECLARE
  nombre_completo VARCHAR;
BEGIN
  SELECT nombre || ' ' || apellido INTO nombre_completo
  FROM Clientes
  WHERE id = cliente_id;
  RETURN nombre_completo;
END;
$$ LANGUAGE plpgsql;

```

Data Output | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 75 msec.
```

Función para calcular el descuento de un producto:

```
CREATE OR REPLACE FUNCTION calcular_descuento(precio DECIMAL, descuento DECIMAL)
RETURNS DECIMAL AS $$
DECLARE
  precio_con_descuento DECIMAL;
BEGIN
  precio_con_descuento := precio - (precio * descuento / 100);
  RETURN precio_con_descuento;
END;
$$ LANGUAGE plpgsql;
```

Data Output | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 78 msec.
```

Función para calcular el total de un pedido:

```
CREATE OR REPLACE FUNCTION calcular_total_pedido(pedido_id INT)
RETURNS DECIMAL AS $$
DECLARE
  total DECIMAL := 0;
BEGIN
  SELECT SUM(d.cantidad * d.precio_unitario)
  INTO total
  FROM Detalles_Pedido d
  WHERE d.pedido_id = pedido_id;
  RETURN total;
END;
$$ LANGUAGE plpgsql;
```

Data Output | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 83 msec.
```

Función para verificar la disponibilidad de stock de un producto:

```
CREATE OR REPLACE FUNCTION verificar_stock(producto_id INT, cantidad INT)
RETURNS BOOLEAN AS $$
DECLARE
  stock_disponible INT;
BEGIN
  SELECT stock INTO stock_disponible
  FROM Productos
  WHERE id = producto_id;

  IF stock_disponible >= cantidad THEN
    RETURN TRUE;
  ELSE
    RETURN FALSE;
  END IF;
END;
$$ LANGUAGE plpgsql;
```

Data Output | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 76 msec.
```

Función para calcular la antigüedad de un cliente:

```sql
CREATE OR REPLACE FUNCTION calcular_antiguedad(cliente_id INT)
RETURNS INT AS $$
DECLARE
  antiguedad INT;
  fecha_registro DATE;
BEGIN
  SELECT fecha_registro INTO fecha_registro
  FROM Clientes
  WHERE id = cliente_id;

  SELECT EXTRACT(YEAR FROM age(fecha_registro)) INTO antiguedad;

  RETURN antiguedad;
END;
$$ LANGUAGE plpgsql;
```

Data Output | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 105 msec.
```

5. Consultas de Uso de Funciones:

```sql
--Consulta para obtener el nombre completo de un cliente dado su cliente_id:
SELECT obtener_nombre_cliente(1) AS nombre_completo;

--Consulta para calcular el descuento de un producto dado su precio y un descuento del 10%:
SELECT calcular_descuento(100.00, 10) AS precio_con_descuento;

--Consulta para calcular el total de un pedido dado su pedido_id:
SELECT calcular_total_pedido(1) AS total_pedido;

--Consulta para verificar si un producto tiene suficiente stock para una cantidad solicitada:
SELECT verificar_stock(1, 5) AS stock_suficiente;
```

PARTE 2

Instrucciones:

1. Transcripción y análisis del código SQL.

#Ejercicio 1: CalcularTotalOrden

```
DELIMITER $$

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: CalcularEdad

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: VerificarStock
DELIMITER $$
CREATE FUNCTION VerificarStock(producto_id INT)
RETURNS BOOLEAN
DETERMINISTIC
```

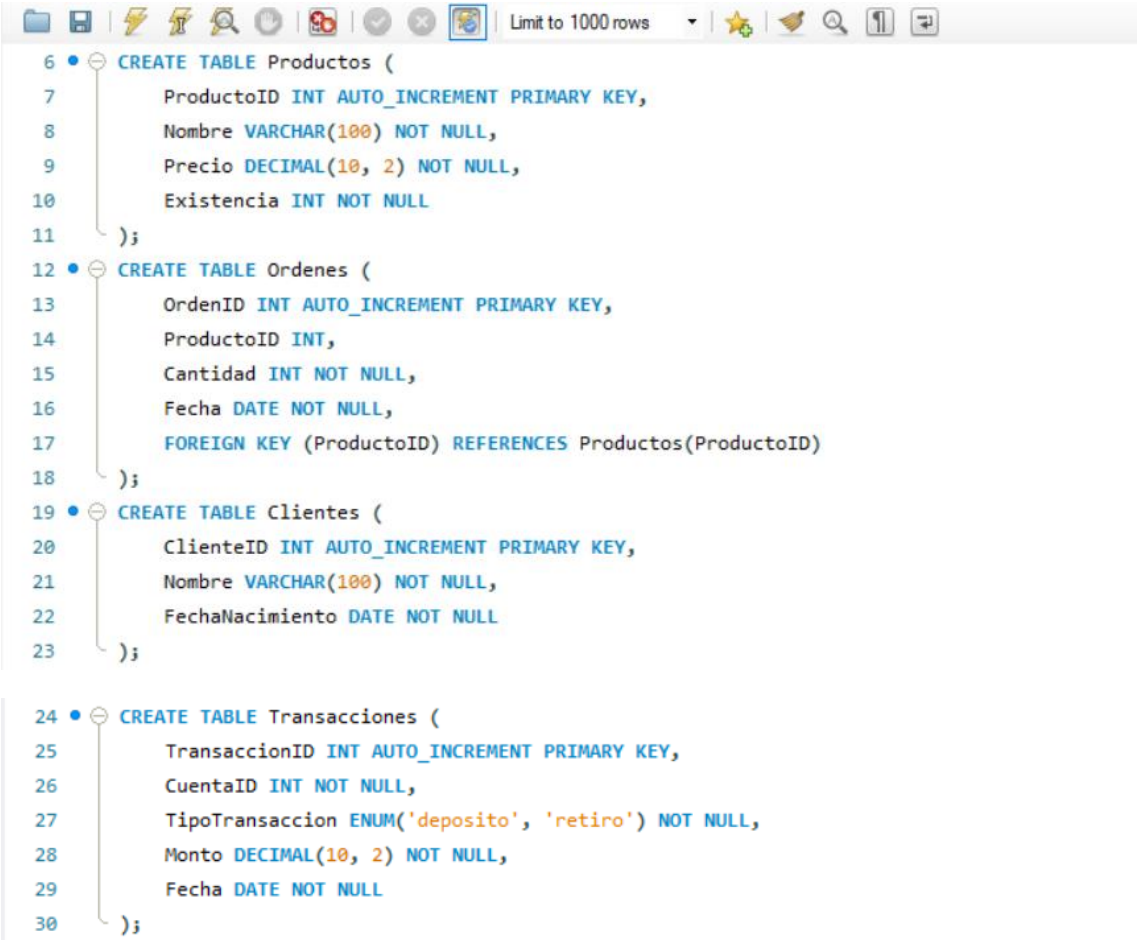
```
BEGIN
  DECLARE stock INT;
  SELECT Existencia INTO stock
  FROM Productos
  WHERE ProductoID = producto_id;
  IF stock > 0 THEN
    RETURN TRUE;
  ELSE
    RETURN FALSE;
  END IF;
END $$
DELIMITER ;

#Ejercicio 4: CalcularSaldo
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 ;

2. Creación de las tablas necesarias para almacenar los datos.



```sql
CREATE TABLE Productos (
    ProductoID INT AUTO_INCREMENT PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Precio DECIMAL(10, 2) NOT NULL,
    Existencia INT NOT NULL
);
CREATE TABLE Ordenes (
    OrdenID INT AUTO_INCREMENT PRIMARY KEY,
    ProductoID INT,
    Cantidad INT NOT NULL,
    Fecha DATE NOT NULL,
    FOREIGN KEY (ProductoID) REFERENCES Productos(ProductoID)
);
CREATE TABLE Clientes (
    ClienteID INT AUTO_INCREMENT PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    FechaNacimiento DATE NOT NULL
);
```

```sql
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
);
```

3. Ejecución de las funciones SQL creadas y captura de los resultados.

```sql
# Ejercicio 1: CalcularTotalOrden
DELIMITER $$
CREATE FUNCTION CalcularTotalOrden2(id_orden INT)
RETURNS DECIMAL(10, 2)
DETERMINISTIC
BEGIN
    DECLARE total DECIMAL(10, 2);
    DECLARE iva DECIMAL(10, 2);
    SET iva = 0.15;
    SELECT SUM(P.precio * O.cantidad) INTO total
    FROM Ordenes O
    JOIN Productos P ON O.ProductoID = P.ProductoID
    WHERE O.OrdenID = id_orden;
    IF total IS NULL THEN
        SET total = 0;
    END IF;
    SET total = total + (total * iva);
    RETURN total;
END $$
DELIMITER ;
```

```sql
# Ejercicio 1
SELECT CalcularTotalOrden2(1);
```

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

CalcularTotalOrden2(1)
230.00

```sql
# Ejercicio 2: CalcularEdad

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 ;
```

```sql
# Ejercicio 2
SELECT CalcularEdad('1990-05-15');
SELECT CalcularEdad('1985-10-25');

# Ejercicio 3
```

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

CalcularEdad('1990-05-15')
34

```sql
# Ejercicio 2
SELECT CalcularEdad('1990-05-15');
SELECT CalcularEdad('1985-10-25');

# Ejercicio 3
```

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

CalcularEdad('1985-10-25')
39

```sql
# Ejercicio 3: VerificarStock

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 ;
```

```
153    # Ejercicio 3
154    SELECT VerificarStock(1);
155    SELECT VerificarStock(2);
156
157    # Ejercicio 4
```

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

	VerificarStock(1)
▶	1

```
117    # Ejercicio 4: CalcularSaldo
118    DELIMITER $$
119    CREATE FUNCTION CalcularSaldo(id_cuenta INT)
120    RETURNS DECIMAL(10, 2)
121    DETERMINISTIC
122    BEGIN
123        DECLARE saldo DECIMAL(10, 2);
124        SELECT SUM(CASE
125            WHEN tipo_transaccion = 'deposito' THEN monto
126            WHEN tipo_transaccion = 'retiro' THEN -monto
127            ELSE 0
128        END) INTO saldo
129        FROM Transacciones
130        WHERE cuenta_id = id_cuenta;
131        IF saldo IS NULL THEN
132            SET saldo = 0;
133        END IF;
134        RETURN saldo;
135    END $$
136    DELIMITER ;
```

4. Explicación detallada de cada línea del código.

```
62     # Ejercicio 1: CalcularTotalOrden
63     DELIMITER $$
64     CREATE FUNCTION CalcularTotalOrden2(id_orden INT)
65     RETURNS DECIMAL(10, 2)
66     DETERMINISTIC
67     BEGIN
68         DECLARE total DECIMAL(10, 2);
69         DECLARE iva DECIMAL(10, 2);
70         SET iva = 0.15;
71         SELECT SUM(P.precio * O.cantidad) INTO total
72         FROM Ordenes O
73         JOIN Productos P ON O.ProductoID = P.ProductoID
74         WHERE O.OrdenID = id_orden;
75         IF total IS NULL THEN
76             SET total = 0;
77         END IF;
78         SET total = total + (total * iva);
79         RETURN total;
80     END $$
81     DELIMITER ;
```

- CREATE FUNCTION: Crea una función llamada CalcularTotalOrden, que toma como parámetro un id_orden de tipo INT.
- RETURNS DECIMAL(10,2): La función devuelve un valor de tipo decimal con 2 dígitos después del punto decimal.
- Declaración de variables: Se definen dos variables, total y iva, ambas de tipo DECIMAL(10,2).
- Cálculo del IVA: Se asigna el valor 0.15 a la variable iva, lo que representa un 15% de impuesto.
- Cálculo del total: Se calcula el total de la orden sumando el precio de los productos multiplicado por su cantidad (a través de un JOIN entre las tablas Ordenes y Productos).
- Aplicación del IVA: El total se incrementa en el 15% (total + (total * iva)).

- RETURN: Finalmente, se retorna el total con el IVA incluido.

```
# Ejercicio 2: CalcularEdad

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 ;
```

- CREATE FUNCTION: Define una función llamada CalcularEdad que toma como parámetro una fecha de nacimiento (fecha_nacimiento de tipo DATE).
- RETURNS INT: La función devuelve un valor de tipo entero, que será la edad calculada.
- TIMESTAMPDIFF(YEAR, fecha_nacimiento, CURDATE()): Utiliza la función TIMESTAMPDIFF para calcular la diferencia en años entre la fecha de nacimiento proporcionada y la fecha actual (CURDATE()).
- RETURN edad: Retorna el valor de la edad calculada.

```
# Ejercicio 3: VerificarStock

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: Define una función llamada VerificarStock que toma como parámetro un producto_id de tipo INT.
- RETURNS BOOLEAN: La función devuelve un valor de tipo BOOLEAN (TRUE o FALSE).
- SELECT Existencia INTO stock: Se realiza una consulta para obtener la cantidad de stock disponible de un producto en la tabla Productos, usando el producto_id como clave de búsqueda.
- IF: Si el stock es mayor que 0, retorna TRUE, lo que indica que hay suficiente stock disponible; de lo contrario, retorna FALSE.

```sql
# Ejercicio 4: CalcularSaldo
DELIMITER $$
CREATE FUNCTION CalcularSaldo(id_cuenta INT)
RETURNS DECIMAL(10, 2)
DETERMINISTIC
BEGIN
    DECLARE saldo DECIMAL(10, 2);
    SELECT SUM(CASE
        WHEN tipo_transaccion = 'deposito' THEN monto
        WHEN tipo_transaccion = 'retiro' THEN -monto
        ELSE 0
    END) INTO saldo
    FROM Transacciones
    WHERE cuenta_id = id_cuenta;
    IF saldo IS NULL THEN
        SET saldo = 0;
    END IF;
    RETURN saldo;
END $$
DELIMITER ;
```

- **`CREATE FUNCTION`**: Define una función llamada `CalcularSaldo` que toma como parámetro un `id_cuenta` de tipo `INT`.
- **`RETURNS DECIMAL(10, 2)`**: La función devuelve un valor decimal con 2 dígitos después del punto decimal (el saldo de la cuenta).
- **`CASE`**: Dentro de la consulta `SELECT`, se utiliza un `CASE` para evaluar el tipo de transacción:

  - Si es un `deposito`, se suma el monto.
  - Si es un `retiro`, se resta el monto.
  - Si no es ninguno de estos, se considera `0`.

- **`SELECT INTO`**: La suma total de las transacciones (positivas para depósitos y negativas para retiros) se almacena en la variable `saldo`.
- **`RETURN`**: Finalmente, la función retorna el saldo calculado.