Proyecto DB SQL

Introducción

Descripción de la Situación de Negocio

En nuestro marketplace, nos dedicamos a facilitar la compra y venta de productos y servicios entre usuarios. Este negocio en línea permite a vendedores y compradores registrados realizar transacciones en una variedad de categorías, desde productos electrónicos hasta servicios profesionales. Los usuarios pueden publicar anuncios de productos o servicios que deseen vender y buscar artículos de su interés. Además, ofrecemos funciones avanzadas, como

calificaciones y reseñas de usuarios, para fomentar la confianza y la transparencia en las transacciones.

Objetivo de la Base de Datos

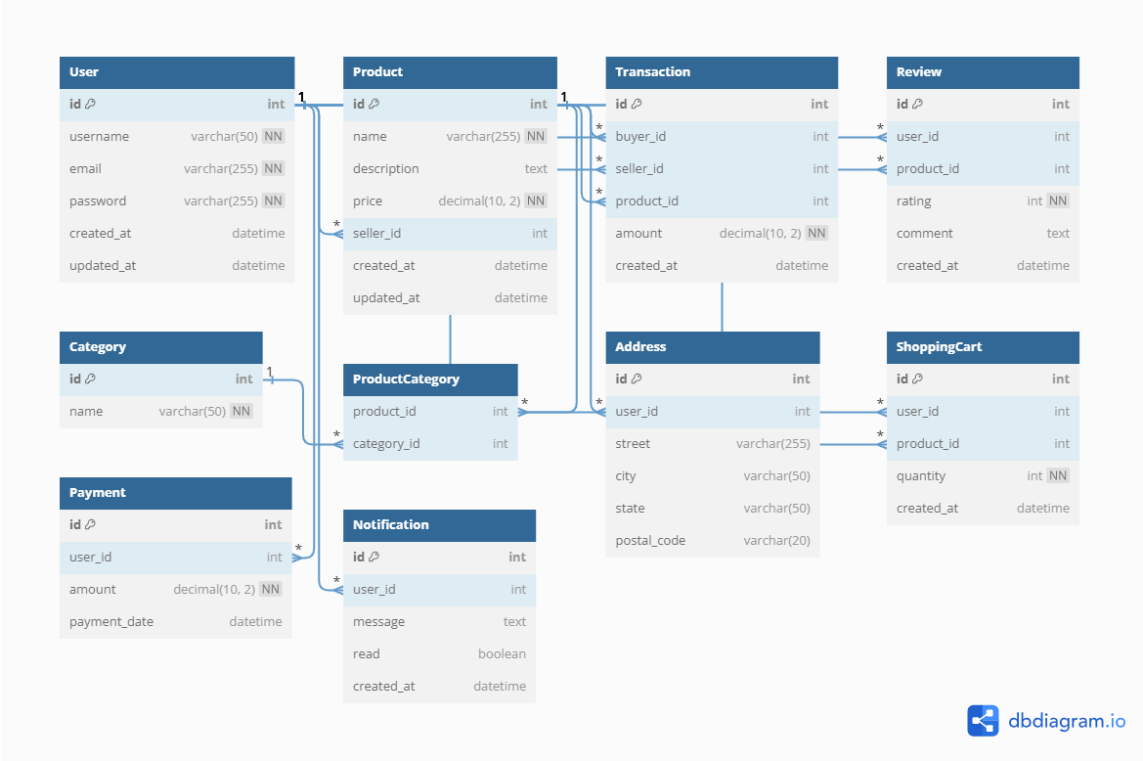
- Almacenar información detallada de los usuarios, incluyendo datos personales, información de contacto y credenciales de inicio de sesión.
- Mantener un registro completo de los productos y servicios disponibles para la venta, incluyendo descripciones, precios y detalles relevantes.
- Registrar las transacciones realizadas, incluyendo la fecha, los vendedores y compradores involucrados, y los detalles de los productos o servicios adquiridos.
- Almacenar calificaciones y reseñas de usuarios para garantizar la confiabilidad de los vendedores y fomentar la transparencia en el marketplace.

Problemas a resolver

1. Gestión Eficiente de la Información: Con la creciente cantidad de usuarios y productos, es esencial mantener y acceder a la información de manera eficiente. La base de datos permitirá una gestión eficaz de los datos de usuarios, productos y transacciones.
2. Seguridad de Datos: La seguridad de los datos personales y financieros de los usuarios es una prioridad. La base de datos garantizará la protección de esta información confidencial mediante medidas de seguridad adecuadas.
3. Facilitación de Transacciones: Facilitará la realización de transacciones, asegurando que los usuarios puedan buscar productos, realizar compras y gestionar sus ventas de manera efectiva.
4. Confiabilidad y Transparencia: Fomentará la confiabilidad entre los usuarios mediante la gestión de calificaciones y reseñas, lo que ayudará a tomar decisiones informadas al comprar o vender productos y servicios.
5. Escalabilidad: La base de datos debe ser escalable para adaptarse al crecimiento del marketplace a lo largo del tiempo sin comprometer el rendimiento.

Entidades y relaciones

Diagrama Entidad-Relación



Descripción de tablas

Tabla: User

Esta tabla almacena información sobre los usuarios registrados en el marketplace.

Tipo de Clave	Nombre del Campo	Tipo de Datos	Descripción
PK	id	INT	Identificador único de usuario.
	username	VARCHAR(50)	Nombre de usuario.
	email	VARCHAR(255)	Dirección de correo electrónico.
	password	VARCHAR(255)	Contraseña del usuario.

	created_at	DATETIME	Fecha de creación del usuario.
	updated_at	DATETIME	Fecha de última actualización.

Tabla: Product

Esta tabla almacena información sobre los productos disponibles en el marketplace.

Tipo de Clave	Nombre del Campo	Tipo de Datos	Descripción
PK	id	INT	Identificador único del producto.
	name	VARCHAR(255)	Nombre del producto.
	description	VARCHAR(255)	Descripción detallada del producto.
	price	DECIMAL(10, 2)	Precio del producto.
FK	seller_id	INT	ID del vendedor.
	created_at	DATETIME	Fecha de creación del producto.
	updated_at	DATETIME	Fecha de última actualización.

Tabla: Transaction

Esta tabla registra las transacciones realizadas en el marketplace.

Tipo de Clave	Nombre del Campo	Tipo de Datos	Descripción
PK	id	INT	Identificador único de la transacción.
FK	buyer_id	INT	ID del comprador.
FK	seller_id	INT	ID del vendedor.
FK	product_id	INT	ID del producto involucrado.
	amount	DECIMAL(10, 2)	Monto de la transacción.
	created_at	DATETIME	Fecha de la transacción.

Tabla: Review

Esta tabla almacena las reseñas y calificaciones de los productos por parte de los usuarios.

Tipo de Clave	Nombre del Campo	Tipo de Datos	Descripción
PK	id	INT	Identificador único de la reseña.
	user_id	INT	ID del usuario que realiza la reseña (clave foránea).
	product_id	INT	ID del producto reseñado (clave foránea).
	rating	INT	Calificación numérica del producto.
	comment	VARCHAR(255)	Comentario o reseña del producto.
	created_at	DATETIME	Fecha de creación de la reseña.

Tabla: Category

Esta tabla almacena las categorías disponibles para clasificar los productos.

Tipo de Clave	Nombre del Campo	Tipo de Datos	Descripción
PK	id	INT	Identificador único de la categoría.
	name	VARCHAR(50)	Nombre de la categoría.

Tabla: ProductCategory

Esta tabla establece relaciones entre productos y categorías.

Tipo de Clave	Nombre del Campo	Tipo de Datos	Descripción
FK	product_id	INT	ID del producto.
FK	category_id	INT	ID de la categoría.

Tabla: Address

Esta tabla almacena las direcciones de los usuarios.

Tipo de Clave	Nombre del Campo	Tipo de Datos	Descripción
PK	id	INT	Identificador único de la dirección.
FK	user_id	INT	ID del usuario asociado.
	street	VARCHAR(255)	Calle y número de la dirección.
	city	VARCHAR(50)	Ciudad de la dirección.
	state	VARCHAR(50)	Estado o provincia de la dirección.
	postal_code	VARCHAR(20)	Código postal de la dirección.

Tabla: ShoppingCart

Esta tabla registra los productos en el carrito de compras de los usuarios.

Tipo de Clave	Nombre del Campo	Tipo de Datos	Descripción
PK	id	INT	Identificador único del carrito de compras.
	user_id	INT	ID del usuario dueño del carrito (clave foránea).
	product_id	INT	ID del producto en el carrito (clave foránea).
	quantity	INT	Cantidad de productos en el carrito.
	created_at	DATETIME	Fecha de creación del registro.

Tabla: Payment

Esta tabla registra los pagos realizados por los usuarios.

Tipo de Clave	Nombre del Campo	Tipo de Datos	Descripción
PK	id	INT	Identificador único del pago.
FK	user_id	INT	ID del usuario que realiza el pago.

	amount	DECIMAL(10, 2)	Monto del pago.
	payment_date	DATETIME	Fecha y hora del pago.

Tabla: Notification

Esta tabla registra las notificaciones enviadas a los usuarios.

Tipo de Clave	Nombre del Campo	Tipo de Datos	Descripción
PK	id	INT	Identificador único de la notificación.
FK	user_id	INT	ID del usuario destinatario de la notificación.
	message	VARCHAR(255)	Mensaje de la notificación.
	read	BOOLEAN	Indica si la notificación ha sido leída.
	created_at	DATETIME	Fecha de creación de la notificación.

Scripts

Creación de tablas

```
-- Crear la base de datos del marketplace
CREATE DATABASE IF NOT EXISTS marketplace_db;

USE marketplace_db;

-- Tabla de Usuarios
CREATE TABLE IF NOT EXISTS user
  (
     id INT auto_increment PRIMARY KEY,username VARCHAR(50) NOT NULL
UNIQUE,
       email VARCHAR(255) NOT NULL UNIQUE,password VARCHAR(255) NOT
NULL,
     created_at DATETIME,updated_at DATETIME
  );

-- Tabla de Productos
CREATE TABLE IF NOT EXISTS product
  (
     id INT auto_increment PRIMARY KEY,name VARCHAR(255) NOT NULL,
      description VARCHAR(255),price DECIMAL(10, 2) NOT NULL,seller_id
INT,
```

```
    created_at DATETIME,updated_at DATETIME,
    FOREIGN KEY (seller_id) REFERENCES user(id)
  );

-- Tabla de Transacciones
CREATE TABLE IF NOT EXISTS transaction
  (
        id INT auto_increment PRIMARY KEY,buyer_id INT,seller_id
INT,product_id INT
    ,amount DECIMAL(10, 2) NOT NULL,created_at DATETIME,
    FOREIGN KEY (buyer_id) REFERENCES user(id),
    FOREIGN KEY (seller_id) REFERENCES user(id),
    FOREIGN KEY (product_id) REFERENCES product(id)
  );

-- Tabla de Reseñas
CREATE TABLE IF NOT EXISTS review
  (
    id INT auto_increment PRIMARY KEY,user_id INT,product_id INT,
    rating INT NOT NULL,comment VARCHAR(255),created_at DATETIME,
    FOREIGN KEY (user_id) REFERENCES user(id),
    FOREIGN KEY (product_id) REFERENCES product(id)
  );

-- Tabla de Categorías
CREATE TABLE IF NOT EXISTS category
  (
    id INT auto_increment PRIMARY KEY,name VARCHAR(50) NOT NULL
  );

-- Tabla de Relación entre Productos y Categorías
CREATE TABLE IF NOT EXISTS productcategory
  (
    product_id INT,category_id INT,
    FOREIGN KEY (product_id) REFERENCES product(id),
    FOREIGN KEY (category_id) REFERENCES category(id)
  );

-- Tabla de Direcciones
CREATE TABLE IF NOT EXISTS address
  (
          id INT auto_increment PRIMARY KEY,user_id INT,street
VARCHAR(255),city
    VARCHAR(50),state VARCHAR(50),postal_code VARCHAR(20),
    FOREIGN KEY (user_id) REFERENCES user(id)
  );

-- Tabla de Carritos de Compras
CREATE TABLE IF NOT EXISTS shoppingcart
  (
    id INT auto_increment PRIMARY KEY,user_id INT,product_id INT,
```

```
    quantity INT NOT NULL,created_at DATETIME,
    FOREIGN KEY (user_id) REFERENCES user(id),
    FOREIGN KEY (product_id) REFERENCES product(id)
  );

-- Tabla de Pagos
CREATE TABLE IF NOT EXISTS payment
  (
    id INT auto_increment PRIMARY KEY,user_id INT,
    amount DECIMAL(10, 2) NOT NULL,payment_date DATETIME,
    FOREIGN KEY (user_id) REFERENCES user(id)
  );

-- Tabla de Notificaciones
CREATE TABLE IF NOT EXISTS notification
  (
          id  INT  auto_increment  PRIMARY  KEY,user_id  INT,message
VARCHAR(255),READ
    BOOLEAN DEFAULT false,created_at DATETIME,
    FOREIGN KEY (user_id) REFERENCES user(id)
  );
```

Inserción de datos

```
-- Inserción de datos en la tabla User
INSERT INTO user
            (username,email,password,created_at,updated_at)
VALUES      ('JuanPerez','juanperez@email.com','contrasena1',
             '2023-01-15 10:30:00',
             '2023-03-20 15:45:00'),
            ('MariaLopez','marialopez@email.com','contrasena2',
             '2023-02-05 14:20:00',
             '2023-04-10 09:15:00'),
            ('CarlosGomez','carlosgomez@email.com','contrasena3',
             '2023-03-10 08:45:00',
             '2023-05-25 12:30:00'),

('LauraFernandez','laurafernandez@email.com','contrasena4',
             '2023-04-20 17:10:00',
             '2023-06-30 16:05:00'),

('PedroRodriguez','pedrorodriguez@email.com','contrasena5',
             '2023-05-25 11:55:00',
             '2023-07-05 14:25:00'),
            ('SofiaTorres','sofiatorres@email.com','contrasena6',
             '2023-06-10 09:30:00',
             '2023-08-10 18:40:00'),
            ('AndresSanchez','andressanchez@email.com','contrasena7',
             '2023-07-15 13:15:00',
             '2023-09-15 10:30:00'),
```

```
('IsabellaMartinez','isabellamartinez@email.com','contrasena8',
             '2023-08-05 16:40:00','2023-09-25 19:15:00'),
            ('LuisGonzalez','luisgonzalez@email.com','contrasena9',
             '2023-09-01 10:20:00',
             '2023-09-10 11:50:00'),
            ('AnaRamirez','anaramirez@email.com','contrasena10',
             '2023-09-10 14:30:00',
             '2023-09-15 15:20:00');

-- Inserción de datos en la tabla Product
INSERT INTO product
            (name,description,price,seller_id,created_at,updated_at)
VALUES          ('Balón de Fútbol Adidas','Balón de fútbol de alta
calidad',25.5
             ,1,
             '2023-01-05 11:30:00','2023-01-10 16:45:00'),
                     ('Raqueta de Tenis Wilson','Raqueta de tenis
profesional',89.9,2,
             '2023-02-10 09:20:00','2023-02-15 15:10:00'),
            ('Zapatillas de Running Nike','Zapatillas deportivas para
correr',
             65.0,3,
             '2023-03-15 13:45:00','2023-03-20 17:30:00'),
                ('Balón de Baloncesto Spalding','Balón de baloncesto
oficial',19.8
             ,4,
             '2023-04-20 12:10:00','2023-04-25 14:55:00'),
                ('Tabla de Surf Quiksilver','Tabla de surf para olas
grandes',75.2,5
             ,
             '2023-05-25 10:25:00','2023-05-30 16:15:00'),
              ('Patines en Línea Rollerblade','Patines en línea para
patinaje',
             54.3,6,
             '2023-06-01 15:30:00','2023-06-05 18:20:00'),
                ('Raqueta de Squash Head','Raqueta de squash de alto
rendimiento',
             72.7,7,
             '2023-07-05 08:30:00','2023-07-10 12:15:00'),
              ('Casco de Ciclismo Giro','Casco ligero y seguro para
ciclistas',
             35.1,8,
             '2023-08-10 09:30:00','2023-08-15 13:20:00'),
                 ('Pelota de Golf Titleist','Pelota de golf de tour
profesional',45.6
             ,9,
             '2023-09-10 11:10:00','2023-09-15 15:50:00'),
             ('Red de Voleibol Mikasa','Red de voleibol resistente y
duradera',
             29.0,10,
```

```
            '2023-09-15 14:20:00','2023-09-20 17:15:00');

-- Inserción de datos en la tabla Transaction
INSERT INTO transaction
            (buyer_id,seller_id,product_id,amount,created_at)
VALUES      (2,1,1,25.5,'2023-01-15 09:30:00'),
            (3,2,2,89.9,'2023-02-10 10:20:00'),
            (4,3,3,65.0,'2023-03-15 11:45:00'),
            (5,4,4,19.8,'2023-04-20 14:10:00'),
            (6,5,5,75.2,'2023-05-25 15:25:00'),
            (7,6,6,54.3,'2023-06-01 16:30:00'),
            (8,7,7,72.7,'2023-07-05 09:30:00'),
            (9,8,8,35.1,'2023-08-10 12:40:00'),
            (10,9,9,45.6,'2023-09-10 13:10:00'),
            (1,10,10,29.0,'2023-09-15 10:15:00');

-- Inserción de datos en la tabla Review
INSERT INTO review
            (user_id,product_id,rating,comment,created_at)
VALUES              (1,1,4,'Excelente  balón  de  fútbol','2023-01-16
08:45:00'),
                     (2,2,5,'La  mejor  raqueta  de  tenis','2023-02-11
11:30:00'),
                   (3,3,4,'Zapatillas cómodas para correr','2023-03-16
13:20:00'),
            (4,4,3,'Buen balón de baloncesto','2023-04-21 16:10:00'),
            (5,5,5,'Gran tabla de surf','2023-05-26 17:45:00'),
            (6,6,4,'Patines de alta calidad','2023-06-02 10:30:00'),
                (7,7,5,'Raqueta de squash impresionante','2023-07-06
12:15:00'),
                     (8,8,4,'Casco  cómodo  para  ciclismo','2023-08-11
14:20:00'),
            (9,9,3,'Buena pelota de golf','2023-09-11 15:50:00'),
                   (10,10,4,'Red  de  voleibol  resistente','2023-09-16
09:20:00');

-- Inserción de datos en la tabla Category
INSERT INTO category
            (name)
VALUES      ('Fútbol'),
            ('Tenis'),
            ('Running'),
            ('Baloncesto'),
            ('Surf'),
            ('Patinaje'),
            ('Squash'),
            ('Ciclismo'),
            ('Golf'),
            ('Voleibol');
```

```sql
-- Inserción de datos en la tabla ProductCategory (relación entre productos y categorías)
INSERT INTO productcategory
            (product_id,category_id)
VALUES      (1,1),
            (2,2),
            (3,3),
            (4,4),
            (5,5),
            (6,6),
            (7,7),
            (8,8),
            (9,9),
            (10,10);

-- Inserción de datos en la tabla Address
INSERT INTO address
            (user_id,street,city,state,postal_code)
VALUES      (1,'Calle A','Buenos Aires','Buenos Aires','1234'),
            (2,'Calle B','Córdoba','Córdoba','5678'),
            (3,'Calle C','Rosario','Santa Fe','9012'),
            (4,'Calle D','Mendoza','Mendoza','3456'),
            (5,'Calle E','Mar del Plata','Buenos Aires','7890'),
            (6,'Calle F','Salta','Salta','2345'),
            (7,'Calle G','La Plata','Buenos Aires','6789'),
            (8,'Calle H','Tucumán','Tucumán','0123'),
            (9,'Calle I','Neuquén','Neuquén','4567'),
            (10,'Calle J','San Juan','San Juan','8901');

-- Inserción de datos en la tabla ShoppingCart
INSERT INTO shoppingcart
            (user_id,product_id,quantity,created_at)
VALUES      (1,1,2,'2023-01-20 10:30:00'),
            (2,2,1,'2023-02-15 14:20:00'),
            (3,3,3,'2023-03-20 08:45:00'),
            (4,4,2,'2023-04-25 17:10:00'),
            (5,5,1,'2023-05-30 11:55:00'),
            (6,6,4,'2023-06-05 09:30:00'),
            (7,7,2,'2023-07-10 13:15:00'),
            (8,8,1,'2023-08-15 16:40:00'),
            (9,9,3,'2023-09-10 10:20:00'),
            (10,10,2,'2023-09-15 14:30:00');

-- Inserción de datos en la tabla Payment
INSERT INTO payment
            (user_id,amount,payment_date)
VALUES      (1,102.0,'2023-01-25 12:30:00'),
            (2,89.9,'2023-02-20 15:45:00'),
            (3,195.0,'2023-03-25 09:20:00'),
            (4,39.6,'2023-04-30 14:10:00'),
            (5,75.2,'2023-05-05 15:25:00'),
```

```
            (6,217.2,'2023-06-10 16:30:00'),
            (7,145.4,'2023-07-15 09:30:00'),
            (8,35.1,'2023-08-20 12:40:00'),
            (9,136.8,'2023-09-01 13:10:00'),
            (10,58.0,'2023-09-15 10:15:00');

-- Inserción de datos en la tabla Notification
INSERT INTO notification
            (user_id,message,READ,created_at)
VALUES      (1,'¡Nuevo producto en oferta!',0,'2023-01-16 08:45:00'),
            (2,'Tienes una nueva revisión en tu producto',0,
             '2023-02-11 11:30:00'),
                   (3,'Confirmación de pago recibida',0,'2023-03-16
13:20:00'),
               (4,'Actualización de estado de entrega',0,'2023-04-21
16:10:00'),
                                  (5,'Oferta especial en productos
deportivos',0,'2023-05-26 17:45:00'
),
              (6,'Mensaje de bienvenida a la plataforma',0,'2023-06-02
10:30:00'),
                  (7,'Nuevas notificaciones disponibles',0,'2023-07-06
12:15:00'),
                   (8,'Producto destacado de la semana',0,'2023-08-11
14:20:00'),
                (9,'Recordatorio de carrito de compras',0,'2023-09-11
15:50:00'),
                   (10,'¡Gracias por unirse a nosotros!',0,'2023-09-16
09:20:00');
```

Vistas principales

TOP 3 de productos más vendidos

```
CREATE OR REPLACE view topsellingproducts
AS
  SELECT p.name         AS product_name,
         Sum(t.amount) AS total_sales
  FROM   product p
         JOIN transaction t
           ON p.id = t.product_id
  GROUP  BY p.id
  ORDER  BY total_sales DESC
  LIMIT  3;
```

Facturación mensual

```
CREATE OR REPLACE view monthlyrevenue
```

```
AS
  SELECT Date_format(t.created_at, '%Y-%m') AS month,
         Sum(t.amount) AS total_revenue
  FROM   transaction t
  GROUP  BY month
  ORDER  BY month;
```

Usuarios que más compras realizó

```
CREATE OR REPLACE view topbuyer
AS
  SELECT u.username AS top_buyer,Count(t.id) AS total_transactions
  FROM   user u
         LEFT JOIN transaction t
                ON u.id = t.buyer_id
  GROUP  BY u.id
  ORDER  BY total_transactions DESC
  LIMIT  1;
```

Funciones

Calcular el promedio de calificaciones para un producto específico:

Esta función calcula el promedio de calificaciones de un producto en función de las reseñas asociadas a ese producto.

```
DELIMITER //

CREATE function
  calculateaveragerating(productid INT) returns DECIMAL(3,2)
begin
  DECLARE avgrating DECIMAL(3,2);SELECT Avg(rating)
  INTO   avgrating
  FROM   review
  WHERE  product_id = productid;RETURN avgrating;END //
delimiter ;
```

Obtener la lista de productos en el carrito de compras de un usuario:

Esta función permite recuperar la lista de productos en el carrito de compras de un usuario en función de su ID de usuario.

```
DELIMITER //
```

```
CREATE function
  getcartproducts(userid INT) returns VARCHAR(255)
begin
  DECLARE cartproducts VARCHAR(255);SELECT Group_concat(p.name)
  INTO   cartproducts
  FROM   shoppingcart sc
  JOIN   product p
  ON     sc.product_id = p.id
  WHERE  sc.user_id = userid;RETURN cartproducts;END //
delimiter ;
```

Calcular el total gastado por un usuario en transacciones:

Esta función calcula el total gastado por un usuario en todas sus transacciones registradas en la base de datos.

```
DELIMITER //

CREATE function
  calculatetotalspent(userid INT) returns DECIMAL(10,2)
begin
  DECLARE totalspent DECIMAL(10,2);SELECT Sum(amount)
  INTO   totalspent
  FROM   transaction
  WHERE  buyer_id = userid;RETURN totalspent;END //
delimiter ;
```

Stored Procedures

Procedimiento para calcular el total gastado por un usuario

Este procedimiento acepta como entrada el ID de un usuario y calcula el total gastado por ese usuario en transacciones de compra.

```
DELIMITER //

CREATE PROCEDURE
  calculatetotalspentbyuser(IN userid      INT,
                            OUT totalspent DECIMAL(10, 2))
begin
  SELECT sum(amount)
  INTO   totalspent
```

```
  FROM   transaction
  WHERE  buyer_id = userid;END;//
delimiter ;
```

Procedimiento para agregar productos al carrito de compras

Este procedimiento permite a un usuario agregar productos a su carrito de compras. Toma como entrada el ID del usuario, el ID del producto y la cantidad a agregar. Luego, inserta una fila en la tabla shoppingcart para registrar el producto en el carrito del usuario.

```
DELIMITER //
CREATE PROCEDURE
  addproducttoshoppingcart(IN userid    INT,
                           IN productid INT,
                           IN quantity  INT)
begin
  INSERT INTO shoppingcart
              (
                          user_id,
                          product_id,
                          quantity,
                          created_at
              )
              VALUES
              (
                          userid,
                          productid,
                          quantity,
                          now()
              );END;//
delimiter ; DELIMITER //
CREATE PROCEDURE
  calculatetotalspentbyuser(IN userid      INT,
                            OUT totalspent DECIMAL(10, 2))
begin
  SELECT sum(amount)
  INTO   totalspent
  FROM   transaction
  WHERE  buyer_id = userid;END;//
delimiter ;
```

Procedimiento para marcar una notificación como leída

Este procedimiento permite a un usuario marcar una notificación como leída. Toma como entrada el ID de la notificación y el ID del usuario. Luego, actualiza el estado de la notificación a "leída" en la tabla notification.

```
DELIMITER //
CREATE PROCEDURE
```

```
  marknotificationasread(IN notificationid INT,
                         IN userid         INT)
begin
  UPDATE notification
  SET    READ = TRUE
  WHERE  id = notificationid
  AND    user_id = userid;END;//
delimiter ;
```

Triggers

Creación tablas de registro de usuarios y producto

```
CREATE TABLE IF NOT EXISTS user_log
            (
                        id          INT auto_increment PRIMARY KEY,
                        user_id     INT NOT NULL,
                        action_type VARCHAR(50) NOT NULL,
                        action_date DATE NOT NULL,
                        action_time TIME NOT NULL,
                        user_action VARCHAR(255) NOT NULL,
                        FOREIGN KEY (user_id) REFERENCES user(id)
            );

CREATE TABLE IF NOT EXISTS product_log
            (
                          id              INT auto_increment PRIMARY
KEY,
                        product_id     INT NOT NULL,
                        action_type    VARCHAR(50) NOT NULL,
                        action_date    DATE NOT NULL,
                        action_time    TIME NOT NULL,
                        product_action VARCHAR(255) NOT NULL,
                               FOREIGN KEY (product_id) REFERENCES
product(id)
            );
```

Trigger 1: registrar inserciones de nuevos usuarios

```
DELIMITER //
CREATE TRIGGER user_before_insert
  BEFORE
  INSERT
  ON user FOR EACH row begin
  INSERT INTO user_log
              (
                        user_id,
                        action_type,
```

```
                        action_date,
                        action_time,
                        user_action
                )
                VALUES
                (
                        new.id,
                        'INSERT',
                        curdate(),
                        curtime(),
                                concat('Nuevo usuario creado: ',
new.username)
                );
END;//
delimiter ;
```

Trigger 2: registrar actualizaciones de usuarios

```
DELIMITER //
CREATE TRIGGER user_after_update
  after
  UPDATE
  ON user FOR EACH row begin
  INSERT INTO user_log
                (
                        user_id,
                        action_type,
                        action_date,
                        action_time,
                        user_action
                )
                VALUES
                (
                        new.id,
                        'UPDATE',
                        curdate(),
                        curtime(),
                                concat('Usuario   actualizado:   ',
new.username)
                );
END;//
delimiter ;
```

Trigger 3: registrar actualizaciones de productos

```
CREATE TRIGGER product_after_update
  after
  UPDATE
```

```
  ON product FOR EACH row begin
  INSERT INTO product_log
              (
                          product_id,
                          action_type,
                          action_date,
                          action_time,
                          product_action
              )
              VALUES
              (
                          new.id,
                          'UPDATE',
                          curdate(),
                          curtime(),
                          concat('Producto actualizado: ', new.name)
              );
END;//
delimiter ;
```