ASSIGNMENT-1

DATABASE MANAGEMENT SYSTEM
CSA0593

SATHISH.R
192324204

Online Shopping Cart and Order Processing System

System Overview:

An Online Shopping Cart and Order Processing System is designed to help e-commerce websites manage users, products, shopping carts, orders, and payments. This system allows users to browse products, add them to a cart, place orders, and make payments. The database ensures all operations are handled efficiently, with referential integrity, and provides reports to help the business track trends, popular products, and revenue.

Database Structure

1. Tables:

Users: This table stores information about the customers who use the website. It includes personal details, contact information, and login credentials.

Column Name	Data Type	Description
user_id	INT	Primary key, unique user ID
name	VARCHAR	User's full name
email	VARCHAR	User's email address
password	VARCHAR	User's hashed password
phone	VARCHAR	User's contact number
address	VARCHAR	User's shipping address

Products: This table holds information about the products available for purchase.

Column Name	Data Type	Description
product_id	INT	Primary key, unique product ID
name	VARCHAR	Name of the product
description	TEXT	Description of the product
price	DECIMAL	Price of the product
stock_quantity	INT	Quantity of the product available in stock
category	VARCHAR	Category of the product (e.g., electronics)

Carts: This table keeps track of the items a user has added to their cart before they proceed to checkout.

Column Name	Data Type	Description
cart_id	INT	Primary key, unique cart ID
user_id	INT	Foreign key, linked to Users table
created_at	TIMESTAMP	Timestamp when the cart was created

Cart_Items: This table connects users' carts to the products they have added.

Column Name	Data Type	Description
cart_item_id	INT	Primary key, unique cart item ID
cart_id	INT	Foreign key, linked to Carts table
product_id	INT	Foreign key, linked to Products table
quantity	INT	Quantity of the product in the cart
price	DECIMAL	Price of the product at the time of adding to cart

Orders: This table stores information about orders placed by users.

Column Name	Data Type	Description
order_id	INT	Primary key, unique order ID
user_id	INT	Foreign key, linked to Users table
order_date	TIMESTAMP	Timestamp when the order was placed
total_amount	DECIMAL	Total price of the order
shipping_address	VARCHAR	Shipping address for the order
order_status	VARCHAR	Status of the order (e.g., processing, shipped)

Order_Items: This table links the products in an order.

Column Name	Data Type	Description
order_item_id	INT	Primary key, unique order item ID
order_id	INT	Foreign key, linked to Orders table
product_id	INT	Foreign key, linked to Products table
quantity	INT	Quantity of the product in the order
price	DECIMAL	Price of the product at the time of order

Payments: This table stores payment information once an order is paid.

Column Name	Data Type	Description
payment_id	INT	Primary key, unique payment ID
order_id	INT	Foreign key, linked to Orders table
payment_date	TIMESTAMP	Date and time when the payment was made
payment_amount	DECIMAL	Total amount paid
payment_status	VARCHAR	Status of payment (e.g., completed, failed)

System Constraints:

Referential Integrity:

1. Users must be linked to their respective carts and orders.
2. Products must be linked to cart items and order items.
3. Orders and payments must be related.

Foreign Key Constraints:

- The user_id in the Carts table references the user_id in the Users table.
- The cart_id in the Cart_Items table references the cart_id in the Carts table.
- The product_id in the Cart_Items, Order_Items table references the product_id in the Products table.
- The user_id in the Orders table references the user_id in the Users table.
- The order_id in the Payments table references the order_id in the Orders table.

Stored Procedures:

Add Item to Cart: This procedure adds a product to a user's cart and updates the cart items.

```
CREATE PROCEDURE AddItemToCart(IN user_id INT, IN product_id INT, IN quantity INT)BEGIN

 DECLARE cart_id INT;

 -- Check if the user has an existing cart

 SELECT cart_id INTO cart_id FROM Carts WHERE user_id = user_id;

 IF cart_id IS NULL THEN

  -- If no cart exists, create a new cart

  INSERT INTO Carts(user_id) VALUES(user_id);

  SET cart_id = LAST_INSERT_ID();
```

```
 END IF;


 -- Add item to cart

 INSERT INTO Cart_Items(cart_id, product_id, quantity, price)

 SELECT cart_id, product_id, quantity, price FROM Products WHERE product_id = product_id;END;
```

Place Order: This procedure places an order, calculates the total amount, and updates the stock.

```
sql

CREATE PROCEDURE PlaceOrder(IN user_id INT)BEGIN

 DECLARE order_id INT;

 DECLARE total_amount DECIMAL(10, 2);


 -- Create the order

 INSERT INTO Orders(user_id, order_date, total_amount)

 VALUES(user_id, NOW(), 0);

 SET order_id = LAST_INSERT_ID();


 -- Calculate total amount and add order items

 DECLARE done INT DEFAULT 0;

 DECLARE cur CURSOR FOR

   SELECT product_id, quantity, price FROM Cart_Items WHERE cart_id IN (SELECT cart_id FROM Carts WHERE user_id = user_id);

 DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;


 OPEN cur;

 read_loop: LOOP

   FETCH cur INTO product_id, quantity, price;

   IF done THEN

     LEAVE read_loop;

   END IF;


   -- Insert into Order_Items
```

```
    INSERT INTO Order_Items(order_id, product_id, quantity, price) VALUES(order_id, product_id, quantity, price);



    -- Update product stock

    UPDATE Products SET stock_quantity = stock_quantity - quantity WHERE product_id = product_id;

  END LOOP;



  CLOSE cur;



  -- Update total order amount

  SELECT SUM(quantity * price) INTO total_amount FROM Order_Items WHERE order_id = order_id;

  UPDATE Orders SET total_amount = total_amount WHERE order_id = order_id;END;
```

Process Payment: This procedure processes the payment for an order and updates the order status.

```
sql

CREATE PROCEDURE ProcessPayment(IN order_id INT, IN payment_amount DECIMAL)BEGIN

  DECLARE payment_status VARCHAR(50);

    -- Insert payment record

  INSERT INTO Payments(order_id, payment_date, payment_amount, payment_status)

  VALUES(order_id, NOW(), payment_amount, 'Completed');

    -- Update order status

  UPDATE Orders SET order_status = 'Paid' WHERE order_id = order_id;END;
```

Triggers:

Update Stock Level When Order is Placed: This trigger ensures that the stock quantity is updated when an order is placed.

```
sql

CREATE TRIGGER UpdateStockAfterOrder

AFTER INSERT ON Order_ItemsFOR EACH ROWBEGIN

  UPDATE Products SET stock_quantity = stock_quantity - NEW.quantity

  WHERE product_id = NEW.product_id;END;
```

Update Stock Level When Item is Removed from Cart: This trigger ensures that stock levels are updated when an item is removed from a cart.

```
sql

CREATE TRIGGER UpdateStockAfterCartRemoval

AFTER DELETE ON Cart_ItemsFOR EACH ROWBEGIN

  UPDATE Products SET stock_quantity = stock_quantity + OLD.quantity

  WHERE product_id = OLD.product_id;END;
```

Conclusion:

The Online Shopping Cart and Order Processing System efficiently manages key e-commerce functions such as user accounts, product inventory, cart management, order processing, and payments. With strong data integrity, automated processes through stored procedures and triggers, and the ability to generate insightful reports, the system ensures smooth operations and an optimal shopping experience for users. This comprehensive approach helps businesses streamline their processes while maintaining accuracy and reliability in managing orders and inventory.