DBMS Project
Gada Electronics Online Retail Store

Group 10
Vikranth Udandarao - 22570, Tharun Harish - 22541,
Hemanth Dindigallu - 22212, Aditya Prasad - 22036

Conflicting Transactions

```
1. try:
    conn = mysql_connection()
    cursor = conn.cursor(dictionary=True)
    conn.start_transaction()

    # Lock the relevant tables including aliases properly
    cursor.execute("""
        LOCK TABLES
        Cart AS c WRITE,
        Product AS p WRITE,
        Warehouse AS w WRITE,
        Orders WRITE;
    """)

    # Fetch and attempt to update warehouse quantities
    cursor.execute("""
        SELECT c.Product_ID, c.Quantity, p.Price, p.Name, w.Pincode AS Warehouse_ID
        FROM Cart AS c
        JOIN Product AS p ON c.Product_ID = p.Product_ID
        JOIN Warehouse AS w ON p.Product_ID = w.Product_ID
        WHERE c.Cart_ID = %s AND c.Customer_ID = %s
    """, (cart_id, customer_id))
    cart_items = cursor.fetchall()

      # Assuming Payment_ID is generated somewhere within your application
    payment_id = session['payment_id']
    order_id = "ORD" + ''.join(random.choices(string.ascii_uppercase + string.digits, k=10)

    print(payment_id)
    print(order_id)
    print(cart_items)

    for item in cart_items:
        cursor.execute("""
            UPDATE Warehouse AS w
            SET w.Warehouse_Quantity = w.Warehouse_Quantity - %s
            WHERE w.Product_ID = %s AND w.Pincode = %s AND w.Warehouse_Quantity >= %s
```

```
        """, (item['Quantity'], item['Product_ID'], item['Warehouse_ID'], item['Quantity'])
        print(item['Quantity'])

        # Insert into Orders table
        cursor.execute("""
            INSERT INTO Orders (Order_ID, Customer_ID, Payment_ID, Product_ID, Quantity)
            VALUES (%s, %s, %s, %s, %s)
        """, (order_id, customer_id, payment_id, item['Product_ID'], item['Quantity']))
        print("done")

    # Update payment status
    cursor.execute("UPDATE Payment SET Status = 'Completed' WHERE Payment_ID", (payment_id)

    if any([cursor.rowcount == 0 for item in cart_items]):
        raise ValueError("Unable to complete order due to insufficient stock for one or mor

    conn.commit()
    total = sum(item['Price'] * item['Quantity'] for item in cart_items)

    flash('Order placed successfully!', 'success')
    return render_template('receipt.html', order_id='some_id', order_date='some_date',
                           payment_method='Credit Card', total=total, cart_items=cart_items

except MySQLError as err:
    conn.rollback()
    flash('A database error occurred. Please try again.', 'error')
    return redirect(url_for('checkout'))
except ValueError as ve:
    conn.rollback()
    flash(str(ve), 'error')
    return redirect(url_for('checkout'))
finally:
    cursor.execute("UNLOCK TABLES;")  # Ensure tables are unlocked even if an error occurs
    cursor.close()
    conn.close()

2. try:
    conn = mysql_connection()
    cursor = conn.cursor(dictionary=True)
    conn.start_transaction()

    # Lock the relevant tables including aliases properly
    cursor.execute("""
        LOCK TABLES
        Customer WRITE;
    """)

    cursor.execute("INSERT INTO Customer (Customer_ID, Name, Email, PhoneNo, Password) VALU
                (Customer_ID, name, email, phone, password))

    new_cart_id = generate_cart_id()
    # Insert new cart and pending payment entry
    new_payment_id = insert_new_cart_and_payment(cursor, new_cart_id, Customer_ID)
```

```
    conn.commit()
    session['customer_id'] = Customer_ID
    session['payment_id'] = new_payment_id

    session['cart_id'] = new_cart_id
    session['cart_count'] = 0
    session['cart'] = {}

    return redirect(url_for('index'))

except MySQLError as e:
    conn.rollback()
    if e.errno == 1062:
        error_message = "An account with this email or phone number already exists."
    else:
        error_message = "An unexpected error occurred. Please try again later."
    return render_template('error.html', error_message=error_message, back_url=url_for('reg

finally:
    cursor.close()
    conn.close()
```

Non Conflicting Transactions

```
1. try:
    conn = mysql_connection()
    cursor = conn.cursor(dictionary=True)
    conn.start_transaction()
    cursor.execute("SELECT * FROM Customer WHERE Email = %s", (email,))
    user = cursor.fetchone()

    if user and bcrypt.checkpw(password.encode(), user['Password'].encode()):
        session['customer_id'] = user['Customer_ID']

        # Handle user address and warehouse ID setup
        setup_user_session(user, cursor)

        # Handle pending payments and cart setup
        handle_pending_payments(user['Customer_ID'], cursor)

        conn.commit()
        return redirect(url_for('index'))
    else:
        flash('Invalid email or password', 'error')
        return redirect(url_for('login'))

except mysql.connector.Error as err:
    # Handle specific MySQL error
    print("MySQL Error: ", str(err))
```

```
    return render_template('error.html', error_message="A database error occurred.", back_u
finally:
    cursor.close()
    conn.close()

2. if action == "remove":
    product_id = request.form['product_id']
    cart_id = session['cart_id']
    customer_id = session['customer_id']

    conn = mysql_connection()
    cursor = conn.cursor(dictionary=True)

    conn.start_transaction()

    # Get the quantity of the product to be removed
    cursor.execute("""
        SELECT Quantity FROM Cart
        WHERE Cart_ID = %s AND Product_ID = %s
    """, (cart_id, product_id))

    product = cursor.fetchone()

    if product:
        remove_quantity = int(product['Quantity'])
        print("remove_quantity",remove_quantity)

        # Delete the product from the Cart table
        query = "DELETE FROM Cart WHERE Customer_ID = %s AND Product_ID = %s"
        cursor.execute(query, (customer_id, product_id))
        conn.commit()

3. elif action == 'add':
    product_id = request.form['product_id']
    add_quantity = int(request.form['quantity'])
    customer_id = session['customer_id']

    conn = mysql_connection()
    cursor = conn.cursor(dictionary=True)

    conn.start_transaction()
    # Check if the product already exists in the user's cart
    cursor.execute("""
        SELECT Quantity FROM Cart
        WHERE Cart_ID = %s AND Customer_ID = %s AND Product_ID = %s
    """, (cart_id, customer_id, product_id))
    existing_product = cursor.fetchone()

    # Fetch the product details for price and discount
    cursor.execute("SELECT Price, Discount FROM Product WHERE Product_ID = %s", (product_id
    product_data = cursor.fetchone()

    print(product_data)
```

```
    if product_data:
        price = product_data['Price']
        discount = product_data['Discount']
        offer = price * (1 - discount / 100)
        offer = min(offer, 99999999.99)  # Ensure offer doesn't exceed DECIMAL(10, 2) range

        if existing_product:
            # Product exists in cart, so update the quantity
            print("testtingngg",existing_product)
            new_quantity = int(existing_product['Quantity']) + add_quantity
            cursor.execute("""
                UPDATE Cart SET Quantity = %s
                WHERE Cart_ID = %s AND Customer_ID = %s AND Product_ID = %s
            """, (new_quantity, cart_id, customer_id, product_id))



        else:
            # Product does not exist in cart, so insert as new entry
            cursor.execute("""
                INSERT INTO Cart (Cart_ID, Customer_ID, Product_ID, Price, Offer, Quantity)
                VALUES (%s, %s, %s, %s, %s, %s)
            """, (cart_id, customer_id, product_id, price, offer, add_quantity))

        conn.commit()
        cursor.close()
        conn.close()

4. conn = mysql_connection()
    cursor = conn.cursor()
    conn.start_transaction()

    cursor.execute("""
        INSERT INTO Address (Address_ID, Customer_ID, Street_Name, Flat_No, City, State, Pi
        VALUES (%s, %s, %s, %s, %s, %s, %s)
    """, (address_id, customer_id, street_name, flat_no, city, state, pincode))

    conn.commit()
    cursor.close()
    conn.close()
```

Contributions

All the group members contributed equally to this submission.