Conflicting Transactions:

Transaction 1: Place Order
This transaction modifies the quantity field in the Products table when placing an order.
Concurrent executions of this transaction could lead to race conditions if multiple orders are being placed for the same product simultaneously.

Transaction 2: Remove Product by Vendor
This transaction also modifies the quantity field in the Products table by reducing the quantity of a product when the vendor removes it from inventory.
Concurrent executions of this transaction with the "Place Order" transaction could lead to inconsistencies in the product quantity.

Transaction 3: Restock Product
Similar to the "Remove Product by Vendor" transaction, this transaction modifies the quantity field in the Products table by increasing the quantity of a product when the vendor restocks it.
Concurrent executions of this transaction with the "Place Order" transaction could lead to inconsistencies in the product quantity.

Transaction 4: Get Vendor Name
Although this transaction does not directly modify data, it retrieves data from the Vendors table.
Concurrent executions of this transaction with transactions modifying vendor data could potentially lead to inconsistencies if isolation levels are not appropriately set.

Non-Conflicting Transactions:

Transaction 5: Customer Login
This transaction validates customer credentials by performing a read operation on the Customers table based on email and password.
It does not modify any data and can execute concurrently with other read operations without causing conflicts.

Transaction 6: List Categories and List Products
These transactions involve reading data from the same table (categories) but do not modify any data. Hence, they are non-conflicting.

Transaction 7: Get Customer Name and Get Customer ID
Both transactions involve reading data from the customers table based on the email column. Since they are only reading data and not modifying it, they can execute concurrently without conflicts.

Transaction 8: Get Total Sales and Get Sales by Vendor
These transactions involve reading aggregate data (sum of sales) from multiple tables (orders, order_details, products, product_inventory, vendors). Since they are read-only operations and do not modify any data, they can execute concurrently without conflicts.

```
import mysql. connector
from datetime import datetime


# Updated place_order function
def place_order(customer_id, product_id, quantity):
   try:
       conn = mysql.connector.connect(host='localhost', username='root',
password='mysql2023', database='drop_24')
       my_cursor = conn.cursor()

       conn.start_transaction()

       my_cursor.execute("SELECT quantity FROM Products WHERE product_id = %s
FOR UPDATE", (product_id,))
       product_row = my_cursor.fetchone()

       if product_row is None:
           print(f"Product with ID {product_id} does not exist.")
           conn.close()
           return

       available_quantity = product_row[0]

       if available_quantity >= quantity:
           my_cursor.execute("INSERT INTO Orders (customer_id, order_date)
VALUES (%s, %s)", (customer_id, datetime.now()))
           order_id = my_cursor.lastrowid

           my_cursor.execute("UPDATE Products SET quantity = quantity - %s
WHERE product_id = %s", (quantity, product_id))

           my_cursor.execute("INSERT INTO Order_details (order_id, product_id,
quantity) VALUES (%s, %s, %s)", (order_id, product_id, quantity))
```

```
            my_cursor.execute("UPDATE product_inventory SET quantity = quantity
- %s WHERE product_id = %s", (quantity, product_id))

            conn.commit()
            print("\nOrder placed successfully!")
        else:
            print(f"Insufficient stock for product {product_id}. Available
quantity: {available_quantity}")

    except mysql.connector.Error as err:
        print("Error:", err)
        conn.rollback()

    finally:
        if 'conn' in locals():
            my_cursor.close()
            conn.close()




def low_stock_products():
    conn=mysql. connector .connect(host='localhost', username='root', password =
'mysql2023',database = 'drop_24')
    my_cursor = conn.cursor()

    my_cursor.execute("""
        SELECT product_name, quantity
        FROM products
        WHERE quantity <= 5
    """)

    low_stock_items = my_cursor.fetchall()

    if low_stock_items:
        print("\nFollowing products have low stock:")
```

```
        for item in low_stock_items:
            print(f"- {item[0]} (Quantity: {item[1]})")
    else:
        print("\nAll products have sufficient stock.")

    conn.close()









def show_tables():
    conn=mysql. connector .connect(host='localhost', username='root', password =
'mysql2023',database = 'drop_24')
    my_cursor = conn.cursor()
    my_cursor.execute("""Show tables""")
    conn.close()









def get_admin_id(admin_name):
    try:
        conn = mysql.connector.connect(host='localhost', username='root',
password='mysql2023', database='drop_24')
        my_cursor = conn.cursor()

        my_cursor.execute("SELECT admin_id FROM Admins WHERE admin_name = %s",
(admin_name,))
        result = my_cursor.fetchone()
```

```
        if result:
            admin_id = result[0]
            return admin_id
        else:
            print("Admin not found.")
            return None

    except mysql.connector.Error as err:
        print("Error:", err)

    finally:
        if 'conn' in locals():
            my_cursor.close()
            conn.close()









def admin_loginn(username, password):
    try:
        conn = mysql.connector.connect(host='localhost', username='root',
password='mysql2023', database='drop_24')
        my_cursor = conn.cursor()

        # verify if admin is blocked
        admin_id = get_admin_id(username)
        if admin_id:
            my_cursor.execute("SELECT locked_out, lockout_end FROM
admin_lockouts WHERE admin_id = %s", (admin_id,))
            lockout_status = my_cursor.fetchone()

            if lockout_status and lockout_status[0] == 1:
                if lockout_status[1] > datetime.now():
```

```
                    print("\nAdmin is currently blocked. Please try again
later.")
                    return False
                elif lockout_status[1] <= datetime.now():
                        my_cursor.execute("UPDATE admin_lockouts SET locked_out
= 0 WHERE admin_id = %s", (admin_id,))
                        conn.commit()
                        print("\Block period is expired. Admin is now
unblocked.")

            my_cursor.execute("SELECT * FROM Admins WHERE admin_name = %s",
(username,))
            user = my_cursor.fetchone()
            if user:
                if user[2] == password:  # Check if password matches
                    print("\nLogin successful!")
                    return True
                else:
                    my_cursor.execute("INSERT INTO login_attempts (admin_id,
attempt_time, success) VALUES (%s, %s, %s)",
                                      (admin_id, datetime.now(), False))
                    conn.commit()
                    print("\nInvalid password.")
                    return False
            else:
                print("\nInvalid username.")
                return False
        else:
            print("\nAdmin not found.")
            return False

    except mysql.connector.Error as err:
        print("Error:", err)

    finally:
        if 'conn' in locals():
            my_cursor.close()
            conn.close()
```

```
def customer_login(email, password):
   try:
       conn=mysql. connector .connect(host='localhost', username='root',
password = 'mysql2023',database = 'drop_24')
       my_cursor = conn.cursor()

       my_cursor.execute("SELECT * FROM customers WHERE email = %s AND password
= %s", (email, password))
       user = my_cursor.fetchone()

       if user:
           print("Login successful!")
           return True
       else:
           print("Invalid username or password.")
           return False

   except mysql.connector.Error as err:
       print("Error:", err)

   finally:
       if 'conn' in locals():
           my_cursor.close()
           conn.close()
```

```
def vendor_login(email, password):
   try:
       conn=mysql. connector .connect(host='localhost', username='root',
password = 'mysql2023',database = 'drop_24')
       my_cursor = conn.cursor()

       my_cursor.execute("SELECT * FROM vendors WHERE vendor_email = %s AND
password = %s", (email, password))
       user = my_cursor.fetchone()

       if user:
           print("Login successful!")
           print()
           return True
       else:
           print("Invalid username or password.")
           return False

   except mysql.connector.Error as err:
       print("Error:", err)

   finally:
       if 'conn' in locals():
           my_cursor.close()
           conn.close()









def get_customer_name(email):

   conn=mysql. connector .connect(host='localhost', username='root', password =
'mysql2023',database = 'drop_24')
```

```
    my_cursor = conn.cursor()

    query = "SELECT customer_name FROM customers WHERE email = %s"

    my_cursor.execute(query, (email,))

    result = my_cursor.fetchone()

    my_cursor.close()
    conn.close()

    if result:
        return result[0]
    else:
        return None









def get_customer_id(email):
    conn=mysql. connector .connect(host='localhost', username='root', password =
'mysql2023',database = 'drop_24')
    my_cursor = conn.cursor()

    query = "SELECT customer_id FROM customers WHERE email = %s"



    my_cursor.execute(query, (email,))

    result = my_cursor.fetchone()

    my_cursor.close()
    conn.close()
```

```
    if result:
        return result[0]
    else:
        return None









def get_vendor_id(email):
    conn=mysql. connector .connect(host='localhost', username='root', password =
'mysql2023',database = 'drop_24')
    my_cursor = conn.cursor()

    query = "SELECT vendor_id FROM vendors WHERE vendor_email = %s"


    my_cursor.execute(query, (email,))

    result = my_cursor.fetchone()

    my_cursor.close()
    conn.close()

    if result:
        return result[0]
    else:
        return None
```

```
def my_orders(customer_id):
   conn=mysql. connector .connect(host='localhost', username='root', password =
'mysql2023',database = 'drop_24')
   my_cursor = conn.cursor()

   my_cursor.execute("""
       SELECT order_id, order_date
       FROM orders
       WHERE customer_id = %s
       ORDER BY order_date
   """, (customer_id,))
   orders = my_cursor.fetchall()


   for order in orders:
       order_id, order_date = order
       print()
       print("..............................")
       print("Order Date:", order_date)

       my_cursor.execute("""
           SELECT od.product_id, p.product_name, od.quantity, p.price
           FROM order_details od
           JOIN products p ON od.product_id = p.product_id
           WHERE od.order_id = %s
       """, (order_id,))
       order_details = my_cursor.fetchall()

       total_order_value = 0


       for order_detail in order_details:
           product_id, product_name, quantity, price = order_detail
           total_order_value += quantity * price
           print(f"{product_name}  x{quantity}         -> {price}")
```

```
        print("Total order value        ->", total_order_value)
        print("..............................")
        print()
        print()


    my_cursor.close()
    conn.close()









def list_products():
    try:
        conn = mysql.connector.connect(host='localhost', username='root',
password='mysql2023', database='drop_24')
        my_cursor = conn.cursor(dictionary=True)

        my_cursor.execute("SELECT category_id, category_name FROM categories")
        categories = my_cursor.fetchall()

        for category in categories:
            category_id = category['category_id']
            category_name = category['category_name']
            print("\n" + category_name + ":")

            my_cursor.execute("""
                SELECT DISTINCT p.product_id, p.product_name, p.price
                FROM products p
                JOIN product_inventory pi ON p.product_id = pi.product_id
                WHERE p.category_id = %s
            """, (category_id,))
            products = my_cursor.fetchall()
```

```
            for product in products:
                product_id = product['product_id']
                product_name = product['product_name']
                price = product['price']
                print(f"Product ID: {product_id}, Product Name: {product_name},
Price: {price}")


        my_cursor.close()
        conn.close()

    except mysql.connector.Error as err:
        print("Error:", err)









def list_categories():
    try:
        conn = mysql.connector.connect(host='localhost', username='root',
password='mysql2023', database='drop_24')
        my_cursor = conn.cursor(dictionary=True)

        my_cursor.execute("SELECT category_id, category_name FROM categories")
        categories = my_cursor.fetchall()

        for category in categories:
            category_id = category['category_id']
            category_name = category['category_name']
            print("\n" + "ID: " + str(category_id) + "    " + category_name )

        print()
```

```
        my_cursor.close()
        conn.close()

    except mysql.connector.Error as err:
        print("Error:", err)
```

```
def get_total_sales():
    conn=mysql. connector .connect(host='localhost', username='root', password =
'mysql2023',database = 'drop_24')
    my_cursor = conn.cursor()

    my_cursor.execute("""
        SELECT SUM(p.price * od.quantity)
        FROM orders o
        JOIN order_details od ON o.order_id = od.order_id
        JOIN products p ON od.product_id = p.product_id
    """)
    total_sales = my_cursor.fetchone()[0] or 0

    my_cursor.close()
    conn.close()

    return total_sales
```

```
def get_sales_by_vendors():
   conn=mysql. connector .connect(host='localhost', username='root', password =
'mysql2023',database = 'drop_24')
   my_cursor = conn.cursor(dictionary=True)


   my_cursor.execute("""
       SELECT v.vendor_name, SUM(p.price * od.quantity) AS vendor_sales
       FROM orders o
       JOIN order_details od ON o.order_id = od.order_id
       JOIN products p ON od.product_id = p.product_id
       JOIN product_inventory pi ON p.product_id = pi.product_id
       JOIN vendors v ON pi.vendor_id = v.vendor_id
       GROUP BY v.vendor_name
   """)
   sales_by_vendors = my_cursor.fetchall()

   my_cursor.close()
   conn.close()

   return sales_by_vendors







def get_sales_by_categories():
   conn=mysql. connector .connect(host='localhost', username='root', password =
'mysql2023',database = 'drop_24')
   my_cursor = conn.cursor(dictionary=True)

   my_cursor.execute("""
       SELECT c.category_name, SUM(p.price * od.quantity) AS category_sales
       FROM orders o
```

```
        JOIN order_details od ON o.order_id = od.order_id
        JOIN products p ON od.product_id = p.product_id
        JOIN categories c ON p.category_id = c.category_id
        GROUP BY c.category_name
    """)
    sales_by_categories = my_cursor.fetchall()

    my_cursor.close()
    conn.close()

    return sales_by_categories









def remove_customer(email):
    try:
        conn=mysql. connector .connect(host='localhost', username='root',
password = 'mysql2023',database = 'drop_24')
        my_cursor = conn.cursor()

        my_cursor.execute("DELETE FROM customers WHERE email = %s", (email,))
        conn.commit()

    except mysql.connector.Error as err:
        print("Error:", err)

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

```
#updated pdated remove_vendor function
def remove_vendor(email):
   try:
       conn = mysql.connector.connect(host='localhost', username='root',
password='mysql2023', database='drop_24')
       my_cursor = conn.cursor()

       my_cursor.execute("SELECT * FROM Vendors WHERE vendor_email = %s FOR
UPDATE", (email,))
       vendor = my_cursor.fetchone()

       if vendor:
           my_cursor.execute("DELETE FROM Vendors WHERE vendor_email = %s",
(email,))
           conn.commit()
           print("Vendor removed successfully!")
       else:
           print("Vendor not found.")

   except mysql.connector.Error as err:
       print("Error:", err)
       conn.rollback()

   finally:
       if 'conn' in locals():
           my_cursor.close()
           conn.close()




def get_vendor_name(email):
   try:
```

```
        conn = mysql.connector.connect(host='localhost', username='root',
password='mysql2023', database='drop_24')
        my_cursor = conn.cursor()

        my_cursor.execute("SELECT vendor_name FROM vendors WHERE vendor_email =
%s", (email,))
        result = my_cursor.fetchone()

        if result:
            vendor_name = result[0]
            return vendor_name
        else:
            print("Vendor not found.")
            return None

    except mysql.connector.Error as err:
        print("Error:", err)

    finally:
        if 'conn' in locals():
            my_cursor.close()
            conn.close()









def add_vendor():
    while True:
        name = input("Enter vendor name: ")
        email = input("Enter vendor email: ")
        if not validate_email(email):
            print("Invalid email format. Please try again.")
```

```
            continue
        phone_number = input("Enter vendor phone number: ")
        password = input("Enter vendor password: ")

        try:
            conn=mysql. connector .connect(host='localhost', username='root',
password = 'mysql2023',database = 'drop_24')
            my_cursor = conn.cursor()

            my_cursor.execute("""
                INSERT INTO vendors (vendor_name, vendor_email, vendor_phone,
password)
                VALUES (%s, %s, %s, %s)
            """, (name, email, phone_number, password))
            conn.commit()

            print("Vendor added successfully.")

        except mysql.connector.Error as err:
            print("Error:", err)

        finally:
            my_cursor.close()
            conn.close()

        break









def add_customer():
    while True:
        name = input("Enter customer name: ")
        email = input("Enter customer email: ")
```

```
        if not validate_email(email):
            print("Invalid email format. Please try again.")
            continue
        phone_number = input("Enter customer phone number: ")
        password = input("Enter customer password: ")
        address = input("Enter Your Address")
        dob = input("Enter customer date of birth (YYYY-MM-DD): ")
        if not validate_dob(dob):
            print("Invalid date of birth format. Please try again.")
            continue
        age = calculate_age(dob)

        try:
            conn=mysql. connector .connect(host='localhost', username='root',
password = 'mysql2023',database = 'drop_24')
            my_cursor = conn.cursor()

            my_cursor.execute("""
                INSERT INTO Customers (customer_name, customer_address, email,
password, DOB, age, phone_number)
                VALUES (%s, %s, %s, %s, %s, %s, %s)
            """, (name, address, email, password, dob, age, phone_number))
            conn.commit()

            print("Customer added successfully.")

        except mysql.connector.Error as err:
            print("Error:", err)

        finally:
            my_cursor.close()
            conn.close()

        break
```

```
def add_product(product_name, price, category_id, quantity, vendor_id):
    try:
        conn = mysql.connector.connect(host='localhost', username='root',
password='mysql2023', database='drop_24')
        my_cursor = conn.cursor()

        my_cursor.execute("INSERT INTO products (product_name, price,
category_id, quantity) VALUES (%s, %s, %s, %s)",
                          (product_name, price, category_id, quantity))
        product_id = my_cursor.lastrowid

        my_cursor.execute("INSERT INTO product_inventory (product_id,
category_id, quantity, vendor_id) VALUES (%s, %s, %s, %s)",
                          (product_id, category_id, quantity, vendor_id))

        conn.commit()
        print("Product added successfully!")

    except mysql.connector.Error as err:
        print("Error:", err)

    finally:
        if 'conn' in locals():
            my_cursor.close()
            conn.close()









def list_all_product_of_vendor(vendor_id):
    try:
```

```
        conn = mysql.connector.connect(host='localhost', username='root',
password='mysql2023', database='drop_24')
        my_cursor = conn.cursor()

        my_cursor.execute("""
            SELECT p.product_id, p.product_name, p.price, pi.quantity
            FROM products p
            JOIN product_inventory pi ON p.product_id = pi.product_id
            WHERE pi.vendor_id = %s
        """, (vendor_id,))
        products = my_cursor.fetchall()

        # Print the products
        if products:
            print("Vendor Products:")
            for product in products:
                print(f"Product ID: {product[0]},    Product Name: {product[1]},
Quantity: {product[3]}")
        else:
            print("No products found for the vendor.")

        print()
        print("Enter X to exit ")

    except mysql.connector.Error as err:
        print("Error:", err)

    finally:
        if 'conn' in locals():
            my_cursor.close()
            conn.close()




# Updated function
def remove_product_by_vendor(vendor_id, product_id, amount):
```

```
    try:
        conn = mysql.connector.connect(host='localhost', username='root',
password='mysql2023', database='drop_24')
        my_cursor = conn.cursor()

        my_cursor.execute("SELECT * FROM Products WHERE product_id = %s FOR
UPDATE", (product_id,))
        product = my_cursor.fetchone()

        if product:
            if product[6] == vendor_id:
                my_cursor.execute("UPDATE Products SET quantity = quantity - %s
WHERE product_id = %s", (amount, product_id))

                my_cursor.execute("UPDATE Product_Inventory SET quantity =
quantity - %s WHERE product_id = %s AND vendor_id = %s", (amount, product_id,
vendor_id))

                conn.commit()
                print("Product removed successfully!")
            else:
                print("The specified product does not belong to the vendor.")
        else:
            print("Product not found.")

    except mysql.connector.Error as err:
        print("Error:", err)
        conn.rollback()

    finally:
        if 'conn' in locals():
            my_cursor.close()
            conn.close()
```

```
#Updated restock function
def restock_product(vendor_id, product_id, amount):
    try:
        conn = mysql.connector.connect(host='localhost', username='root',
password='mysql2023', database='drop_24')
        my_cursor = conn.cursor()

        my_cursor.execute("SELECT * FROM Products WHERE product_id = %s FOR
UPDATE", (product_id,))
        product = my_cursor.fetchone()

        if product:
            my_cursor.execute("SELECT * FROM Product_Inventory WHERE product_id
= %s AND vendor_id = %s FOR UPDATE", (product_id, vendor_id))
            inventory = my_cursor.fetchone()

            if inventory:
                my_cursor.execute("UPDATE Products SET quantity = quantity + %s
WHERE product_id = %s", (amount, product_id))

                my_cursor.execute("UPDATE Product_Inventory SET quantity =
quantity + %s WHERE product_id = %s AND vendor_id = %s", (amount, product_id,
vendor_id))

                conn.commit()
                print("Product restocked successfully!")
            else:
                print("The specified product does not belong to the vendor.")
        else:
            print("Product not found.")

    except mysql.connector.Error as err:
        print("Error:", err)
        conn.rollback()

    finally:
        if 'conn' in locals():
```

```
            my_cursor.close()
            conn.close()









def vendor_notification(vendor_id):
    try:
        conn = mysql.connector.connect(host='localhost', username='root',
password='mysql2023', database='drop_24')
        my_cursor = conn.cursor()

        my_cursor.execute("SELECT message, created_at FROM notifications WHERE
vendor_id = %s", (vendor_id,))
        messages = my_cursor.fetchall()
        conn.commit()

        if messages:
            print("Notifications: ")
            for product in messages:
                print(f"Message: {product[0]}     Time: {product[1]}")
        else:
            print("No New Messages for you")
        print()

    except mysql.connector.Error as err:
        print("Error:", err)

    finally:
        if 'conn' in locals():
            my_cursor.close()
            conn.close()
```

```
def validate_email(email):
   return "@" in email and "." in email

def validate_dob(dob):
   try:
       datetime.strptime(dob, "%Y-%m-%d")
       return True
   except ValueError:
       return False

def calculate_age(dob):
   dob_date = datetime.strptime(dob, "%Y-%m-%d")
   today = datetime.today()
   age = today.year - dob_date.year - ((today.month, today.day) <
(dob_date.month, dob_date.day))
   return age
```

```
def main():
   while True:
       print("\nRetail Store Management System")
       print("1. Admin Login")
       print("2. Customer Login")
       print("3. Vendor Login")
       print("4. Exit")


       login_choice = input("Enter your choice: ")

       if login_choice == '1':
           username = input("Enter your username: ")
           password = input("Enter your password: ")
           if admin_loginn(username, password):
               while True:
                   print("\nRetail Store Management System")
                   print("1. Add / Remove customer")
                   print("2. Add / Remove vendor")
                   print("3. Check Low Stock Products")
                   print("4. Total Sales")
                   print("5. Total Sales by vendor")
                   print("6. Total Sales by category")
                   print("7. Logout")

                   admin_choice = input("Enter your choice: ")

                   if admin_choice == '1':
                       print("1. Add customer")
                       print("2. Remove customer")

                       adre_choice = input("Enter Your choice: ")

                       if adre_choice == '1':
                           add_customer()
                       elif adre_choice == '2':
```

```
                        cus_email = input("Enter customer email to remove:
")
                        print()
                        cus_name = get_customer_name(cus_email)
                        print("are you sure you want to remove ", cus_name,
" from database?")
                        print("y/n")
                        if input()=='y':
                            print(cus_name, " removed succesfully from
database")
                            remove_customer(cus_email)
                        print()

                elif admin_choice == '2':
                    print("1. Add Vendor")
                    print("2. Remove Vendor")

                    adre_choice = input("Enter Your choice: ")

                    if adre_choice == '1':
                        add_vendor()
                    elif adre_choice == '2':
                        vend_email = input("Enter customer email to remove:
")
                        print()
                        vend_name = get_vendor_name(vend_email)
                        print("are you sure you want to remove ", vend_name,
" from database?")
                        print("y/n")
                        if input()=='y':
                            print(vend_name, " removed succesfully from
database")
                            remove_vendor(vend_email)
                        print()

                elif admin_choice == '3':
                    low_stock_products()
```

```
                elif admin_choice == '4':
                    total_sales = get_total_sales()
                    print()
                    print("Total Sales ->", total_sales)
                    print()

                elif admin_choice == '5':
                    sales_by_vendors = get_sales_by_vendors()
                    print()
                    print("\nSales by Vendors:")
                    for record in sales_by_vendors:
                        print(record['vendor_name'], "->",
record['vendor_sales'])
                    print()

                elif admin_choice == '6':
                    sales_by_categories = get_sales_by_categories()
                    print()
                    print("\nSales by Categories:")
                    for record in sales_by_categories:
                        print(record['category_name'], "->",
record['category_sales'])
                    print()

                elif admin_choice == '7':
                    print("Logging out...")
                    print(".")
                    print(".")
                    print(".")
                    print(".")
                    print(".")
                    print(".")
                    print(".")
                    print("Logged out")
                    break


        elif login_choice == '2':
```

```
        email = input("Enter your Email: ")
        password = input("Enter your password: ")
        if customer_login(email, password):
            username = get_customer_name(email)
            customer_id = get_customer_id(email)
            while True:
                print("\nWelcome ", username, "!")
                print("1. Place Order")
                print("2. My Orders")
                print("3. Logout")

                customer_choice = input("Enter Your Choice: ")

                if customer_choice == '1':
                    list_products()
                    product_id = int(input("Enter product ID: "))
                    quantity = int(input("Enter quantity: "))
                    place_order(customer_id, product_id, quantity)
                elif customer_choice == '2':
                    my_orders(customer_id)
                elif customer_choice == '3':
                    print("Logging out...")
                    print(".")
                    print(".")
                    print(".")
                    print(".")
                    print(".")
                    print(".")
                    print(".")
                    print("Logged out")
                    break
                else:
                    print("Invalid choice. Please try again.\n")

    elif login_choice == '3':
        email = input("Enter Your Email: ")
        password = input("Enter Your Password: ")
        if vendor_login(email, password):
```

```
                ven_id = get_vendor_id(email)
                vendor_notification(ven_id)
                while True:
                    print("1. Add Product")
                    print("2. Remove Product")
                    print("3. restock")
                    print("4. Log Out")

                    vend_choice = input("Enter Your Choice: ")

                    if vend_choice == '1':
                        name = input("Enter Product's name: ")
                        price = input("Enter Price of product: ")
                        list_categories()
                        category = input("Sele which category it falls into: ")
                        quantityy = input("Enter the quantity: ")
                        add_product(name, price, category, quantityy, ven_id)
                    elif vend_choice == '2':
                        list_all_product_of_vendor(ven_id)
                        print()
                        remove_pro = input("Enter ID of product to remove: ")
                        if (remove_pro == "X") or (remove_pro == "x"):
                            print()
                        else:
                            hello = input("Enter the quantity you want to
delete: ")
                            remove_product_by_vendor(ven_id, remove_pro, hello)
                    elif vend_choice == '3':
                        list_all_product_of_vendor(ven_id)
                        print()
                        restock = input("Enter ID of product to restock: ")
                        if (restock == "X") or (restock == "x"):
                            print()
                        else:
                            amount = input("Enter the number of items you want
to add: ")
                            restock_product(ven_id, restock, amount)
                    elif vend_choice == '4':
```

```
                    print("Logging out...")
                    print(".")
                    print(".")
                    print(".")
                    print(".")
                    print(".")
                    print(".")
                    print(".")
                    print("Logged out")
                    break


        elif login_choice == '4':
            print("Exiting...")
            return
        else:
            print("Invalid choice. Please try again.\n")

if __name__ == "__main__":
    main()
```