Transactions In Laundramat2 Services

1. Non Conflicting Transactions
    a. Place Order and Make Payment

```
# Transaction 1: Place Order and Make Payment
def place_order_and_make_payment(customer_id, service_type,
amount):
    try:
        conn = create_connection()
        cursor = conn.cursor()

        cursor.execute("INSERT INTO orders (Amount, Type,
CustomerID) VALUES (%s, %s, %s)",
                       (amount, service_type, customer_id))

        cursor.execute("INSERT INTO payments (Amount, Date,
CustomerID) VALUES (%s, CURDATE(), %s)",
                       (amount, customer_id))

        conn.commit()
        print("Order placed and payment made successfully.")

    except mysql.connector.Error as err:
        print(f"Error: {err}")
        conn.rollback()

    finally:
        if conn.is_connected():
            conn.close()
```

    b. View Available Services and Discounts

```
# Transaction 2: View Available Services and Discounts
def view_services_and_discounts():
    try:
        conn = create_connection()
        cursor = conn.cursor()

        cursor.execute("SELECT * FROM profile")
        services = cursor.fetchall()
```

```
        cursor.execute("SELECT * FROM discount WHERE ValidFrom
<= CURDATE() AND ValidTo >= CURDATE()")
        discounts = cursor.fetchall()

        conn.commit()
        print("Available Services:")
        for service in services:
            print(service)
        print("\nActive Discounts:")
        for discount in discounts:
            print(discount)

    except mysql.connector.Error as err:
        print(f"Error: {err}")
        conn.rollback()

    finally:
        if conn.is_connected():
            conn.close()
```

c. Add Feedback for Completed Order

```
# Transaction 3: Add Feedback for Completed Order
def add_feedback(order_id, rating, review):
    try:
        conn = create_connection()
        cursor = conn.cursor()

        cursor.execute("INSERT INTO feedback (OrderID, Rating,
Review) VALUES (%s, %s, %s)",
                       (order_id, rating, review))

        conn.commit()
        print("Feedback added successfully.")

    except mysql.connector.Error as err:
        print(f"Error: {err}")
        conn.rollback()

    finally:
        if conn.is_connected():
```

```
            conn.close()
```

d. Send Notification for Low Inventory

```
# Transaction 4: Send Notification for Low Inventory
def send_notification_for_low_inventory(service_type):
    try:
        conn = create_connection()
        cursor = conn.cursor()

        cursor.execute("SELECT COUNT(*) FROM orders WHERE Type
= %s", (service_type,))
        order_count = cursor.fetchone()[0]

        if order_count <= 5:
            cursor.execute("INSERT INTO notification (UserID,
Message, SentDateTime) VALUES (%s, %s, NOW())",
                           (1, f"Low inventory for service:
{service_type}"))

        conn.commit()
        print("Notification sent for low inventory.")

    except mysql.connector.Error as err:
        print(f"Error: {err}")
        conn.rollback()

    finally:
        if conn.is_connected():
            conn.close()
```

2. Conflicting Transactions
    a. Place Order and Deduct Inventory

```
# Transaction 1: Place Order and Deduct Inventory
def place_order_and_deduct_inventory(customer_id,
service_type, amount):
    try:
        conn = create_connection()
        cursor = conn.cursor()
```

```
        cursor.execute("INSERT INTO orders (Amount, Type,
CustomerID) VALUES (%s, %s, %s)",
                       (amount, service_type, customer_id))

        cursor.execute("UPDATE services SET InventoryCount =
InventoryCount - 1 WHERE Type = %s",
                       (service_type,))

        conn.commit()
        print("Order placed and inventory deducted
successfully.")

    except mysql.connector.Error as err:
        print(f"Error: {err}")
        conn.rollback()

    finally:
        if conn.is_connected():
            conn.close()
```

b. View Available Services and Inventory

```
def view_services_and_inventory():
    try:
        conn = create_connection()
        cursor = conn.cursor()

        cursor.execute("SELECT * FROM profile")
        services = cursor.fetchall()

        cursor.execute("SELECT Type, InventoryCount FROM
services")
        inventory = cursor.fetchall()
        conn.commit()

        print("Available Services:")
        for service in services:
            print(service)
        print("\nInventory Count:")
        for item in inventory:
            print(item)
```

```
except mysql.connector.Error as err:
    print(f"Error: {err}")
    conn.rollback()

finally:
    if conn.is_connected():
        conn.close()
```