Transactions

T1: This transaction is for checking coupon discount from the entered coupon code and then applying this to the cart total amount.

```
START TRANSACTION;
SELECT C.discount
FROM coupon C
WHERE coupon_code = 'uY3*+V!5H';

UPDATE cart C
SET C.total_amount1 = C.total_amount1*(100-discount)/100
WHERE C.cart_id = 1;
COMMIT;
```

T2: While checking-out a cart, for each cart-item A, we are reducing the availability of the product(i.e stock) by the number of items that has been bought by that customer while checking out.

```
START TRANSACTION;
SELECT ci.product_id1, ci.quantity
INTO @productID, @quantity
FROM cart_items ci
JOIN cart c ON ci.cart_id1 = c.cart_id
WHERE c.cart_id = 5;

UPDATE products AS P
SET P.stock = P.stock - @quantity
WHERE P.product_id = @productID;
COMMIT;
```

T3: This transaction is for someone who is viewing that product and after finding it available, he adds that product into his cart.

```
START TRANSACTION;
SELECT ca.cart_id INTO @current_cart_id
FROM cart ca
JOIN customer cu ON ca.customer_id3 = cu.customer_id
WHERE cu.customer_id = 3;

INSERT INTO cart_items (cart_id1, product_id1, quantity)
VALUES (@current_cart_id, 5, 1);
COMMIT;
```

T4: This transaction is for admin who adds products to stock (i.e., increases the quantity)

```
START TRANSACTION;
UPDATE products
SET stock = stock + 10
WHERE product_id = 2;
COMMIT;
```

T5: In this transaction in which a customer buys the same product (i.e. decreases the quantity)

```
START TRANSACTION;
UPDATE products
SET stock = CASE
        WHEN (SELECT stock FROM products WHERE product_id = 2) >= 5
        THEN stock - 5
        ELSE stock
      END
WHERE product_id = 2;
COMMIT;
```

T6: In the first transaction, the customer buys the product. Reads The products price and inserts the order item

```
START TRANSACTION;
SELECT price INTO @product_price
FROM products
WHERE product_id = 4;

INSERT INTO order_item (quantity, price, order_id1, product_id1)
   VALUES (1, @product_price, 27, 4);
COMMIT;
```

T7: In this transaction, the product price is increased by the specified amount.

```
START TRANSACTION;

UPDATE products
SET price = price + 100
WHERE product_id = 4;

COMMIT;
```

All above transactions in read write format.

Transaction1
READ(discount) from coupon **WRITE(total_amount) in cart** **COMMIT**

Transaction2	Transaction3
READ(quantity) from cart_items **WRITE(quantity) to products** **COMMIT**	**READ(quantity) from products** **WRITE(quantity) to cart_items** **COMMIT**

Transaction4	Transaction5
READ(quantity) from products Q = Q + q1 **WRITE(quantity) from products** **COMMIT**	**READ(quantity) from cart_items** Q = Q - q2 **WRITE(quantity) to products** **WRITE(Order)** **COMMIT**

Transaction6	Transaction7
READ(product_price) from products **WRITE(product_price) to orders** **COMMIT**	**READ(product_price) from products** **WRITE(product_price) to products** **COMMIT**

CONFLICT SERIALIZABLE SCHEDULE:

T2	T3	T1
READ(quantity) from cart_items		
		READ(discount)
WRITE(quantity) to products		
COMMIT		
	READ(quantity) from products	
	WRITE(quantity) to cart_items	
	COMMIT	
		WRITE(total_amount)
		COMMIT

NON-CONFLICT SERIALIZABLE SCHEDULE:

T1	T2	T3
READ(discount)		
	READ(quantity) from cart_items	
		READ(quantity) from products
	WRITE(quantity) to products	
WRITE(total_amount)		

COMMIT		
		WRITE(quantity) to cart_item
		COMMIT
	COMMIT	

Read-Write and Write-Write conflict in the above scedule.

CONFLICT SERIALIZABLE SCHEDULE:

T4	T5
	READ(quantity) from cart_items
	WRITE(quantity)
READ(quantity) from products	
WRITE(quantity) from products	
	WRITE(Order)
	COMMIT
COMMIT	

The above schedule is a conflict serializable scedule as, in this graph(T4, T5) **no loop** is formed.

NON-CONFLICT SERIALIZABLE SCHEDULE:

T4	T5
	READ(quantity) from cart_items
READ(quantity) from products	
WRITE(quantity) from products	
	WRITE(quantity) in products

	WRITE(Order)
	COMMIT
COMMIT	

CONFLICT SERIALIZABLE SCHEDULE:

T6	T7
READ(product_price) from products	
	READ(product_price) from products
	WRITE(product_price) to products
WRITE(product_price) to orders	
COMMIT	
	COMMIT

NON-CONFLICT SERIALIZABLE SCHEDULE:

T6	T7
	READ(product_price) from products
READ(product_price) from products	
	WRITE(product_price) to products
WRITE(product_price) to orders	
COMMIT	
	COMMIT