NON CONFLICTING TRANSACTIONS

1. Transaction Regarding viewing all Products and Adding them to cart

- T1: Transaction to view all Products that are present
- T2: Transaction to add products to the cart

T1	T2
R(PRODUCT)	
COMMIT	
	R(PRODUCT)
	R(CART)
	W(CART)
	COMMIT

Explanation: T1 and T2 are both reading from the products table , hence, there is No conflict (RR). This is a serialisable schedule

2. Transaction Regarding viewing Products by category and Searching product by Keywords

- T3: Transaction to view Products category wise
- T4: Transaction to search products by keyword

T3	T4
R(PRODUCT)	
COMMIT	
	R(PRODUCT)
	COMMIT

Explanation: T3 and T4 are both reading from the products table , hence, there is No conflict (RR). This is a serialisable schedule

CONFLICTING TRANSACTION

1. Conflicting scheduling between delete Admin Transaction and Updating Admin details Transaction

- T1: Transaction to delete Admin
- T2: Transaction to update the details of Admin

T1	T2
R(ADMIN)	
	R(ADMIN)
	W(ADMIN)
W(ADMIN)	
	COMMIT
COMMIT	

Explanation: The above-mentioned schedule is conflicting since it has potential to cause a conflict . There can be a scenario that an admin tries to update the address of an admin and simultaneously another user deletes the admin which the other updated, this can cause a conflict in the database. Thus, we handled this with a lock.

Additional Transactions: (Optional)

2. Conflicting schedule Place Order and Log In

- T3: Transaction to view products
- T4: Transaction to update the products

T3	T4
R(CART)	
R(PRODUCT)	
W (CUSTOMER)	
	R (CUSTOMER)
	COMMIT
COMMIT	

Explanation: T3 is writing to the Customer table and T4 is reading from the customer table, leading to a conflict. Thus we handled this with a lock.