We have covered implemented 8+ transactions, as defined in the file crud.py.

Let T1 and T2 be the two transactions defined in the functions **complete_ride()** and **view_past_bookings_by_driver()**, respectively.

As both transactions act on the same object (i.e. *Booking*), we can define T1 and T2 as follows:

T1 := [R(A), W(A), W(A)]
T2 := [R(A)]

—
Consider the following schedule, S1:

T1	**T2**
R(A)	
W(A)	
	R(A)
W(A)	

The precedence graph for this schedule would be a **cyclic graph** with directed edges from T1 to T2 as well as T2 to T1, as the pairs of instructions (2, 3) and (3, 4) are conflicting.

Hence, the above schedule S1 is **non-conflict serialisable**.

—
Consider the following schedule, S1:

T1	**T2**
R(A)	
	R(A)
W(A)	
W(A)	

The precedence graph for this schedule would be **acyclic**, with only one edge from T2 to T1, as the pair of instructions (2, 3) forms an RW conflict. However, the pair of instructions (1, 2) is non-conflicting, and upon swapping them, we get the serial schedule **S := T1 -> T2.**

Hence, the above schedule S2 is **conflict serialisable**.