Transaction 1: Update Ride Dropoff Time

1. Begin transaction
2. Retrieve the ride with Ride_ID = 1234 and check if its Dropoff_Time is less than the current time.
3. If the Dropoff_Time is less than the current time, update the Dropoff_Time to the current time.
4. Commit transaction

BEGIN TRANSACTION;

UPDATE driver SET Status = 'Busy' WHERE Driver_ID = '10001';
UPDATE ride SET Driver_ID = '10001' WHERE Ride_ID = '10123';

COMMIT;

Transaction 2: Update Cab Availability

1. Begin transaction
2. Retrieve the driver with Driver_ID = 5678 and check if their cab is currently available (i.e. the Status is "Available").
3. If the cab is available, update the Status to "Busy".
4. Commit transaction

BEGIN TRANSACTION;

UPDATE cab SET Available = 'No' WHERE Cab_ID = 3580;
INSERT INTO ride (Ride_ID, Dropoff_Time, Pickup_Location, Dropoff_Location, Request_ID, Driver_ID, Cab_ID)
VALUES ('10124', '2023-04-25 12:00:00', 'Central Park', 'Empire State Building', '10694', '10002', ‘3580’);

COMMIT;

Conflict Serializable Schedule:

Assume the following order of execution:

- T1 begins and retrieves Ride with Ride_ID = 1234

- T2 begins and retrieves Driver with Driver_ID = 5678
- T2 updates Driver's Status to "Busy"
- T2 commits
- T1 checks Ride's Dropoff_Time and updates it to the current time
- T1 commits

T1: BEGIN TRANSACTION;
T1: UPDATE driver SET Status = 'Busy' WHERE Driver_ID = '10001';
T2: BEGIN TRANSACTION;
T2: UPDATE cab SET Available = 'No' WHERE Cab_ID = 3580;
T1: UPDATE ride SET Driver_ID = '10001' WHERE Ride_ID = '10123';
T2: INSERT INTO ride (Ride_ID, Dropoff_Time, Pickup_Location, Dropoff_Location, Request_ID, Driver_ID, Cab_ID) VALUES ('10124', '2023-04-25 12:00:00', 'Central Park', 'Empire State Building', '10694', '10002', ‘3580’);
T1: COMMIT;
T2: COMMIT;

Non-Conflict Serializable Schedule:

Assume the following order of execution:

- T1 begins and retrieves Ride with Ride_ID = 1234
- T2 begins and retrieves Driver with Driver_ID = 5678
- T1 checks Ride's Dropoff_Time and updates it to the current time
- T1 commits
- T2 updates Driver's Status to "Busy"
- T2 commits

In this case, the order of execution leads to a non-conflict serializable schedule because T1 and T2 do not conflict with each other, and the order of execution does not affect the final result.

T1: BEGIN TRANSACTION;
T1: UPDATE driver SET Status = 'Busy' WHERE Driver_ID = '10001';
T1: UPDATE ride SET Driver_ID = '10001' WHERE Ride_ID = '10123';
T1: COMMIT;
T2: BEGIN TRANSACTION;
T2: UPDATE cab SET Available = 'No' WHERE Cab_ID = 3580;
T2: INSERT INTO ride (Ride_ID, Dropoff_Time, Pickup_Location, Dropoff_Location, Request_ID, Driver_ID, Cab_ID) VALUES ('10124', '2023-04-25 12:00:00', 'Central Park', 'Empire State Building', '10694', '10002', ‘3580’);
T2: COMMIT;