Arjun Moitra 2021313 | Aryan Malik 2021315

Transaction 1: A customer books a trip with a driver

- Insert a new trip record into the Trip table with the Driver_Id and Customer_Id, Journey_Begins, Journey_Ends, Distance, and Total_Amount values.
- Update the Driver table to reflect the new trip and raise the driver's rating.
- Update the Customer database to reflect the new driver rating the client provided.

Transaction 2: A user updates their address and mobile number

- Update the User table for the user with the specified User_Id to reflect the new Address and Mobile_No values.

Schedule 1 (Conflict Serializable):

T1: Add a fresh trip record to the trips table.

T2: Update address and mobile number for User

Step	Transaction	Read	Write
1	T1	-	Insert into Trip table
2	T2	Read User table	Update User table

- Since T1 and T2 do not read or write the same data, this schedule is conflict serializable and can be executed in any order without causing any conflicts.

Schedule 2 (Non-Conflict Serializable):

T1: Add a fresh trip record to the trips table.

T2: Update the Driver table's Driver rating.

T3: Update the Customer table's customer rating.

Step	Transaction	Read	Write
1	T1	-	Insert into Trip table
2	T2	Read Trip table	Update Driver table (based on Trip data)
3	T3	Read Trip and Customer tables	Update Customer table (based on Trip data)

- This schedule is non-conflict serializable since T2 and T3 are dependent on each other in a cycle, they read the data that was inserted by T1 and update data that impacts the other transaction. The Driver table's rating for the driver must be updated when T2 reads the Driver_Id from the Trip table, which was inserted by T1. The Customer_Id from the Trip table, which was also added by T1, must be read by T3, who will then change the customer's rating in the Customer table. T2 and T3 cannot be executed in a conflict serializable way since they are mutually dependent.