This schedule is conflict serializable because it can be transformed into two equivalent schedules:

Schedule 1:
START TRANSACTION;
UPDATE Customer SET Rating = Rating + 1 WHERE CustomerID = 1;
UPDATE Driver SET TotalEarning = TotalEarning + 500 WHERE DriverID = 2;
COMMIT;

Schedule 2:

START TRANSACTION;
UPDATE Customer SET Rating = Rating - 1 WHERE CustomerID = 1;
UPDATE Driver SET TotalEarning = TotalEarning - 500 WHERE DriverID = 2;
COMMIT;

Both schedules have the same effect as the original schedule, and they are executed in a conflict serializable manner because they have the same order of transactions.

To make the given schedule non-conflict serializable, we can modify it as follows:

START TRANSACTION T1;
UPDATE Customer SET Rating = Rating + 1 WHERE CustomerID = 1;
COMMIT;

START TRANSACTION T2;
UPDATE Customer SET Rating = Rating - 1 WHERE CustomerID = 1;
COMMIT;

START TRANSACTION T3;
UPDATE Driver SET TotalEarning = TotalEarning + 500 WHERE DriverID = 2;
UPDATE Driver SET TotalEarning = TotalEarning - 500 WHERE DriverID = 2;
COMMIT;

In this modified schedule, the transactions T1 and T2 are executed independently, without any conflict. Similarly, transaction T3 can be executed independently after T1 and T2 are completed. This schedule is non-conflict serializable because there are no conflicting transactions that need to be executed in a specific order.