```
Following is the transaction 1 query
START TRANSACTION;
SET @wallet id = NULL;
SET @balance = NULL;
SET @wallet id = (SELECT Wallet id FROM rider WHERE Rider Id = '199');
SET @request_id = ((SELECT MAX(Request_ID) FROM request) + 1);
INSERT INTO request (Request ID, Rider Id, preference, Preffered Vehicle,
Pickup Location x, Pickup Location y, Drop Location x, Drop Location y, Status) VALUES
(@request id, '199', 1, 'sedan', 0, 0, 1, 1, 1);
SET @driver id = (SELECT d.driver id FROM driver d JOIN vehicle v ON d.Vehicle id =
v.vehicle id WHERE d.Status = 1 AND v.type = 'sedan' ORDER BY d.Current Rating DESC
LIMIT 1);
SET @p x = (SELECT Pickup Location x FROM request WHERE Request ID =
@request id);
SET @p y = (SELECT Pickup Location y FROM request WHERE Request ID =
@request_id);
SET @d x = (SELECT Drop Location x FROM request WHERE Request ID = @request id);
SET @d y = (SELECT Drop Location y FROM request WHERE Request ID = @request id);
SET @distance = SQRT(POW(@p x - @d x, 2) + POW(@p y - @d y, 2));
SET @rate = (SELECT rate FROM driver WHERE driver_id = @driver_id);
SET @fare = @distance * @rate;
INSERT INTO trip (Request_ID, Driver_id, Fare, Date_Time, Distance, Rating) VALUES
(@request id, @driver id, @fare, NOW(), @distance, 4);
SET @balance = (SELECT amount FROM wallet WHERE wallet_id = @wallet_id);
IF (@fare <= @balance) THEN
  INSERT INTO billing (Request ID, Driver ID, Rider ID, Date TIME, Distance, Fare,
Pickup Location x, Pickup Location y, Drop Location x, Drop Location y) VALUES
(@request id, @driver id, '199', NOW(), @distance, @fare, @p x, @p y, @d x, @d y);
  SET @balance = @balance - @fare;
  UPDATE wallet SET amount = @balance WHERE wallet id = @wallet id;
  COMMIT:
ELSE
  ROLLBACK:
END IF;
```

Following is the transaction 2 query:

START TRANSACTION;

UPDATE wallet SET amount = amount + 1000 WHERE Wallet\_id = @wallet\_id; COMMIT;

Subtract fare from wallet balance loosers  Subtract fare from wallet balance above  Supdate wallet balance commit;  Here a condition of hose and this is therefore a condition of hose and this is therefore a condition of the could either put a affect before reading a	Je Wholate (wallet Balonu of Rider)  Commit;  Sy Update may arise conflicting pairs of transactions  non-conflict serializable transaction  Shared lock on the wallet of the rider balonce or we could perform  to balonce (in 72) before reading in Ti.
We could either put a	shored lock on the wallet of the rider vallet balonce or we could perform
ie either $T_1 \longrightarrow T_2$ or $T_2 \longrightarrow T_1$	

To make it serializable we can either put a share lock on the wallet in transaction 1 which is done in the following code

```
START TRANSACTION:
SELECT Wallet id FROM rider WHERE Rider Id = '199' LOCK IN SHARE MODE;
SET @wallet id = LAST INSERT ID();
SELECT MAX(Request ID) FROM request;;
SET @request id = LAST INSERT ID() + 1;
INSERT INTO request (Request ID, Rider Id, preference, Preffered Vehicle,
Pickup Location x, Pickup Location y, Drop Location x, Drop Location y, Status) VALUES
(@request_id, '199', 1, 'sedan', 0, 0, 1, 1, 1);
SELECT d.driver id FROM driver d JOIN vehicle v ON d.Vehicle id = v.vehicle id WHERE
d.Status = 1 AND v.type = 'sedan' ORDER BY d.Current_Rating DESC LIMIT 1 LOCK IN
SHARE MODE:
SET @driver id = LAST INSERT ID();
SELECT Pickup_Location_x, Pickup_Location_y, Drop_Location_x, Drop_Location_y FROM
request WHERE Request ID = @request id LOCK IN SHARE MODE;
SET @p_x = LAST_INSERT_ID();
SET @p y = LAST INSERT ID();
SET @d x = LAST INSERT ID();
SET @d_y = LAST_INSERT_ID();
SET @distance = SQRT(POW(@p x - @d x, 2) + POW(@p y - @d y, 2));
SELECT rate FROM driver WHERE driver id = @driver id LOCK IN SHARE MODE;
SET @rate = LAST_INSERT_ID();
SET @fare = @distance * @rate;
INSERT INTO trip (Request_ID, Driver_id, Fare, Date_Time, Distance, Rating) VALUES
(@request id, @driver id, @fare, NOW(), @distance, 4) LOCK IN EXCLUSIVE MODE;
SELECT amount FROM wallet WHERE wallet id = @wallet id FOR UPDATE;
SET @balance = LAST_INSERT_ID();
IF (@fare <= @balance) THEN
  INSERT INTO billing (Request ID, Driver ID, Rider ID, Date TIME, Distance, Fare,
Pickup Location x, Pickup Location y, Drop Location x, Drop Location y) VALUES
(@request_id, @driver_id, '199', NOW(), @distance, @fare, @p_x, @p_y, @d_x, @d_y)LOCK
IN EXCLUSIVE MODE;;
  SET @balance = @balance - @fare;
  UPDATE wallet SET amount = @balance WHERE wallet id = @wallet id;
  COMMIT:
ELSE
  ROLLBACK;
END IF;
```

Or we could simply run the transaction 2 first and then the transaction 1 in that way there would be no condition of lossy update.