Conflicting and Non-Conflicting Transactions

Non-Conflicting Transactions:

1) Books a ride and updates the driver status for the driver:

   ```
   START TRANSACTION;
   INSERT INTO Bookings (Booking_ID, UserID, Dri_ID, Pickup, Dropoff)
   VALUES (111, 15, 5, '8D Isadore Esplanade\nSouth Joel, NSW 7154', '87 /
   21 Adrien Pathway\nNew Rahulfurt, WA 6956');
   UPDATE Driver SET Dr_Status = 'inactive' WHERE Driver_ID = 5;
   COMMIT;
   ```

2) A booking is made, and after the completion of the ride, the corresponding payment is calculated based on the distance traveled, and payment is made:

   ```
   START TRANSACTION;
   INSERT INTO Bookings (Booking_ID, UserID, Dri_ID, Pickup, Dropoff)
   VALUES (112, 20, 20, '7D Isadore Esplanade\nSouth Joel, NSW 7154', '87
   / 21 Adrien Pathway\nNew Rahulfurt, WA 69');
   INSERT INTO DrivesFor (DrivID, UsrID, Tim, Distance) VALUES (20, 20,
   '2022-11-17 13:34:29', '50');
   INSERT INTO Payment (PaymentID, BookieID, Payment_Amount,
   Payment_Method, Payment_Status) VALUES (112, 112, 500, 'Debit Card',
   'Paid');
   COMMIT;
   ```

3) Updates the user information, and the user further makes a booking using the updated info.:

   ```
   START TRANSACTION;
   UPDATE Users set
   fname='Mike',lname='Beier',UserPassword='c5881ebc049223408da0bb0
   4cc44c3390f4c5777' WHERE User_ID = 1;
   ```

```
INSERT INTO Bookings (Booking_ID, UserID, Dri_ID, Pickup, Dropoff)
VALUES (113,1,9,'7D Esplanade\nSouth Joel, NSW 7154', '87 / 21
Pathway\nNew Rahulfurt, WA 69');
Commit;
```

4) Transfers a particular booking from one driver to another:

```
START TRANSACTION;
UPDATE Bookings SET Dri_ID = 13 WHERE Booking_ID = 4;
UPDATE Driver SET Dr_Status = 'inactive' WHERE Driver_ID = 13;
UPDATE Driver SET Dr_Status = 'active' WHERE Driver_ID = 104;
Commit;
```

5) Creates a new vehicle and assigns it to the driver and removes the old vehicle:

```
START TRANSACTION;
Delete FROM Vehicle where Vehicle_ID=1;
INSERT INTO Vehicle (Vehicle_ID, DriveeID, License_Platenumber, Model,
Make) VALUES (101, 1, '635588420', '#e999', 'Ratke and Kozey');
Commit;
```

6) A new User is added to the database:

```
START TRANSACTION;
INSERT INTO Users (User_ID,fname, mname, lname, User_Status, E_mail,
UserPassword) VALUES (101,'John', '', 'Doe', 'Normal',
'john.doe@example.com', 'password123');
Commit;
```

Conflicting Transactions:

1)

```
  START TRANSACTION;

  UPDATE Bookings SET Dri_ID = 13 WHERE Booking_ID = 4;

  COMMIT;


  START TRANSACTION;

  UPDATE Bookings SET Dri_ID = 15 WHERE Booking_ID = 4;

  COMMIT;
```

These two transactions are conflicting because they are both trying to modify the same row in the Bookings table, specifically the row with Booking_ID of 4. Transaction 1 is setting the Dri_ID column to 13, while Transaction 2 is setting it to 15. If these transactions were executed concurrently, it's possible that one of the transactions would overwrite the changes made by the other transaction, resulting in inconsistent data.

2)

```
  START TRANSACTION;

  INSERT INTO Bookings (Booking_ID, UserID, Dri_ID, Pickup, Dropoff) VALUES
(123, 15, 5, '8D Isadore Esplanade\nSouth Joel, NSW 7154', '87 / 21 Adrien
Pathway\nNew Rahulfurt, WA 6956');

  UPDATE Driver SET Dr_Status = 'inactive' WHERE Driver_ID = 5;

  COMMIT;



  START TRANSACTION;

  UPDATE Bookings SET Dri_ID = 13 WHERE Booking_ID = 123;
  UPDATE Driver SET Dr_Status = 'inactive' WHERE Driver_ID = 13;
  COMMIT;
```

Transaction 1 is trying to insert a new record into the Bookings table with Booking_ID = 123 and Dri_ID = 5, while Transaction 2 is trying to update the same record with Dri_ID = 13. This creates a conflict because only one of these transactions can successfully update the record, while the other transaction will fail due to a constraint violation.

3)

```
   START TRANSACTION;
   UPDATE Drivers SET Availability = 'Unavailable' WHERE Driver_ID = 10;
   UPDATE Bookings SET Driver_ID = 10 WHERE Booking_ID = 8;
   COMMIT;

   START TRANSACTION;
   UPDATE Drivers SET Availability = 'Unavailable' WHERE Driver_ID = 10;
   UPDATE Bookings SET Driver_ID = 10 WHERE Booking_ID = 9;
   COMMIT;
```

In this case, two users are trying to book the same driver (Driver_ID 10) for two different bookings (Booking_ID 8 and Booking_ID 9) at the same time. Since the driver can only drive one booking at a time, this results in a conflict where only one user can successfully book the driver, while the other user's booking will fail.

4)

```
   START TRANSACTION;
   UPDATE Bookings SET Pickup = '123 Main St' WHERE Booking_ID = 6;
```

```
COMMIT;


START TRANSACTION;

UPDATE Bookings SET Pickup = '456 Elm St' WHERE Booking_ID = 6;

COMMIT;
```

These two transactions are conflicting because they both attempt to update the same record (with Booking_ID = 6) in the Bookings table. The second transaction will overwrite the changes made by the first transaction, resulting in data loss.