P4. Database Schema Implementation  
DRIVESHARE: Team 2

To implement the base schema for all entity tables and their relationships, we have created the necessary .sql scripts, which are attached and available in the GitHub repository.

In addition, we have incorporated several updates based on previous comments and recommendations. The specifics of these updates are mentioned below.

**Changes / Updates Implemented:**

1. We have updated the RegisteredUser's table attribute from the previously mentioned 'ApproverID' to 'AdminID' to accurately reflect the linkage with the Admin table.
2. The Linkage between the ‘Renter’ and ‘Car’ has been correctly implemented to handle a many-to-many relationship between renters and cars. It tracks which cars are rented by which renters, along with the rental duration which in the P3 incorrectly linked 1 to 1 relationship.  
   Additionally for the table we have primary key as a composite key consisting of both RenterID and CarID as per the feedback provided for P3.
3. For the Location Table, we updated the implementation by adding attributes for both the latitude and longitude of the pickup and drop-off stops in the generated TripRequest. Based on this, we are developing a function to calculate a GeohashID, which will serve as the primary key in the Location table.

To manage drivers for multiple locations, we have created a new table that links drivers to multiple locations. This allows us to update a driver’s location and enables them to have multiple locations. We can track these locations based on the 'last updated' timestamp in this table.

Additionally, a similar implementation can be observed for the linkage between Driver and Car, using a new table called Driver\_Car. This table features a composite key made up of driverID and CarID, accounting for the possibility of a driver using different cars on different working days.

1. Removing the linkage of Ratecard to Location we have implemented a relationship directly with TripRequest and it has been updated to handle the forging key to rate card based on the state of the trip’s starting location
2. The lnvoice and PaymentRequest relationships have been updated with the Completed trips generating an Invoice and then the corresponding payment request linkage to the Invoice and Rider.

**Steps to Implement:**

1. STEP 1 : Creation of the database and functions (Step1\_FunctionsRequired.sql)
2. STEP 2 : Tables creation script (Step2\_DDL\_Scripts.sql)
3. STEP 3 : Sample data insertion script (Step3\_DML\_scripts.sql)
4. STEP 4 : (Optional) To delete the database and all underlying objects (Teardown\_Scipt.sql)

For automatic ID generation, you can use SQL Server's IDENTITY property in conjunction with computed columns that have the PERSISTED attribute.  
**[ID] INT IDENTITY(1,1) NOT NULL**

And a uniqueID is generated as a computed column using string concatenation for major entities for their ID generation.