**Carpool Management System Database**

**Project Summary**

This project is a comprehensive backend database system engineered to manage the complete operational lifecycle of a modern carpool service. It is designed as a robust, scalable, and reliable relational database (RDBMS) that serves as the "single source of truth" for all application activities. The system is built to handle complex, real-time interactions between diverse stakeholders—passengers, drivers, and support administrators.

**Core Objectives**

The primary objective is to design and implement a normalized, high-integrity database schema that ensures data consistency and reliability. The system is engineered to solve key business challenges, including:

* **Reliable Bookings:** Ensuring that a ride booking is an atomic transaction (all-or-nothing).
* **Fair Matching:** Logically matching passengers with the nearest suitable driver.
* **Financial Integrity:** Accurately managing financial transactions and balances for all users.
* **User Trust:** Providing a reliable system for reviews, ratings, and support.

**Key Features & Technical Logic**

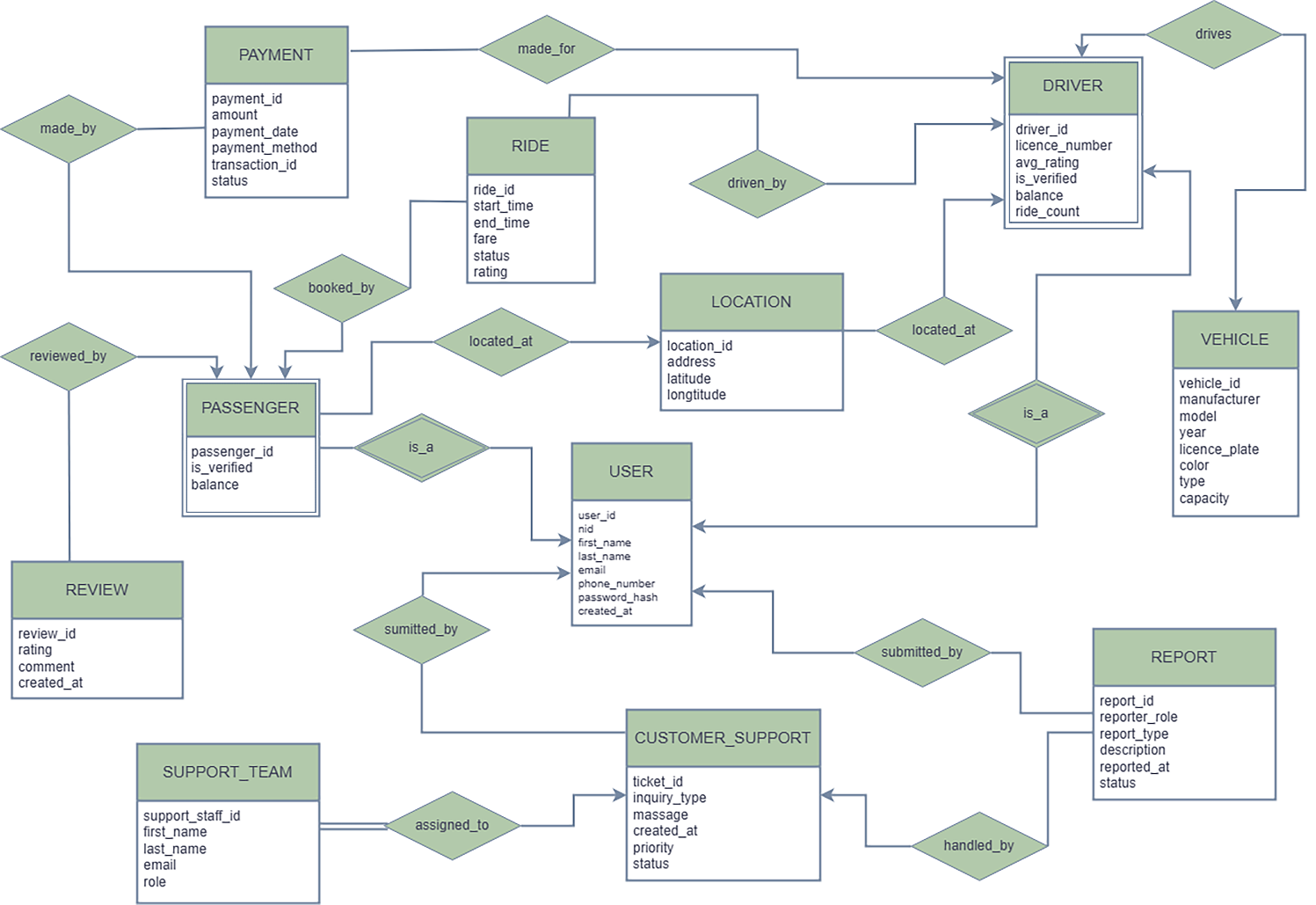
The database schema and its logic are designed to handle complex engineering problems through several key features:

* **Role-Based User Management:** The schema clearly distinguishes between a base User, a Passenger, and a Driver, using foreign keys to link specialized roles to a central identity. This allows for role-specific attributes (e.g., a Driver's avg\_rating and a Passenger's balance).
* **Geospatial Ride Matching:** The system is built to find the nearest driver using complex geospatial analysis (the Haversine formula) by comparing the passenger's latitude/longitude with the locations of all available drivers.
* **Transactional Ride Lifecycle:** The entire ride booking process is wrapped in a single, atomic **transaction**. This critical feature ensures that the fare calculation, the passenger's balance deduction, the driver's balance addition, the ride-count increment, and the creation of the Ride record all succeed or fail together, preventing data corruption.
* **Dynamic Fare & Rating Models:**
  + **Fare:** The fare is not a static value but is calculated dynamically based on the total ride distance (calculated via Haversine) and the Vehicle type.
  + **Rating:** The driver's avg\_rating is a weighted average, accurately recalculated after each ride to provide a fair and evolving representation of their performance.
* **Integrated Support Module:** The system includes a complete subsystem for accountability and support. The Report and Customer\_Support tables allow passengers and drivers to file complaints and tickets, which can then be managed, assigned, and resolved by the Support\_Team.

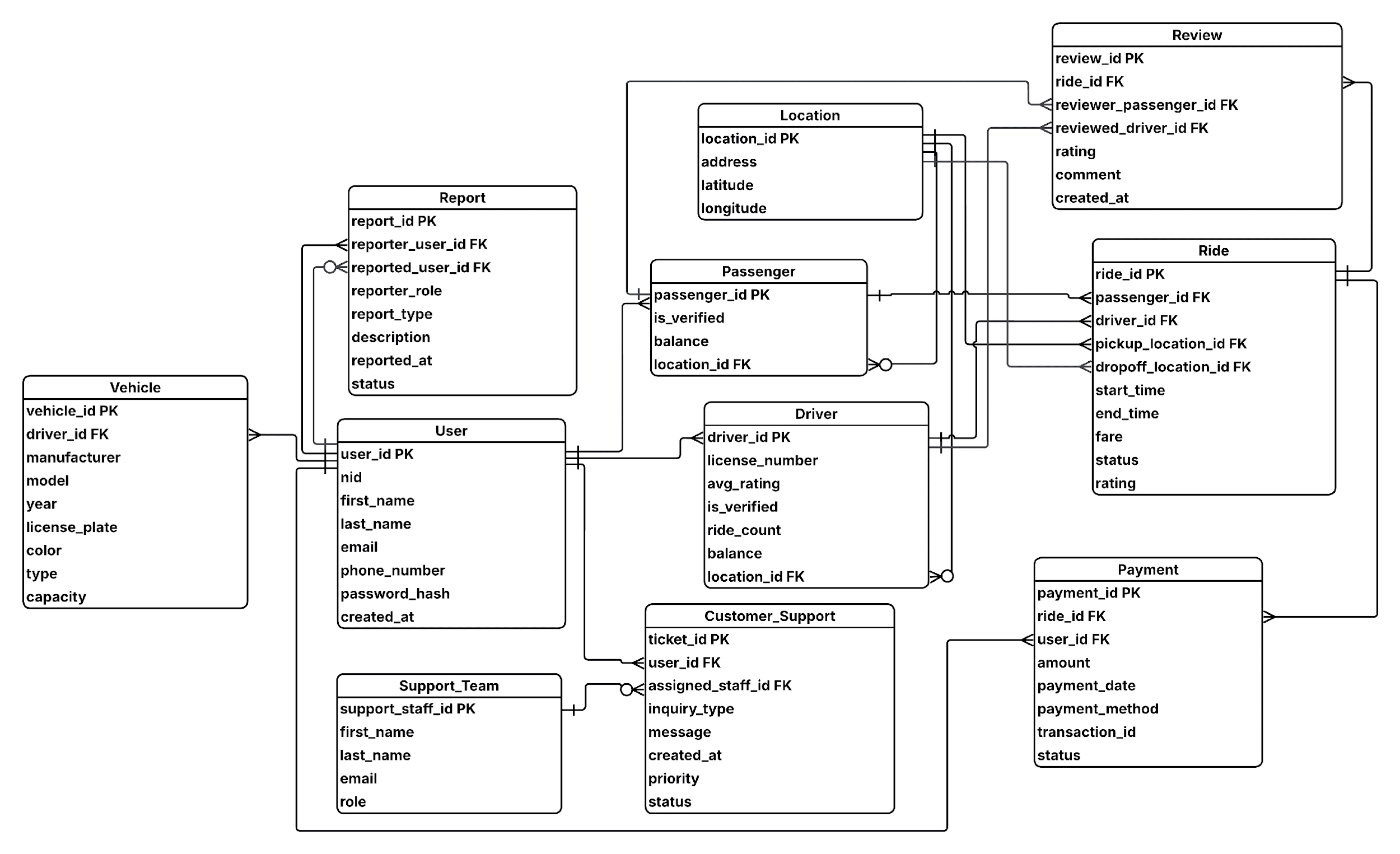
**Technology**

The system is implemented as a relational database using **Microsoft SQL Server**. The design heavily emphasizes data integrity through **normalization (3NF)**, strict **foreign key constraints**, and **transactional logic**.

**ER Diagram**



**Schema Diagram**

****

**DDL QUERY OF TABLE CREATION**

**CREATE DATABASE Carpool**

**GO**

**USE Carpool**

**GO**

**-- USER table to store generic data of driver and passengers**

**CREATE TABLE [User] (**

**user\_id INT PRIMARY KEY,**

**nid VARCHAR(50),**

**first\_name VARCHAR(100) NOT NULL,**

**last\_name VARCHAR(100) NOT NULL,**

**email VARCHAR(255) UNIQUE NOT NULL,**

**phone\_number VARCHAR(20),**

**password\_hash VARCHAR(255) NOT NULL,**

**created\_at DATETIME DEFAULT GETDATE()**

**);**

**-- Location table**

**CREATE TABLE Location (**

**location\_id INT PRIMARY KEY,**

**address VARCHAR(255) NOT NULL,**

**latitude DECIMAL(10, 8) NOT NULL,**

**longitude DECIMAL(11, 8) NOT NULL**

**);**

**-- Vehicle table**

**CREATE TABLE Vehicle (**

**vehicle\_id INT PRIMARY KEY ,**

**driver\_id INT NOT NULL,**

**manufacturer VARCHAR(100),**

**model VARCHAR(100),**

**[year] INT,**

**license\_plate VARCHAR(40) UNIQUE NOT NULL,**

**color VARCHAR(50),**

**[type] VARCHAR(50),**

**capacity INT NOT NULL,**

**FOREIGN KEY (driver\_id) REFERENCES [User](user\_id)**

**);**

**-- Passenger table**

**CREATE TABLE Passenger (**

**passenger\_id INT PRIMARY KEY,**

**is\_verified BIT DEFAULT 0,**

**balance DECIMAL(10, 2) DEFAULT 0.00,**

**location\_id INT NULL,**

**FOREIGN KEY (passenger\_id) REFERENCES [User](user\_id),**

**FOREIGN KEY (location\_id) REFERENCES Location(location\_id)**

**);**

**-- Driver table**

**CREATE TABLE Driver (**

**driver\_id INT PRIMARY KEY,**

**license\_number VARCHAR(50) UNIQUE NOT NULL,**

**avg\_rating DECIMAL(2, 1) DEFAULT 0.0,**

**is\_verified BIT DEFAULT 0,**

**ride\_count INT,**

**balance DECIMAL(10, 2) DEFAULT 0.00,**

**location\_id INT NULL,**

**FOREIGN KEY (driver\_id) REFERENCES [User](user\_id),**

**FOREIGN KEY (location\_id) REFERENCES Location(location\_id)**

**);**

**-- Ride table**

**CREATE TABLE Ride (**

**ride\_id INT PRIMARY KEY,**

**passenger\_id INT NOT NULL,**

**driver\_id INT NOT NULL,**

**pickup\_location\_id INT NOT NULL,**

**dropoff\_location\_id INT NOT NULL,**

**start\_time DATETIME NOT NULL,**

**end\_time DATETIME NULL,**

**fare DECIMAL(8, 2) NOT NULL,**

**[status] VARCHAR(50) NOT NULL,**

**rating DECIMAL(2, 1) DEFAULT 0.0,**

**FOREIGN KEY (passenger\_id) REFERENCES Passenger(passenger\_id),**

**FOREIGN KEY (driver\_id) REFERENCES Driver(driver\_id),**

**FOREIGN KEY (pickup\_location\_id) REFERENCES Location(location\_id),**

**FOREIGN KEY (dropoff\_location\_id) REFERENCES Location(location\_id)**

**);**

**-- Payment table**

**CREATE TABLE Payment (**

**payment\_id INT PRIMARY KEY,**

**ride\_id INT UNIQUE NOT NULL,**

**user\_id INT NOT NULL,**

**amount DECIMAL(8, 2) NOT NULL,**

**payment\_date DATETIME DEFAULT GETDATE(),**

**payment\_method VARCHAR(50),**

**transaction\_id VARCHAR(100) UNIQUE NOT NULL,**

**[status] VARCHAR(50) NOT NULL,**

**FOREIGN KEY (ride\_id) REFERENCES Ride(ride\_id),**

**FOREIGN KEY (user\_id) REFERENCES [User](user\_id)**

**);**

**-- Review table**

**CREATE TABLE Review (**

**review\_id INT PRIMARY KEY ,**

**ride\_id INT NOT NULL,**

**reviewer\_passenger\_id INT NOT NULL,**

**reviewed\_driver\_id INT NOT NULL,**

**rating INT CHECK (rating BETWEEN 1 AND 5) NOT NULL,**

**[comment] VARCHAR(MAX),**

**created\_at DATETIME DEFAULT GETDATE(),**

**FOREIGN KEY (ride\_id) REFERENCES Ride(ride\_id),**

**FOREIGN KEY (reviewer\_passenger\_id) REFERENCES Passenger(passenger\_id),**

**FOREIGN KEY (reviewed\_driver\_id) REFERENCES Driver(driver\_id)**

**);**

**-- Support Team table**

**CREATE TABLE Support\_Team (**

**support\_staff\_id INT PRIMARY KEY IDENTITY(1,1),**

**first\_name VARCHAR(100) NOT NULL,**

**last\_name VARCHAR(100) NOT NULL,**

**email VARCHAR(255) UNIQUE NOT NULL,**

**[role] VARCHAR(50)**

**);**

**-- Report table**

**CREATE TABLE Report (**

**report\_id INT PRIMARY KEY,**

**reporter\_user\_id INT NOT NULL,**

**reported\_user\_id INT NULL,**

**reporter\_role VARCHAR(20) NOT NULL,**

**report\_type VARCHAR(100) NOT NULL,**

**[description] VARCHAR(MAX) NOT NULL,**

**reported\_at DATETIME DEFAULT GETDATE(),**

**[status] VARCHAR(50) NOT NULL,**

**FOREIGN KEY (reporter\_user\_id) REFERENCES [User](user\_id),**

**FOREIGN KEY (reported\_user\_id) REFERENCES [User](user\_id),**

**);**

**-- Customer Support table**

**CREATE TABLE Customer\_Support (**

**ticket\_id INT PRIMARY KEY,**

**user\_id INT NOT NULL,**

**assigned\_staff\_id INT NULL,**

**inquiry\_type VARCHAR(100),**

**[message] VARCHAR(MAX) NOT NULL,**

**created\_at DATETIME DEFAULT GETDATE(),**

**priority VARCHAR(20),**

**[status] VARCHAR(50) NOT NULL,**

**FOREIGN KEY (user\_id) REFERENCES [User](user\_id),**

**FOREIGN KEY (assigned\_staff\_id) REFERENCES Support\_Team(support\_staff\_id)**

**);**

**DATA INSERTION**

**-- User Data**

**INSERT INTO [User] (user\_id, nid, first\_name, last\_name, email, phone\_number, password\_hash) VALUES**

**(1, '1995123456', 'Rahim', 'Ahmed', 'rahim.a@email.com', '01711223344', 'e10adc3949'),**

**(2, '1988765432', 'Fatima', 'Begum', 'fatima.b@email.com', '01822334455', 'f379eaf3c8'),**

**(3, '2001987654', 'Kamal', 'Hossain', 'kamal.h@email.com', '01933445566', '5170365a68'),**

**(4, '1990112233', 'Anika', 'Chowdhury', 'anika.c@email.com', '01644556677', '8a5433a388'),**

**(5, '1985445566', 'Tariq', 'Khan', 'tariq.k@email.com', '01555667788', 'd8578edf84'),**

**(6, '1992778899', 'Sadia', 'Islam', 'sadia.i@email.com', '01766778899', '1679091c5a'),**

**(7, '1980101010', 'Jamal', 'Uddin', 'jamal.u@email.com', '01877889900', '25d55ad283'),**

**(8, '2000202020', 'Nusrat', 'Jahan', 'nusrat.j@email.com', '01988990011', 'e99a18c428'),**

**(9, '1998303030', 'Iqbal', 'Mahmud', 'iqbal.m@email.com', '01699001122', 'c333677015'),**

**(10,'1987404040', 'Rifat', 'Sheikh', 'rifat.s@email.com', '01500112233', '900150983c'),**

**(11,'1993505050', 'Sumaiya', 'Akter', 'sumaiya.a@email.com', '01712345678', '02c75fb22c'),**

**(12,'1989606060', 'Hasan', 'Mahmud', 'hasan.m@email.com', '01823456789', 'fcea920f74'),**

**(13,'2002707070', 'Farhana', 'Yasmin', 'farhana.y@email.com', '01934567890', 'a3dcb4d229'),**

**(14,'1996808080', 'Arif', 'Rahman', 'arif.r@email.com', '01645678901', '7694f4a663'),**

**(15, '1984909090', 'Nazmul', 'Sarkar', 'nazmul.s@email.com', '01556789012', '1f32aa4c9a'),**

**(16, '1999121212', 'Sharmin', 'Sultana', 'sharmin.s@email.com', '01767890123', '4e4d6c332b'),**

**(17, '1986232323', 'Shakib', 'Hasan', 'shakib.h@email.com', '01878901234', 'b59c67bf19'),**

**(18, '2003343434', 'Ayesha', 'Siddika', 'ayesha.s@email.com', '01989012345', 'c56d0e9a7c'),**

**(19, '1991454545', 'Mehedi', 'Hasan', 'mehedi.h@email.com', '01690123456', '5f4dcc3b5a'),**

**(20, '1982565656', 'Tasnim', 'Ferdous', 'tasnim.f@email.com', '01501234567', '3d2172418e');**

**-- Location data**

**INSERT INTO Location VALUES**

**(101, 'Abdullahpur Bus Stand, Uttara, Dhaka', 23.8794, 90.4005),**

**(102, 'Jatrabari Chowrasta, Dhaka', 23.7099, 90.4333),**

**(103, 'Uttara Sector 6 Park, Dhaka', 23.8647, 90.3989),**

**(104, 'Hazrat Shahjalal International Airport, Dhaka', 23.8433, 90.4043),**

**(105, 'Jamuna Future Park, Bashundhara, Dhaka', 23.8133, 90.4243),**

**(106, 'Banani Road 11, Dhaka', 23.7942, 90.4069),**

**(107, 'Gulshan 2 Circle, Dhaka', 23.7925, 90.4183),**

**(108, 'Mohakhali Bus Terminal, Dhaka', 23.7820, 90.4048),**

**(109, 'Farmgate, Tejgaon, Dhaka', 23.7595, 90.3892),**

**(110, 'Dhanmondi 32 Bridge, Dhaka', 23.7533, 90.3813),**

**(111, 'Mohammadpur Town Hall, Dhaka', 23.7644, 90.3667),**

**(112, 'New Market, Dhaka', 23.7346, 90.3876),**

**(113, 'Shahbag Intersection, Dhaka', 23.7410, 90.3958),**

**(114, 'Motijheel Shapla Chattar, Dhaka', 23.7259, 90.4172),**

**(115, 'Sadarghat Launch Terminal, Dhaka', 23.7067, 90.4099);**

**-- Passenger Data**

**INSERT INTO Passenger VALUES**

**(1, 1, 450.50, 101),**

**(3, 1, 1200.00, 103),**

**(5, 0, 125.75, 105),**

**(7, 1, 880.90, 107),**

**(9, 0, 50.00, 109),**

**(11, 1, 200.00, 111),**

**(13, 0, 150.50, 113),**

**(15, 1, 999.99, 115),**

**(16, 1, 300.00, 101),**

**(19, 1, 110.00, 105);**

**-- Driver Data**

**INSERT INTO Driver**

**VALUES**

**(2, 'BD-012345678', 4.8, 1, 110, 5500.25, 102),**

**(4, 'BD-023456789', 4.5, 1, 90, 3200.50, 104),**

**(6, 'BD-034567890', 4.2, 1, 50, 1500.00, 106),**

**(8, 'BD-045678901', 4.9, 1, 200, 8750.80, 108),**

**(10, 'BD-056789012', 4.1, 1, 30, 950.00, 110),**

**(12, 'BD-067890123', 4.7, 1, 150, 6000.00, 112),**

**(14, 'BD-078901234', 4.3, 1, 75, 2550.00, 114),**

**(17, 'BD-089012345', 4.6, 1, 120, 4300.00, 102),**

**(18, 'BD-090123456', 4.4, 1, 100, 3850.00, 103),**

**(20, 'BD-101234567', 4.0, 1, 65, 2100.00, 106);**

**-- Vehicle Data**

**INSERT INTO Vehicle VALUES**

**(1, 2, 'Toyota', 'Premio', 2018, 'Dhaka Metro-Ga 35-1121', 'Silver', 'Sedan', 4),**

**(2, 4, 'Toyota', 'Aqua', 2019, 'Dhaka Metro-Kha 22-3456', 'White', 'Hybrid', 4),**

**(3, 6, 'Honda', 'Vezel', 2020, 'Dhaka Metro-Gha 41-7890', 'Black', 'SUV', 5),**

**(4, 8, 'Toyota', 'Allion', 2017, 'Ctg Metro-Ka 19-2345', 'Red', 'Sedan', 4),**

**(5, 10, 'Toyota', 'HiAce', 2019, 'Dhaka Metro-Cha 53-6789', 'White', 'Microbus', 11),**

**(6, 12, 'Mitsubishi', 'Outlander', 2021, 'Dhaka Metro-Gha 45-1011', 'Grey', 'SUV', 6),**

**(7, 14, 'Toyota', 'Axio', 2020, 'Dhaka Metro-Ga 39-2122', 'Black', 'Sedan', 4),**

**(8, 17, 'Suzuki', 'Swift', 2022, 'Dhaka Metro-Ka 28-3233', 'Blue', 'Hatchback', 4),**

**(9, 18, 'Nissan', 'X-Trail', 2018, 'Dhaka Metro-Gha 33-4344', 'Pearl White', 'SUV', 5),**

**(10, 20, 'Honda', 'Grace', 2019, 'Dhaka Metro-Kha 25-5455', 'Silver', 'Hybrid', 4);**

**-- Support Team data**

**INSERT INTO Support\_Team VALUES**

**('Karim', 'Hossain', 'k.hossain@support.com', 'Tier 1 Agent'),**

**('Tasnia', 'Begum', 't.begum@support.com', 'Tier 2 Specialist'),**

**('Imran', 'Khan', 'i.khan@support.com', 'Manager'),**

**('Sadia', 'Akter', 's.akter@support.com', 'Tier 1 Agent'),**

**('Jamil', 'Ahmed', 'j.ahmed@support.com', 'Tier 1 Agent'),**

**('Afsana', 'Chowdhury', 'a.chowdhury@support.com', 'Tier 2 Specialist'),**

**('Rohan', 'Islam', 'r.islam@support.com', 'Tier 1 Agent'),**

**('Maria', 'Rahman', 'm.rahman@support.com', 'Tier 1 Agent'),**

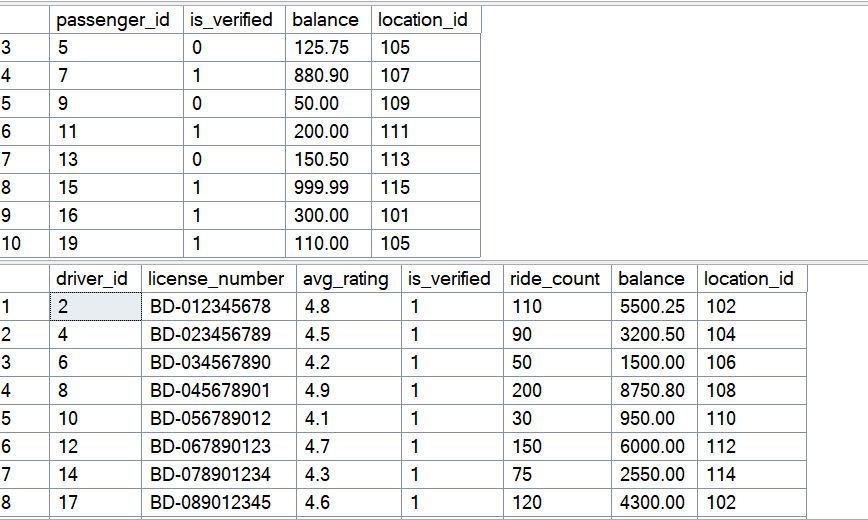
**('Faisal', 'Haque', 'f.haque@support.com', 'Tier 2 Specialist'),**

**('Farhana', 'Sultana', 'f.sultana@support.com', 'Manager');**

**DML QUERY**

**1. Ride Booking (Passenger 15)**

Before booking ride:

****

**BEGIN TRAN;**

**BEGIN TRY**

**-- Paasenger info and Preference**

**DECLARE @PassengerIDToFind INT = 15;**

**DECLARE @DestinationLocationID INT = 111;**

**DECLARE @VehicleTypePreference VARCHAR(50) = 'Microbus';**

**DECLARE @PassengerID INT;**

**DECLARE @PassengerLocationID INT;**

**DECLARE @PassengerLat DECIMAL(10, 8);**

**DECLARE @PassengerLon DECIMAL(11, 8);**

**DECLARE @PassengerBalance DECIMAL(10, 2);**

**-- Destination Info & Fare**

**DECLARE @DestinationLat DECIMAL(10, 8);**

**DECLARE @DestinationLon DECIMAL(11, 8);**

**DECLARE @NearestDriverID INT;**

**DECLARE @RatePerKm DECIMAL(4, 2);**

**DECLARE @CalculatedFare DECIMAL(8, 2);**

**DECLARE @CalculatedDistanceKM DECIMAL(10, 2);**

**--New Ride Info**

**DECLARE @NewRideID INT;**

**DECLARE @NewReviewID INT;**

**DECLARE @NewPaymentID INT;**

**DECLARE @NewRatingGiven DECIMAL(2, 1) = 4.0;**

**-- Find the passenger's details using their ID**

**SELECT**

**@PassengerID = P.passenger\_id,**

**@PassengerLocationID = P.location\_id,**

**@PassengerLat = L.latitude,**

**@PassengerLon = L.longitude,**

**@PassengerBalance = P.balance**

**FROM**

**Passenger AS P**

**JOIN**

**[User] AS U ON P.passenger\_id = U.user\_id**

**JOIN**

**Location AS L ON P.location\_id = L.location\_id**

**WHERE**

**U.user\_id = @PassengerIDToFind;**

**-- Find the destination's coordinates**

**SELECT**

**@DestinationLat = latitude,**

**@DestinationLon = longitude**

**FROM**

**Location**

**WHERE**

**location\_id = @DestinationLocationID;**

**--- Finding Nearest Driver**

**SELECT TOP 1**

**@NearestDriverID = D.driver\_id**

**FROM**

**Driver AS D**

**JOIN**

**Location AS L ON D.location\_id = L.location\_id**

**JOIN**

**Vehicle AS V ON D.driver\_id = V.driver\_id**

**CROSS APPLY (**

**-- Calculating the distance in KM using the Haversine formula**

**SELECT**

**(6371 \* 2 \* ATN2(**

**SQRT(**

**SIN(RADIANS(L.latitude - @PassengerLat) / 2) \* SIN(RADIANS(L.latitude - @PassengerLat) / 2) +**

**COS(RADIANS(@PassengerLat)) \* COS(RADIANS(L.latitude)) \***

**SIN(RADIANS(L.longitude - @PassengerLon) / 2) \* SIN(RADIANS(L.longitude - @PassengerLon) / 2)**

**),**

**SQRT(1 - (**

**SIN(RADIANS(L.latitude - @PassengerLat) / 2) \* SIN(RADIANS(L.latitude - @PassengerLat) / 2) +**

**COS(RADIANS(@PassengerLat)) \* COS(RADIANS(L.latitude)) \***

**SIN(RADIANS(L.longitude - @PassengerLon) / 2) \* SIN(RADIANS(L.longitude - @PassengerLon) / 2)**

**))**

**)) AS Kilometers**

**) AS Dist**

**WHERE**

**D.is\_verified = 1**

**AND V.[type] = @VehicleTypePreference**

**ORDER BY**

**Dist.Kilometers ASC;**

**IF @NearestDriverID IS NULL**

**BEGIN**

**RAISERROR('No available drivers found.', 16, 1);**

**ROLLBACK TRAN;**

**RETURN;**

**END;**

**-- Calculating fare**

**SET @CalculatedDistanceKM = (**

**6371 \* 2 \* ATN2(**

**SQRT(**

**SIN(RADIANS(@DestinationLat - @PassengerLat) / 2) \* SIN(RADIANS(@DestinationLat - @PassengerLat) / 2) +**

**COS(RADIANS(@PassengerLat)) \* COS(RADIANS(@DestinationLat)) \***

**SIN(RADIANS(@DestinationLon - @PassengerLon) / 2) \* SIN(RADIANS(@DestinationLon - @PassengerLon) / 2)**

**),**

**SQRT(1 - (**

**SIN(RADIANS(@DestinationLat - @PassengerLat) / 2) \* SIN(RADIANS(@DestinationLat - @PassengerLat) / 2) +**

**COS(RADIANS(@PassengerLat)) \* COS(RADIANS(@DestinationLat)) \***

**SIN(RADIANS(@DestinationLon - @PassengerLon) / 2) \* SIN(RADIANS(@DestinationLon - @PassengerLon) / 2)**

**))**

**)**

**);**

**-- fare by Vehicle catagory**

**SET @RatePerKm = CASE @VehicleTypePreference**

**WHEN 'Sedan' THEN 25.00**

**WHEN 'SUV' THEN 35.00**

**WHEN 'Hybrid' THEN 22.00**

**WHEN 'Microbus' THEN 40.00**

**ELSE 20.00 -- Default**

**END;**

**SET @CalculatedFare = @CalculatedDistanceKM \* @RatePerKm;**

**-- Check if passenger has enough balance**

**IF @PassengerBalance < @CalculatedFare**

**BEGIN**

**RAISERROR('Insufficient balance for this ride.', 16, 1);**

**ROLLBACK TRAN;**

**RETURN;**

**END;**

**-- Creating new Ride record**

**SELECT @NewRideID = ISNULL(MAX(ride\_id), 0) + 1 FROM Ride;**

**INSERT INTO Ride**

**VALUES (**

**@NewRideID,**

**@PassengerID,**

**@NearestDriverID,**

**@PassengerLocationID,**

**@DestinationLocationID,**

**DATEADD(minute, -15, GETDATE()),**

**GETDATE(),**

**@CalculatedFare,**

**'completed',**

**@NewRatingGiven**

**);**

**-- Updating Passenger balance**

**UPDATE Passenger**

**SET balance = balance - @CalculatedFare**

**WHERE passenger\_id = @PassengerID;**

**-- Updating Driver balance, ride\_count and avg\_rating**

**UPDATE Driver**

**SET**

**balance = balance + @CalculatedFare,**

**avg\_rating = CASE**

**WHEN ride\_count = 0 THEN @NewRatingGiven**

**ELSE ( (avg\_rating \* ride\_count) + @NewRatingGiven ) / (ride\_count + 1)**

**END,**

**ride\_count = ride\_count + 1**

**WHERE**

**driver\_id = @NearestDriverID;**

**-- Add to Review table**

**SELECT @NewReviewID = ISNULL(MAX(review\_id), 0) + 1 FROM Review;**

**INSERT INTO Review (**

**review\_id,**

**ride\_id,**

**reviewer\_passenger\_id,**

**reviewed\_driver\_id,**

**rating,**

**[comment]**

**)**

**VALUES (**

**@NewReviewID,**

**@NewRideID,**

**@PassengerID,**

**@NearestDriverID,**

**@NewRatingGiven,**

**'The driver was very professional. A truly amazing and safe ride.'**

**);**

**SELECT @NewPaymentID = ISNULL(MAX(payment\_id), 0) + 1 FROM Payment;**

**-- Inserting Payment**

**INSERT INTO Payment**

**VALUES (**

**@NewPaymentID,**

**@NewRideID,**

**@PassengerID,**

**@CalculatedFare,**

**GETDATE(),**

**CASE**

**WHEN @NewRideID % 3 = 0 THEN 'Cash'**

**WHEN @NewRideID % 3 = 1 THEN 'Credit Card'**

**ELSE 'Digital Wallet'**

**END,**

**CONCAT('TXN', RIGHT('0000' + CAST(@NewRideID AS VARCHAR(10)), 4)),**

**'Success'**

**);**

**COMMIT TRAN;**

**SELECT**

**'Ride Booked, Completed & Paid Successfully!' AS Result,**

**@NewRideID AS NewRideID,**

**@NewPaymentID AS NewPaymentID,**

**U\_Pass.first\_name AS PassengerName,**

**U\_Drive.first\_name AS DriverName,**

**@CalculatedFare AS Fare,**

**@CalculATEDDistanceKM AS DistanceKM**

**FROM**

**[User] AS U\_Pass**

**JOIN**

**[User] AS U\_Drive ON U\_Drive.user\_id = @NearestDriverID**

**WHERE**

**U\_Pass.user\_id = @PassengerID;**

**END TRY**

**BEGIN CATCH**

**ROLLBACK TRAN;**

**SELECT**

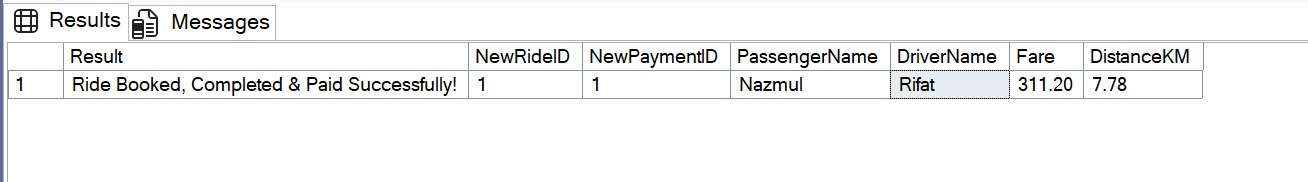
**ERROR\_NUMBER() AS ErrorNumber,**

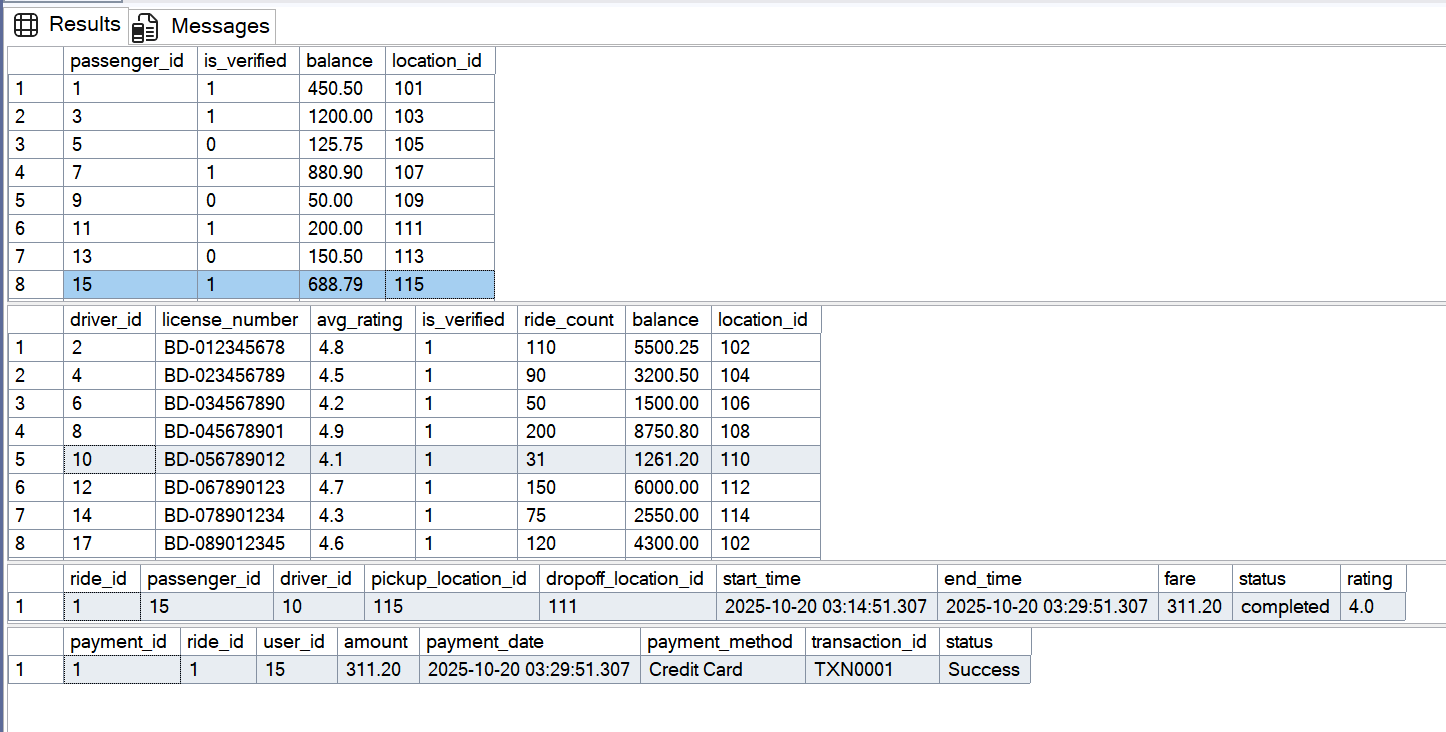
**ERROR\_MESSAGE() AS ErrorMessage,**

**ERROR\_LINE() AS ErrorLine;**

**END CATCH**

**GO**

**AFTER BOOKING A RIDE:**

****

This test case demonstrates the end-to-end ride booking process, using a passenger preference for a 'Microbus'. The system correctly identifies and assigns the only matching provider, Driver ID 10. Following the ride, the script programmatically commits all associated data, creating new records in the Ride, Payment, and Review tables. This core transaction also processes the financial exchange, deducting the calculated fare from the passenger's balance and crediting it to the driver's account.

**2.** **Report Insertion**

**DECLARE @NewReportID INT = (SELECT ISNULL(MAX(report\_id), 0) + 1 FROM Report);**

**DECLARE @UserWhoIsReporting INT = 15;**

**DECLARE @UserBeingReported INT = 10;**

**INSERT INTO Report (**

**report\_id,**

**reporter\_user\_id,**

**reported\_user\_id,**

**reporter\_role,**

**report\_type,**

**[description],**

**[status]**

**)**

**VALUES (**

**@NewReportID,**

**@UserWhoIsReporting,**

**@UserBeingReported,**

**'Passenger',**

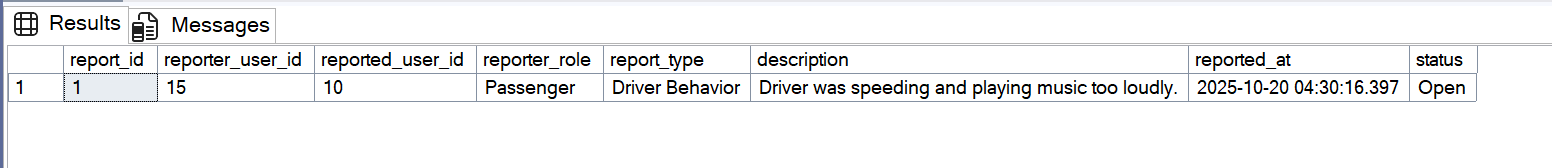
**'Driver Behavior',**

**'Driver was speeding and playing music too loudly.',**

**'Open'**

**);**

**SELECT \* FROM Report WHERE report\_id = @NewReportID;**

****

**3.** **Checking Ticket History (User 3)**

**BEGIN TRAN;**

**BEGIN TRY**

**DECLARE @SubmittingUserID INT = 3;**

**DECLARE @InquiryType VARCHAR(100) = 'Account Help';**

**DECLARE @MessageText VARCHAR(MAX) = 'I forgot my password and cannot reset it.';**

**DECLARE @Priority VARCHAR(20) = 'High';**

**-- Find the next available ticket\_id**

**DECLARE @NewTicketID INT;**

**SET @NewTicketID = (SELECT ISNULL(MAX(ticket\_id), 0) + 1 FROM Customer\_Support);**

**-- Insert the new ticket record**

**INSERT INTO Customer\_Support**

**(**

**ticket\_id,**

**user\_id,**

**assigned\_staff\_id,**

**inquiry\_type,**

**[message],**

**[priority],**

**[status]**

**)**

**VALUES (**

**@NewTicketID,**

**@SubmittingUserID,**

**NULL,**

**@InquiryType,**

**@MessageText,**

**@Priority,**

**'Pending Staff Review'**

**);**

**-- Assigned Staff to Handle Ticket**

**DECLARE @AssignToStaffID INT = 2;**

**UPDATE Customer\_Support**

**SET**

**assigned\_staff\_id = @AssignToStaffID,**

**[status] = 'In Progress'**

**WHERE**

**ticket\_id = @NewTicketID;**

**SELECT**

**'My Open Ticket' AS ViewTitle,**

**CS.ticket\_id,**

**CS.inquiry\_type,**

**CS.[status],**

**CS.created\_at,**

**CS.[message],**

**ISNULL(ST.first\_name + ' ' + ST.last\_name, 'Not Assigned Yet') AS AssignedStaff**

**FROM**

**Customer\_Support AS CS**

**LEFT JOIN**

**Support\_Team AS ST ON CS.assigned\_staff\_id = ST.support\_staff\_id**

**WHERE**

**CS.ticket\_id = @NewTicketID;**

**UPDATE Customer\_Support**

**SET**

**[status] = 'Resolved'**

**WHERE**

**ticket\_id = @NewTicketID;**

**COMMIT TRAN;**

**END TRY**

**BEGIN CATCH**

**ROLLBACK TRAN;**

**SELECT**

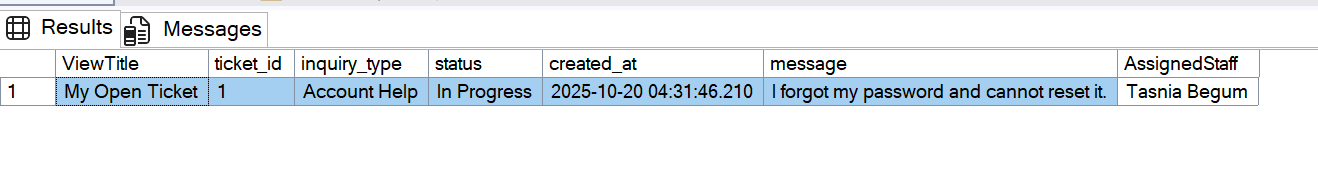
**ERROR\_NUMBER() AS ErrorNumber,**

**ERROR\_MESSAGE() AS ErrorMessage,**

**ERROR\_LINE() AS ErrorLine;**

**END CATCH**

**GO**

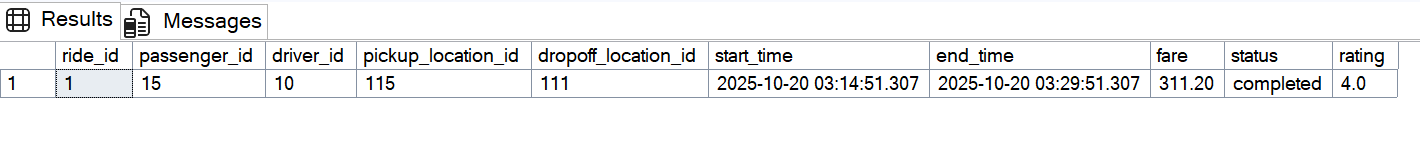
****

**4.** **Ride History for Passenger 15**

This query is executed to retrieve all ride records for Passenger 15.

**SELECT \* FROM Ride**

**WHERE passenger\_id = 15**

****

**5.** **Passenger Balance (Passenger ID 15)**

**SELECT**

**P.passenger\_id AS Passenger\_ID,**

**concat(U.first\_name ,' ', U.last\_name) AS Passenger\_Name,**

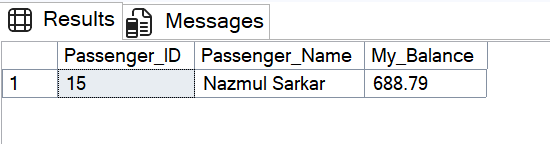
**P.balance AS My\_Balance**

**FROM Passenger AS P**

**JOIN [User] AS U**

**ON P.passenger\_id = U.user\_id**

**WHERE P.passenger\_id = 15**

****

**6.** **Driver Balance (Driver ID 10)**

**SELECT**

**D.driver\_id AS Driver\_ID,**

**CONCAT(U.first\_name, ' ', U.last\_name) AS Driver\_Name,**

**D.balance AS My\_Balance**

**FROM**

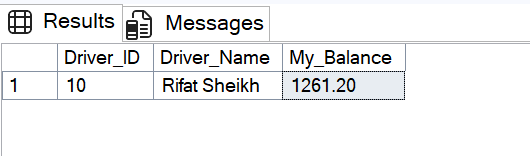
**Driver AS D**

**JOIN**

**[User] AS U ON D.driver\_id = U.user\_id**

**WHERE**

**D.driver\_id = 10;**

****

**7.** **Data Verification: Ride History for Driver 10**

**SELECT**

**R.ride\_id,R.passenger\_id, -- This selects all columns from the Ride table**

**CONCAT(U\_Pass.first\_name, ' ', U\_Pass.last\_name) AS Passenger\_Name,**

**R.driver\_id,**

**CONCAT(U\_Drive.first\_name, ' ', U\_Drive.last\_name) AS DriverName,**

**R.pickup\_location\_id,R.dropoff\_location\_id,R.start\_time,R.end\_time,R.fare,R.status,R.rating**

**FROM**

**Ride AS R**

**JOIN**

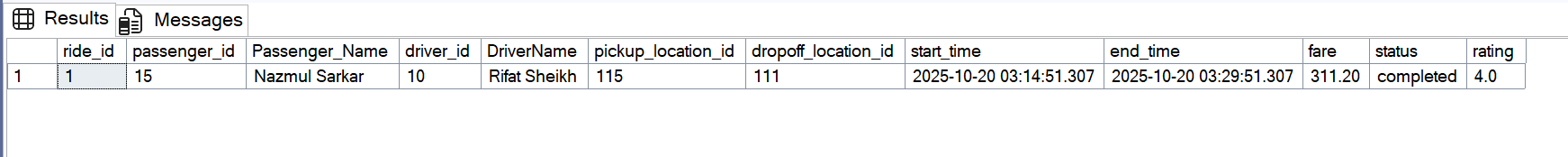
**[User] AS U\_Pass ON R.passenger\_id = U\_Pass.user\_id -- Join to get Passenger name**

**JOIN**

**[User] AS U\_Drive ON R.driver\_id = U\_Drive.user\_id -- Join to get Driver name**

**WHERE**

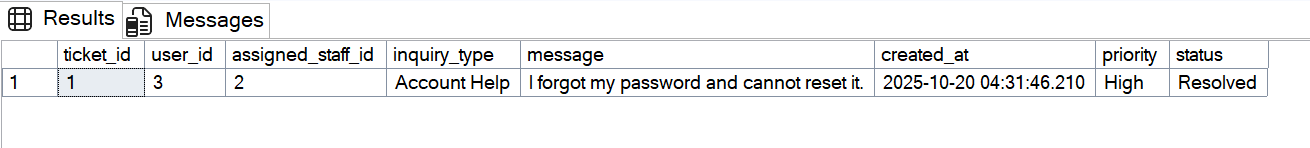
**R.driver\_id = 10;**

****

**8.** **Checking Ticket History for User 3**

**SELECT \* FROM Customer\_Support**

**WHERE user\_id = 3**

****

**CEP Based Mapping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **K's** | **Attributes** | **How K’s are Addressed Through The Project** | **COs** | **POs** |
| **K2** | **Mathematics/Formal Computing** | **Applies Haversine formula for distance and uses statistics to update the driver's average rating avg\_rating.** | **CLO2, CLO3** | **PO(b)** |
| **K4** | **Engineering Specialist Knowledge** | **Implements the Relational Model and ensures data integrity by enforcing ACID properties via database transactions.** | **CLO2, CLO7** | **PO(a)** |
| **K5** | **Engineering Design** | **Designs a functional database schema (tables, keys, constraints) to accurately model all entities of the carpool system.** | **CLO3, CLO4, CLO7** | **PO(c)** |
| **K6** | **Engineering Practice (Technology)** | **Utilizes SQL (DDL, DML) and transactional scripts to implement core business logic like ride booking and payment processing.** | **CLO1, CLO2, CLO7** | **PO(e)** |
| **K7** | **Engineering in Society/Ethics** | **Integrates features like is\\_verified, rating/review systems, and Report mechanisms to ensure safety, ethics, and accountability.** | **CLO5** | **PO(f), PO(g), PO(h)** |
| **K8** | **Research Literature** | **Applies research-based knowledge on spatial calculations (Haversine) and standard database management best practices.** | **CLO3, CLO4** | **PO(d)** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P's** | **Attributes** | **How P’s are Addressed Through The Project** | **COs** | **POs** |
| **P1** | **Depth of knowledge required** | **Requires in-depth knowledge of K4 (Relational Model, ACID transactions) and K5 (Database Design) to build a robust and correct transactional system.** | **CLO2, CLO7** | **PO(a), PO(c)** |
| **P2** | **Range of conflicting requirements** | **Involves conflicting requirements such as balancing user convenience with data integrity (via transactions) and balancing the need for driver availability with vehicle type preferences.** | **CLO3, CLO5** | **PO(c), PO(f)** |
| **P3** | **Depth of analysis required** | **Requires abstract thinking to design a model (schema, relationships, constraints) that accurately represents a real-world carpool system and its complex processes (e.g., the transactional ride booking script).** | **CLO3, CLO4** | **PO(b)** |
| **P6** | **Extent of stakeholder involvement and conflicting needs** | **Addresses diverse stakeholder needs: Passengers (easy booking, safety, balance check), Drivers (fair rating, accurate balance/earnings), and Management (reporting, support, verification).** | **CLO5** | **PO(f), PO(h)** |
| **P7** | **Interdependence** | **Involves a high-level problem (Carpool System) broken into interdependent component parts/sub-problems (User management, Ride matching, Payment, Review, Support, Reporting), all linked via foreign keys.** | **CLO4, CLO7** | **PO(c)** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A's** | **Attributes** | **How A’s are Addressed Through The Project** | **COs** | **POs** |
| **A1** | **Range of resources** | **Manages diverse information/data resources (Users, Locations, Vehicles, Rides, Payments, Reports) and models people (Drivers, Passengers, Support Staff) within a single system.** | **CLO3, CLO4, CLO7** | **PO(c)** |
| **A2** | **Level of interaction** | **Resolves significant problems arising from conflicting issues, such as optimizing ride assignment (nearest driver vs. vehicle preference) while maintaining transactional data integrity.** | **CLO2, CLO7** | **PO(b), PO(c)** |
| **A3** | **Innovation** | **Involves the creative use of engineering principles (database transactions, Haversine formula, rating systems) to model and automate complex, real-world logistics (the ride booking process).** | **CLO3, CLO4** | **PO(c)** |
| **A4** | **Consequences for society and the environment** | **Models activities with significant societal consequences (safety, finance, legal) by including features for driver/passenger verification (is\\_verified), support, and formal reporting.** | **CLO5** | **PO(f), PO(h)** |
| **A5** | **Familiarity** | **Requires applying principles-based approaches (Relational Algebra/Model) to a specific, potentially new domain (a carpool management system) to ensure all operational requirements are met.** | **CLO3, CLO4** | **PO(b)** |