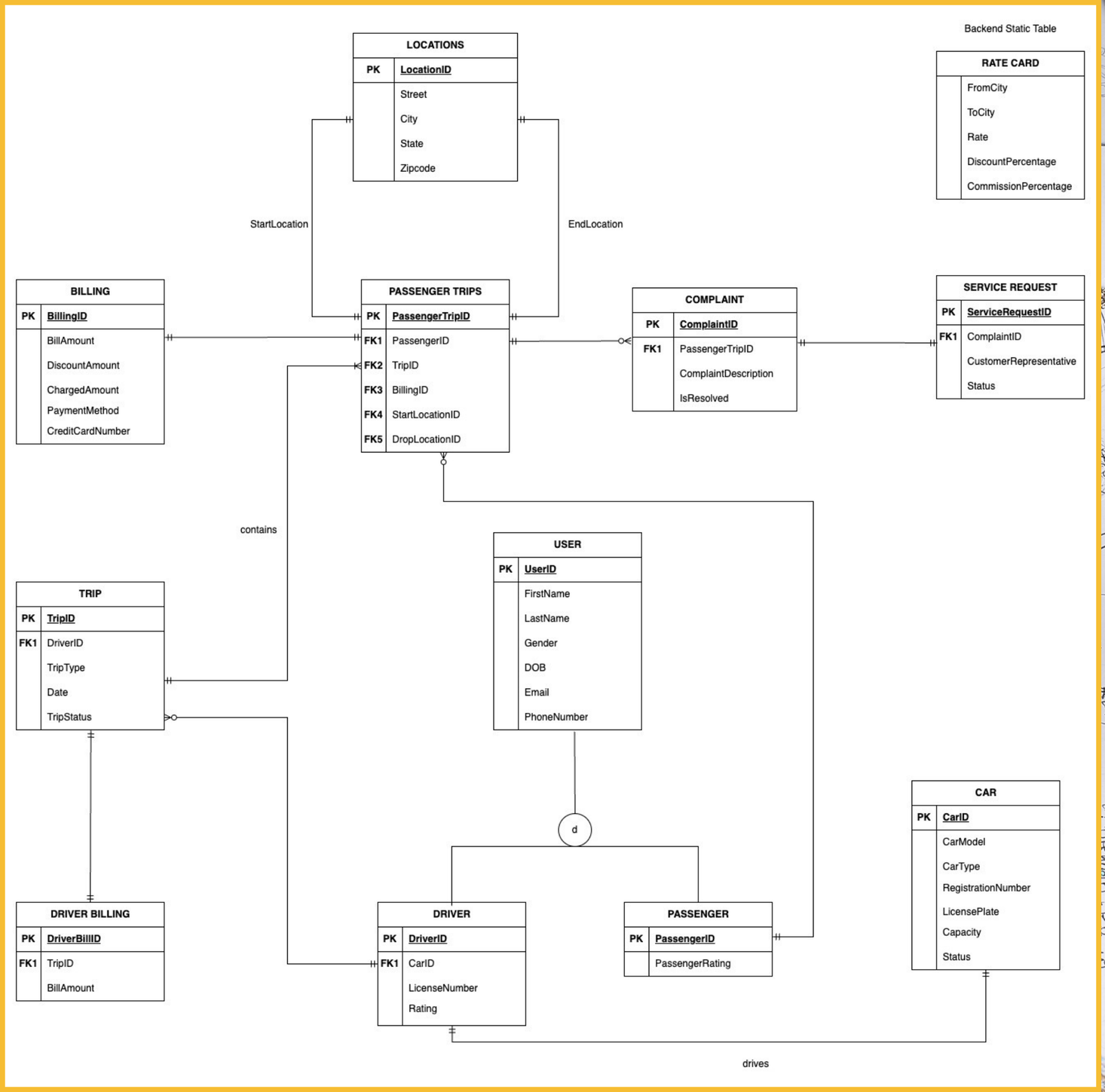
URBAN RIDE-SHARING MANAGEMENT SYSTEM

DAMG6210 : DATA MANAGEMENT AND DATABASE DESIGN
MANUEL MONTROND

TEAM MEMBERS:
SAHITI NALLAMOLU
JUGAL WABLE
PRACHI NAVALE
RHEA SHAH
RONAK MISHRA

OVERVIEW

The Urban Ride-Sharing Management System emerges as a smart friend for city travel, ensuring that we get where we need to go while still being environmentally cum pocket friendly. This centralized database will include information on drivers and passengers who are willing to carpool to nearby destinations. It will help store, manage, organize, maintain, and retrieve data attributed to carpooling rides across the city and beyond.


Backend Static Table
LOCATIONS
PK LocationID
Street
City
State
Zipcode
RATE CARD
FromCity
ToCity
Rate
DiscountPercentage
CommissionPercentage
StartLocation
EndLocation
BILLING
PK BillingID
BillAmount
DiscountAmount
ChargedAmount
PaymentMethod
CreditCardNumber
PASSENGER TRIPS
PK PassengerTripID
FK1 PassengerID
FK2 TripID
FK3 BillingID
FK4 StartLocationID
FK5 DropLocationID
COMPLAINT
PK ComplaintID
FK1 PassengerTripID
ComplaintDescription
IsResolved
SERVICE REQUEST
PK ServiceRequestID
FK1 ComplaintID
CustomerRepresentative
Status
contains
USER
PK UserID
FirstName
LastName
Gender
DOB
Email
PhoneNumber
TRIP
PK TripID
FK1 DriverID
TripType
Date
TripStatus
d
CAR
PK CarID
CarModel
CarType
RegistrationNumber
LicensePlate
Capacity
Status
DRIVER BILLING
PK DriverBillID
FK1 TripID
BillAmount
DRIVER
PK DriverID
FK1 CarID
LicenseNumber
Rating
PASSENGER
PK PassengerID
PassengerRating
drives

DESIGN

STORED PROCEDURE

```sql
---- STORED PROCEDURE 1 : CREATING A NEW USER (DRIVER OR PASSENGER) ----

CREATE PROCEDURE CreateUser
    @FirstName VARCHAR(50),
    @LastName VARCHAR(50),
    @Gender VARCHAR(25),
    @DOB DATE,
    @Email VARCHAR(100),
    @PhoneNumber VARCHAR(25),
    @Role VARCHAR(10), -- 'Driver' or 'Passenger'
    -- Optional Car details, used only for drivers
    @LicenseNumber VARCHAR(50) = NULL,
    @CarModel VARCHAR(25) = NULL,
    @CarType VARCHAR(25) = NULL,
    @RegistrationNumber VARCHAR(25) = NULL,
    @LicensePlate VARCHAR(25) = NULL,
    @Capacity INT = NULL
AS
BEGIN
    SET NOCOUNT ON;

    -- Insert into the USER table
    INSERT INTO [USER] (FirstName, LastName, Gender, DOB, Email, PhoneNumber)
    VALUES (@FirstName, @LastName, @Gender, @DOB, @Email, @PhoneNumber);

    DECLARE @NewUserID INT = SCOPE_IDENTITY();

    IF @Role = 'Driver'
    BEGIN
        -- Validate car details since they're required for drivers
        IF @CarModel IS NULL OR @CarType IS NULL OR @RegistrationNumber IS NULL OR @LicensePlate IS NULL OR @Capacity IS NULL
        BEGIN
            PRINT 'Missing car details. All car details must be provided for a driver.';
            RETURN;
        END

        -- Insert into CAR table
        INSERT INTO CAR (CarModel, CarType, RegistrationNumber, LicensePlate, Capacity, [Status])
        VALUES (@CarModel, @CarType, @RegistrationNumber, @LicensePlate, @Capacity, 'Available');

        DECLARE @NewCarID INT = SCOPE_IDENTITY();

        -- Insert into DRIVER table
        INSERT INTO DRIVER (UserID, CarID, LicenseNumber, Rating)
        VALUES (@NewUserID, @NewCarID, @LicenseNumber, DEFAULT);

        PRINT 'Driver added with UserID ' + CAST(@NewUserID AS VARCHAR) + ' and CarID ' + CAST(@NewCarID AS VARCHAR) + '.';
    END
    ELSE IF @Role = 'Passenger'
    BEGIN
        -- Insert into PASSENGER table; assuming only UserID is needed to link the passenger
        INSERT INTO PASSENGER (UserID, PassengerRating)
        VALUES (@NewUserID, DEFAULT); -- Assuming default rating is set

        PRINT 'Passenger added with UserID ' + CAST(@NewUserID AS VARCHAR) + '.';
    END
    ELSE
    BEGIN
        PRINT 'Role must be either ''Driver'' or ''Passenger''.';
    END
END;
GO
```

```sql
---- STORED PROCEDURE 3 : JOIN EXISTING TRIPS ----

CREATE PROCEDURE JoinExistingTrip
    @PassengerID INT,
    @ExistingTripID INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Variables to hold location IDs and car details
    DECLARE @TripID INT, @PassengerTripID INT, @StartLocationID INT, @DropLocationID INT, @CarID INT, @UpdatedCarCapacity INT;

    -- Retrieve Start and Drop Location IDs from the existing trip
    SELECT TOP 1 @StartLocationID = StartLocationID, @DropLocationID = DropLocationID
    FROM PASSENGERTRIPS WHERE TripID = @ExistingTripID;

    -- Retrieve CarID from the existing trip
    SELECT @CarID = d.CarID FROM DRIVER d INNER JOIN TRIP t ON d.DriverID = t.DriverID WHERE t.TripID = @ExistingTripID;

    -- Add the passenger to the existing trip
    INSERT INTO PASSENGERTRIPS (PassengerID, TripID, BillingID, StartLocationID, DropLocationID)
    VALUES (@PassengerID, @ExistingTripID, NULL, @StartLocationID, @DropLocationID);

    -- After inserting into PASSENGERTRIPS
    SET @PassengerTripID = SCOPE_IDENTITY();
    SELECT 'Passenger added to existing Trip ID: ' + CAST(@TripID AS NVARCHAR(50)) +
        ', PassengerTripID: ' + CAST(@PassengerTripID AS NVARCHAR(50)) +
        ', PassengerID: ' + CAST(@PassengerID AS NVARCHAR(50));

    -- Assume BillingID is auto-generated and linked with PassengerTripID via a trigger
    -- Now, call the procedure to update billing details for this PassengerTripID
    EXEC BillingDetails @PassengerTripID;


    -- Update Car capacity
    UPDATE CAR SET Capacity = Capacity - 1 WHERE CarID = @CarID;

    -- Check if Car capacity has reached 0, update status if necessary
    SELECT @UpdatedCarCapacity = Capacity FROM CAR WHERE CarID = @CarID;

    IF @UpdatedCarCapacity = 0
    BEGIN
        UPDATE CAR SET [Status] = 'Unavailable' WHERE CarID = @CarID;
    END
END
GO
```

```sql
---- STORED PROCEDURE 4 : CREATE A NEW TRIP ----

CREATE PROCEDURE CreateNewTrip
    @PassengerID INT,
    @TripType NVARCHAR(50),
    @CarType NVARCHAR(50),
    @StartStreet NVARCHAR(255),
    @StartCity NVARCHAR(255),
    @DropStreet NVARCHAR(255),
    @DropCity NVARCHAR(255)
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartLocationID INT, @DropLocationID INT, @TripID INT, @PassengerTripID INT, @CarID INT, @DriverID INT;

    -- Create a table variable to store output results
    DECLARE @OutputTable TABLE (
        Message NVARCHAR(255),
        TripID INT,
        PassengerTripID INT,
        PassengerID INT
    );

    -- Find Start and Drop Location IDs
    SELECT @StartLocationID = LocationID FROM LOCATIONS WHERE Street = @StartStreet AND City = @StartCity;
    SELECT @DropLocationID = LocationID FROM LOCATIONS WHERE Street = @DropStreet AND City = @DropCity;

    -- Find a Car that matches the CarType and has available capacity
    SELECT TOP 1 @CarID = CarID FROM CAR WHERE CarType = @CarType AND Capacity > 0 AND [Status] = 'Available' ORDER BY NEWID();

    -- Find an available driver for the car type
    SELECT TOP 1 @DriverID = DriverID FROM DRIVER d INNER JOIN CAR c ON d.CarID = c.CarID WHERE c.CarType = @CarType ORDER BY NEWID();

    IF @CarID IS NOT NULL --AND @DriverID IS NOT NULL
    BEGIN
        -- Insert a new Trip record
        INSERT INTO TRIP (DriverID, TripType, Date, TripStatus)
        VALUES (@DriverID, @TripType, GETDATE(), 'Ongoing');
        SET @TripID = SCOPE_IDENTITY();

        -- Insert a new PassengerTrips record
        INSERT INTO PASSENGERTRIPS (PassengerID, TripID, BillingID, StartLocationID, DropLocationID)
        VALUES (@PassengerID, @TripID, NULL, @StartLocationID, @DropLocationID);
        SET @PassengerTripID = SCOPE_IDENTITY();

        -- Insert result into @OutputTable
        INSERT INTO @OutputTable (Message, TripID, PassengerTripID, PassengerID)
        VALUES ('Passenger added to new Trip', @TripID, @PassengerTripID, @PassengerID);

        -- Assume BillingID is auto-generated and linked with PassengerTripID via a trigger
        -- Now, call the procedure to update billing details for this PassengerTripID
        EXEC BillingDetails @PassengerTripID;

        -- Update Car capacity
        UPDATE CAR SET Capacity = Capacity - 1
        WHERE CarID = @CarID;

        -- You might want to check and update the car's status if capacity reaches 0 in a similar manner
    END
    ELSE
    BEGIN
        -- Handle the case where no car or driver is available
        INSERT INTO @OutputTable (Message, TripID, PassengerTripID, PassengerID)
        VALUES ('No available cars or drivers with the specified type and capacity greater than 0.', NULL, NULL, @PassengerID);
    END

    -- Return the result set
    SELECT * FROM @OutputTable;
END
GO
```

TRIGGERS

```
---- TRIGGER 3 : CREATE BILLING ID WHEN A PASSENGER BOOKS A TRIP ----

CREATE TRIGGER trgAfterPassengerTripCreation
ON PASSENGERTRIPS
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @PassengerTripID INT;
    DECLARE @PassengerID INT;
    DECLARE @NewBillingID INT;

    -- Select the PassengerTripID and PassengerID from the inserted record
    SELECT @PassengerTripID = i.PassengerTripID, @PassengerID = i.PassengerID
    FROM inserted i;

    -- Insert a record into the BILLING table. Assume all other fields are either NULL or have a default value.
    INSERT INTO BILLING (BillAmount, DiscountAmount, ChargedAmount, PaymentMethod, CreditCardNumber)
    VALUES (NULL, NULL, NULL, NULL, NULL); -- Using DEFAULT or a specific value as required by your table design

    -- Get the ID of the newly created billing record
    SET @NewBillingID = SCOPE_IDENTITY();

    -- Update the PASSENGERTRIPS record with the new BillingID
    UPDATE PASSENGERTRIPS
    SET BillingID = @NewBillingID
    WHERE PassengerTripID = @PassengerTripID AND PassengerID = @PassengerID;
END;
GO
```

```
----- TRIGGER 1 : GENERATING SERVICE REQUEST UPON COMPLAINT RAISED -----

CREATE TRIGGER trg_AfterInsertComplaint
ON COMPLAINT
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    -- Insert a new ServiceRequest with a randomly selected Customer Representative
    INSERT INTO ServiceRequest (ComplaintID, CustomerRepresentative, [Status])
    SELECT
        ComplaintID,
        (SELECT TOP 1 CustomerRepresentative
         FROM SERVICEREQUEST
         ORDER BY NEWID()), -- Orders the table randomly and selects the top name
        'Pending'
    FROM inserted;
END;
GO
```

USER DEFINED FUNCTIONS

```sql
---- Computed Column User Define Function : 1 ----------

CREATE FUNCTION CalculateCommissionAmount (@TripID INT)
RETURNS DECIMAL(10,2)
AS
BEGIN
    DECLARE @CommissionPercentage DECIMAL(10,2);
    DECLARE @ChargedAmount DECIMAL(10,2);
    DECLARE @CommissionAmount DECIMAL(10,2);

    -- Get the commission percentage based on the start and end locations
    SELECT @CommissionPercentage = rc.CommissionPercentage
    FROM RATECARD rc
    INNER JOIN LOCATIONS locStart ON rc.FromCity = locStart.City
    INNER JOIN LOCATIONS locEnd ON rc.ToCity = locEnd.City
    INNER JOIN PASSENGERTRIPS pt ON locStart.LocationID = pt.StartLocationID AND locEnd.LocationID = pt.DropLocationID
    WHERE pt.TripID = @TripID;

    -- Get the charged amount for the trip from the billing table
    SELECT @ChargedAmount = ChargedAmount
    FROM BILLING
    WHERE BillingID = (SELECT BillingID FROM PASSENGERTRIPS WHERE TripID = @TripID);

    -- Calculate the commission amount
    SET @CommissionAmount = (@CommissionPercentage / 100) * @ChargedAmount;

    -- Return the commission amount
    RETURN @CommissionAmount;
END;
GO

ALTER TABLE DriverBilling
ADD CommissionAmount AS dbo.CalculateCommissionAmount(TripID);
```

```sql
---- Computed Column User Define Function : 2 -------

GO
CREATE FUNCTION CalculateDriverBilling (@TripID INT)
RETURNS DECIMAL(10,2)
AS
BEGIN
    DECLARE @ChargedAmount DECIMAL(10,2);

    -- Get the charged amount for the trip from the billing table
    SELECT @ChargedAmount = ChargedAmount
    FROM BILLING
    WHERE BillingID = (SELECT BillingID FROM PASSENGERTRIPS WHERE TripID = @TripID);

    -- Calculate the commission amount
    DECLARE @CommissionAmount DECIMAL(10,2);
    SET @CommissionAmount = dbo.CalculateCommissionAmount(@TripID);

    -- Return the total driver billing amount
    RETURN @ChargedAmount - @CommissionAmount;
END;
GO

ALTER TABLE DriverBilling
ADD TotalDriverPay AS dbo.CalculateDriverBilling(TripID);

SELECT * FROM DriverBilling
```

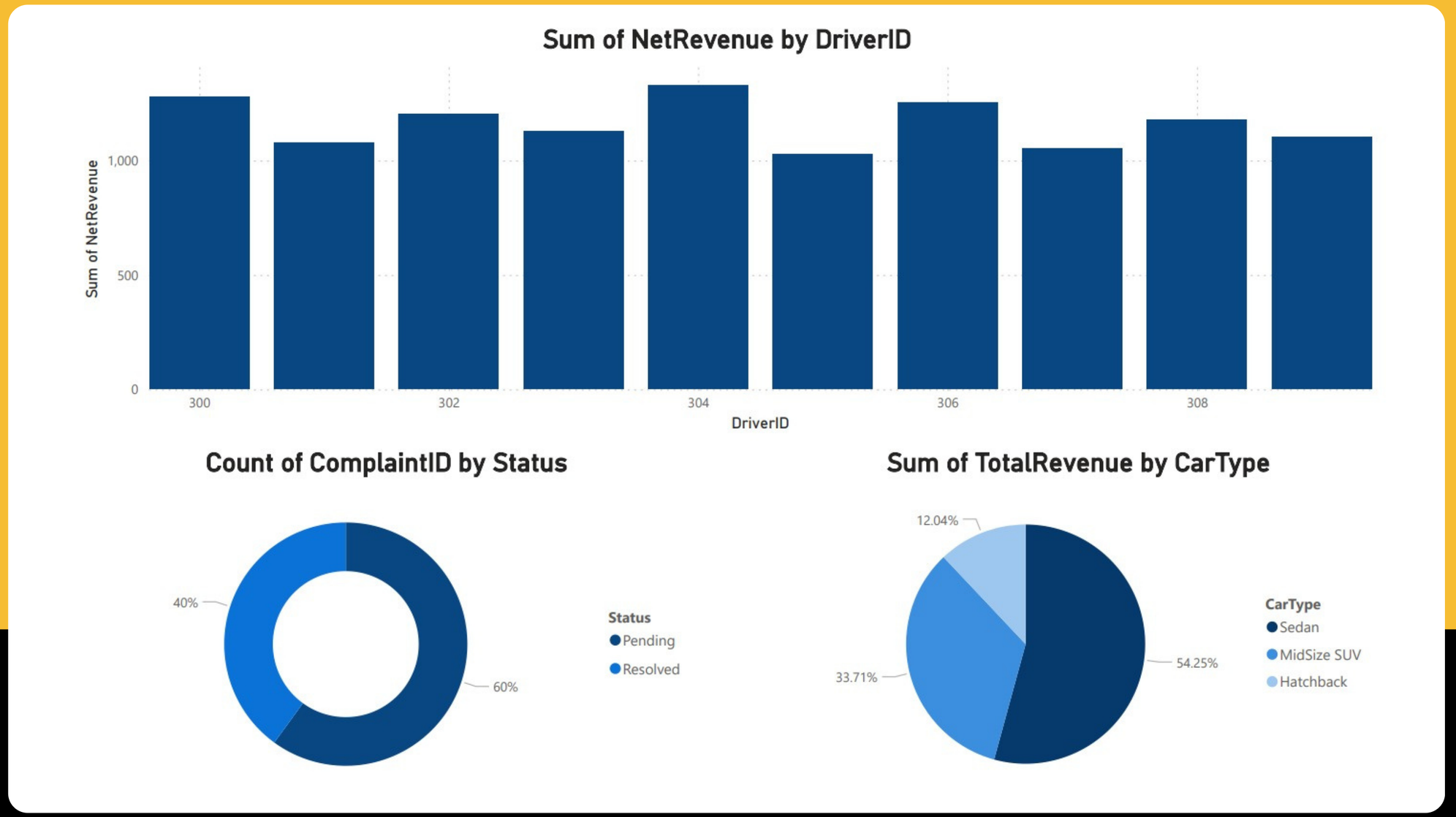
VIEWS

```
---- VIEW 1: REVENUE BY CAR TYPE ----
--Aggregates revenue generated, segmented by car type.
CREATE VIEW RevenueByCarType AS
SELECT
    c.CarType,
    SUM(b.ChargedAmount) AS TotalRevenue
FROM Car c
JOIN DRIVER d ON c.CarID = d.CarID
JOIN Trip t ON d.DriverID = t.DriverID
JOIN PassengerTrips pt ON t.TripID = pt.TripID
JOIN Billing b ON pt.BillingID = b.BillingID
GROUP BY c.CarType;
GO

SELECT * FROM RevenueByCarType;
GO
```

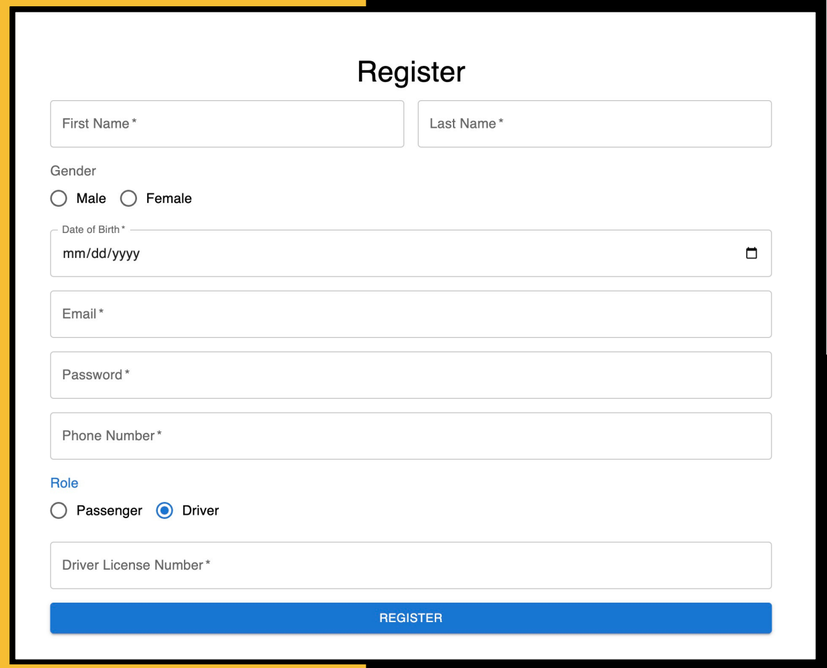
```
---- VIEW 5: DRIVER PERFORMANC SUMMARY ----
--This view aggregates key performance indicators for drivers, such as ave
--It's useful for evaluating driver performance and identifying top or und

CREATE VIEW DriverPerformanceSummary AS
SELECT
    d.DriverID,
    u.FirstName + ' ' + u.LastName AS DriverName,
    AVG(d.Rating) AS AverageRating,
    COUNT(t.TripID) AS TotalTrips,
    AVG(db.CommissionAmount) AS AverageCommissionEarned
FROM DRIVER d
JOIN [USER] u ON d.UserID = u.UserID
JOIN Trip t ON d.DriverID = t.DriverID
JOIN DRIVERBILLING db ON t.TripID = db.TripID
GROUP BY d.DriverID, u.FirstName, u.LastName;
GO
```

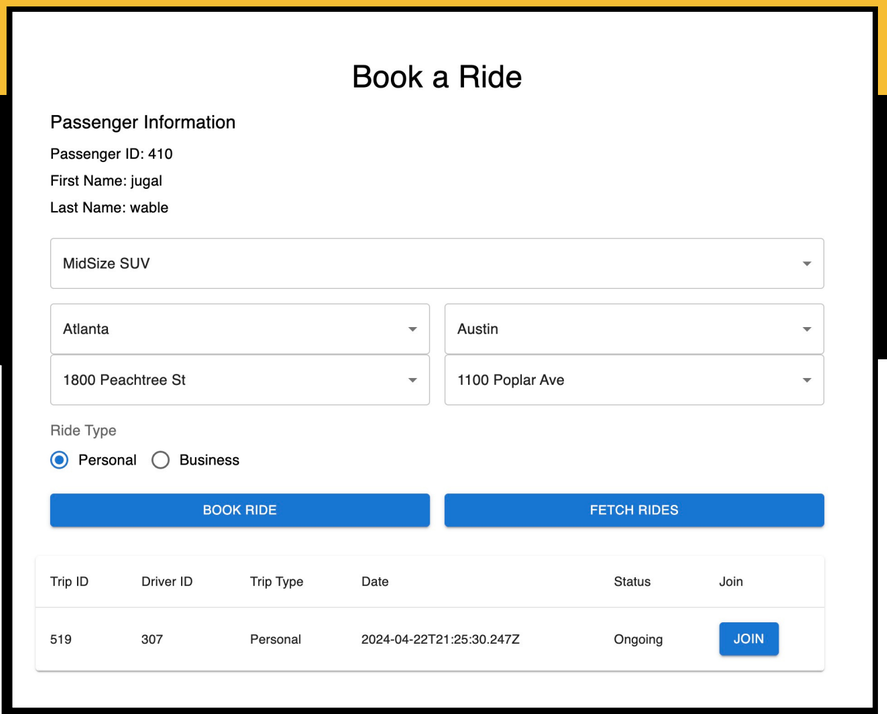
```
--View 3: ServiceRequestOverview
GO
--This view provides an overview of service requests, including complaint details,
--and the customer service representative assigned to the case. It's useful for tra
CREATE VIEW ServiceRequestOverview AS
SELECT
    sr.ServiceRequestID,
    sr.ComplaintID,
    c.PassengerTripID,
    sr.CustomerRepresentative,
    sr.[Status],
    c.ComplaintDescription,
    c.IsResolved
FROM SERVICEREQUEST sr
JOIN COMPLAINT c ON sr.ComplaintID = c.ComplaintID;

GO
SELECT * FROM ServiceRequestOverview;
```

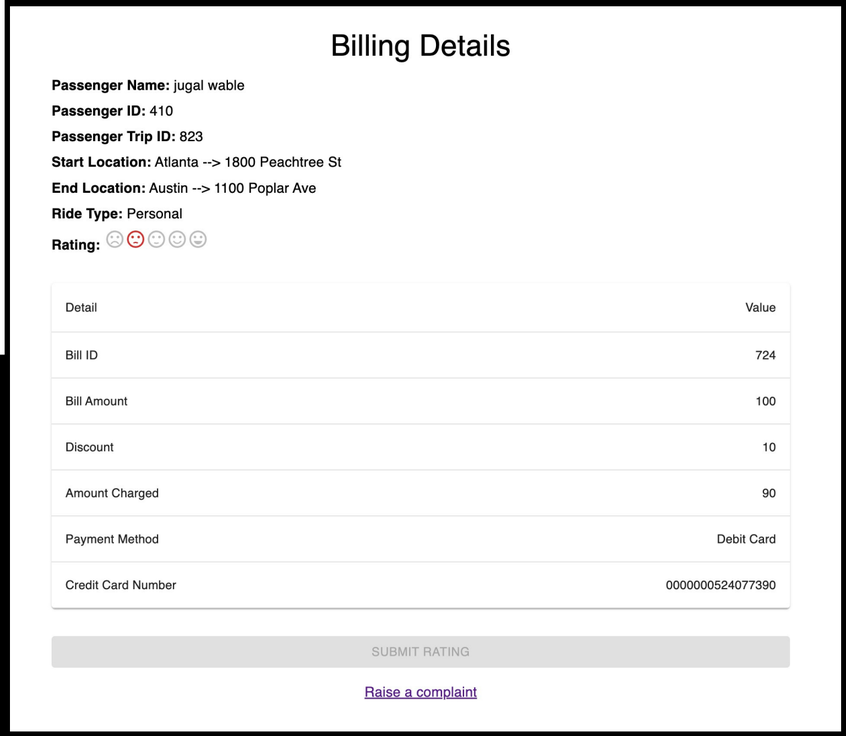
DASHBOARD


Sum of NetRevenue by DriverID
Sum of NetRevenue
1,000
500
0
300
302
304
306
308
DriverID
Count of ComplaintID by Status
40%
60%
Status
Pending
Resolved
Sum of TotalRevenue by CarType
12.04%
54.25%
33.71%
CarType
Sedan
MidSize SUV
Hatchback

GUI - DEMO


Register
First Name *
Last Name *
Gender
Male
Female
Date of Birth *
mm/dd/yyyy
Email *
Password *
Phone Number *
Role
Passenger
Driver
Driver License Number *
REGISTER


Book a Ride
Passenger Information
Passenger ID: 410
First Name: jugal
Last Name: wable
MidSize SUV
Atlanta
Austin
1800 Peachtree St
1100 Poplar Ave
Ride Type
Personal
Business
BOOK RIDE
FETCH RIDES
Trip ID
Driver ID
Trip Type
Date
Status
Join
519
307
Personal
2024-04-22T21:25:30.247Z
Ongoing
JOIN


Billing Details
Passenger Name: jugal wable
Passenger ID: 410
Passenger Trip ID: 823
Start Location: Atlanta --> 1800 Peachtree St
End Location: Austin --> 1100 Poplar Ave
Ride Type: Personal
Rating:
Detail
Value
Bill ID
724
Bill Amount
100
Discount
10
Amount Charged
90
Payment Method
Debit Card
Credit Card Number
0000000524077390
SUBMIT RATING
Raise a complaint

THANK YOU

Enjoy Your Ride!