Car Rental Database Schema - CV Portfolio Project

Core Entity Tables

Addresses

```sql
CREATE TABLE Addresses (
    AddressID INT IDENTITY(1,1) PRIMARY KEY,
    StreetAddress NVARCHAR(200) NOT NULL,
    City NVARCHAR(50) NOT NULL,
    StateProvince NVARCHAR(50) NOT NULL,
    ZipCode NVARCHAR(15) NOT NULL,
    Country NVARCHAR(50) NOT NULL DEFAULT 'USA',
    Latitude DECIMAL(9, 6), -- For map markers: -90 to 90
    Longitude DECIMAL(9, 6), -- For map markers: -180 to 180
    AddressType NVARCHAR(20) DEFAULT 'Primary'
        CHECK (AddressType IN ('Primary', 'Billing', 'Business', 'Location')),
    IsActive BIT DEFAULT 1,
    CreatedDate DATETIME2 DEFAULT GETDATE()
);
```

Persons (Base table for all users)

```sql
```

```sql
CREATE TABLE Persons (
    PersonID INT IDENTITY(1,1) PRIMARY KEY,
    FirstName NVARCHAR(50) NOT NULL,
    LastName NVARCHAR(50) NOT NULL,
    Email NVARCHAR(100) UNIQUE NOT NULL,
    Phone NVARCHAR(20) NOT NULL,
    DateOfBirth DATE,
    NationalID NVARCHAR(30),
    PrimaryAddressID INT,
    ProfileImage NVARCHAR(255), -- Path to profile image
    CreatedDate DATETIME2 DEFAULT GETDATE(),
    IsActive BIT DEFAULT 1,

    CONSTRAINT FK_Persons_PrimaryAddress FOREIGN KEY (PrimaryAddressID)
        REFERENCES Addresses(AddressID)
);
```

Users (System users - Employees/Admins)

sql

```sql
CREATE TABLE Users (
    UserID INT IDENTITY(1,1) PRIMARY KEY,
    PersonID INT NOT NULL UNIQUE,
    Username NVARCHAR(50) UNIQUE NOT NULL,
    PasswordHash NVARCHAR(255) NOT NULL,
    UserRole NVARCHAR(20) NOT NULL
        CHECK (UserRole IN ('Admin', 'Manager', 'Employee')),
    IsActive BIT DEFAULT 1,
    LastLogin DATETIME2,
    CreatedDate DATETIME2 DEFAULT GETDATE(),

    CONSTRAINT FK_Users_Person FOREIGN KEY (PersonID)
        REFERENCES Persons(PersonID)
);
```

Customers (Rental customers)

sql

```
CREATE TABLE Customers (
    CustomerID INT IDENTITY(1,1) PRIMARY KEY,
    PersonID INT NOT NULL UNIQUE,
    DriverLicenseNumber NVARCHAR(30) UNIQUE NOT NULL,
    LicenseExpiryDate DATE,
    CustomerType NVARCHAR(20) DEFAULT 'Individual'
        CHECK (CustomerType IN ('Individual', 'Corporate')),
    TotalRentals INT DEFAULT 0,
    RegistrationDate DATETIME2 DEFAULT GETDATE(),
    IsActive BIT DEFAULT 1,

    CONSTRAINT FK_Customers_Person FOREIGN KEY (PersonID)
        REFERENCES Persons(PersonID)
);
```

RentalLocations (Branches)

sql

```
CREATE TABLE RentalLocations (
    LocationID INT IDENTITY(1,1) PRIMARY KEY,
    LocationName NVARCHAR(100) NOT NULL,
    LocationCode NVARCHAR(10) UNIQUE NOT NULL,
    AddressID INT NOT NULL,
    Phone NVARCHAR(20),
    Email NVARCHAR(100),
    ManagerUserID INT,
    ParkingCapacity INT DEFAULT 0,
    IsActive BIT DEFAULT 1,
    CreatedDate DATETIME2 DEFAULT GETDATE(),

    CONSTRAINT FK_RentalLocations_Address FOREIGN KEY (AddressID)
        REFERENCES Addresses(AddressID),
    CONSTRAINT FK_RentalLocations_Manager FOREIGN KEY (ManagerUserID)
        REFERENCES Users(UserID)
);
```

VehicleCategories

sql

```sql
CREATE TABLE VehicleCategories (
    CategoryID INT IDENTITY(1,1) PRIMARY KEY,
    CategoryName NVARCHAR(50) NOT NULL UNIQUE,
    Description NVARCHAR(200),
    PassengerCapacity TINYINT,
    IsActive BIT DEFAULT 1
);
```

FuelTypes

sql

```sql
CREATE TABLE FuelTypes (
    FuelTypeID INT IDENTITY(1,1) PRIMARY KEY,
    FuelTypeName NVARCHAR(30) NOT NULL UNIQUE,
    IsActive BIT DEFAULT 1
);
```

Vehicles

sql

```sql
CREATE TABLE Vehicles (
    VehicleID INT IDENTITY(1,1) PRIMARY KEY,
    Make NVARCHAR(50) NOT NULL,
    Model NVARCHAR(50) NOT NULL,
    Year SMALLINT NOT NULL CHECK (Year >= 1900 AND Year <= YEAR(GETDATE()) + 1),
    PlateNumber NVARCHAR(20) NOT NULL UNIQUE,
    VIN NVARCHAR(17) UNIQUE,
    CategoryID INT NOT NULL,
    FuelTypeID INT NOT NULL,
    CurrentMileage INT DEFAULT 0,
    RentalPricePerDay DECIMAL(8,2) NOT NULL CHECK (RentalPricePerDay >= 0),
    Color NVARCHAR(30),
    TransmissionType NVARCHAR(20) CHECK (TransmissionType IN ('Manual', 'Automatic')),
    CurrentLocationID INT, -- Which branch is the car at
    IsAvailableForRent BIT DEFAULT 1,
    VehicleStatus NVARCHAR(20) DEFAULT 'Available'
        CHECK (VehicleStatus IN ('Available', 'Rented', 'Maintenance', 'Retired')),
    VehicleImage NVARCHAR(255), -- Path to vehicle image
    CreatedDate DATETIME2 DEFAULT GETDATE(),
    LastUpdated DATETIME2 DEFAULT GETDATE(),

    CONSTRAINT FK_Vehicles_Category FOREIGN KEY (CategoryID)
        REFERENCES VehicleCategories(CategoryID),
    CONSTRAINT FK_Vehicles_FuelType FOREIGN KEY (FuelTypeID)
        REFERENCES FuelTypes(FuelTypeID),
    CONSTRAINT FK_Vehicles_CurrentLocation FOREIGN KEY (CurrentLocationID)
        REFERENCES RentalLocations(LocationID)
);
```

Rental Operations Tables

RentalBookings

sql

```
CREATE TABLE RentalBookings (
    BookingID INT IDENTITY(1,1) PRIMARY KEY,
    BookingNumber AS ('BK' + RIGHT('000000' + CAST(BookingID AS VARCHAR), 6)) PERSISTED,
    CustomerID INT NOT NULL,
    VehicleID INT NOT NULL,
    RentalStartDate DATETIME2 NOT NULL,
    RentalEndDate DATETIME2 NOT NULL,
    PickupLocationID INT NOT NULL,
    DropoffLocationID INT NOT NULL,
    InitialRentalDays AS (DATEDIFF(DAY, RentalStartDate, RentalEndDate)) PERSISTED,
    RentalPricePerDay DECIMAL(8,2) NOT NULL,
    InitialTotalAmount AS (DATEDIFF(DAY, RentalStartDate, RentalEndDate) * RentalPricePerDay) PERSISTED,
    BookingStatus NVARCHAR(20) DEFAULT 'Confirmed'
        CHECK (BookingStatus IN ('Pending', 'Confirmed', 'Active', 'Completed', 'Cancelled')),
    InitialCheckNotes NVARCHAR(500),
    InitialMileage INT,
    BookingDate DATETIME2 DEFAULT GETDATE(),
    ProcessedByUserID INT,

    CONSTRAINT FK_RentalBookings_Customer FOREIGN KEY (CustomerID)
        REFERENCES Customers(CustomerID),
    CONSTRAINT FK_RentalBookings_Vehicle FOREIGN KEY (VehicleID)
        REFERENCES Vehicles(VehicleID),
    CONSTRAINT FK_RentalBookings_PickupLocation FOREIGN KEY (PickupLocationID)
        REFERENCES RentalLocations(LocationID),
    CONSTRAINT FK_RentalBookings_DropoffLocation FOREIGN KEY (DropoffLocationID)
        REFERENCES RentalLocations(LocationID),
    CONSTRAINT FK_RentalBookings_ProcessedBy FOREIGN KEY (ProcessedByUserID)
        REFERENCES Users(UserID),
    CONSTRAINT CK_RentalBookings_DateRange CHECK (RentalEndDate > RentalStartDate)
);
```

VehicleReturns

sql

```sql
CREATE TABLE VehicleReturns (
    ReturnID INT IDENTITY(1,1) PRIMARY KEY,
    BookingID INT NOT NULL UNIQUE,
    ActualReturnDate DATETIME2 NOT NULL,
    ActualReturnLocationID INT NOT NULL,
    ActualRentalDays AS (DATEDIFF(DAY,
        (SELECT RentalStartDate FROM RentalBookings WHERE BookingID = VehicleReturns.BookingID),
        ActualReturnDate)) PERSISTED,
    ReturnMileage INT NOT NULL,
    ConsumedMileage AS (ReturnMileage -
        (SELECT InitialMileage FROM RentalBookings WHERE BookingID = VehicleReturns.BookingID)) PERSISTED,
    FinalCheckNotes NVARCHAR(500),
    VehicleCondition NVARCHAR(20) DEFAULT 'Good'
        CHECK (VehicleCondition IN ('Excellent', 'Good', 'Fair', 'Damaged')),
    AdditionalCharges DECIMAL(8,2) DEFAULT 0,
    LateFees DECIMAL(8,2) DEFAULT 0,
    DamageFees DECIMAL(8,2) DEFAULT 0,
    MileageCharges DECIMAL(8,2) DEFAULT 0,
    ActualTotalDueAmount DECIMAL(10,2),
    ProcessedByUserID INT,
    CreatedDate DATETIME2 DEFAULT GETDATE(),

    CONSTRAINT FK_VehicleReturns_Booking FOREIGN KEY (BookingID)
        REFERENCES RentalBookings(BookingID),
    CONSTRAINT FK_VehicleReturns_ReturnLocation FOREIGN KEY (ActualReturnLocationID)
        REFERENCES RentalLocations(LocationID),
    CONSTRAINT FK_VehicleReturns_ProcessedBy FOREIGN KEY (ProcessedByUserID)
        REFERENCES Users(UserID)
);
```

RentalTransactions

```
sql
```

```sql
CREATE TABLE RentalTransactions (
    TransactionID INT IDENTITY(1,1) PRIMARY KEY,
    TransactionNumber AS ('TXN' + RIGHT('000000' + CAST(TransactionID AS VARCHAR), 6)) PERSISTED,
    BookingID INT NOT NULL,
    PaymentDate DATETIME2 DEFAULT GETDATE(),
    PaymentMethod NVARCHAR(30)
        CHECK (PaymentMethod IN ('Cash', 'Credit Card', 'Debit Card', 'Bank Transfer')),
    Amount DECIMAL(10,2) NOT NULL,
    Notes NVARCHAR(300),
    ProcessedByUserID INT,

    CONSTRAINT FK_RentalTransactions_Booking FOREIGN KEY (BookingID)
        REFERENCES RentalBookings(BookingID),
    CONSTRAINT FK_RentalTransactions_ProcessedBy FOREIGN KEY (ProcessedByUserID)
        REFERENCES Users(UserID)
);
```

Maintenance Table

MaintenanceRecords

```sql
CREATE TABLE MaintenanceRecords (
    MaintenanceID INT IDENTITY(1,1) PRIMARY KEY,
    VehicleID INT NOT NULL,
    MaintenanceType NVARCHAR(50) NOT NULL,
    Description NVARCHAR(500) NOT NULL,
    MaintenanceDate DATE NOT NULL,
    Cost DECIMAL(8,2) DEFAULT 0,
    ServiceProvider NVARCHAR(100),
    MileageAtService INT,
    ScheduledByUserID INT,
    CreatedDate DATETIME2 DEFAULT GETDATE(),

    CONSTRAINT FK_MaintenanceRecords_Vehicle FOREIGN KEY (VehicleID)
        REFERENCES Vehicles(VehicleID),
    CONSTRAINT FK_MaintenanceRecords_ScheduledBy FOREIGN KEY (ScheduledByUserID)
        REFERENCES Users(UserID)
);
```

Indexes for Performance

```
sql

-- Address indexes
CREATE INDEX IX_Addresses_City ON Addresses(City);
CREATE INDEX IX_Addresses_Coordinates ON Addresses(Latitude, Longitude) WHERE Latitude IS NOT NULL;

-- Person indexes
CREATE INDEX IX_Persons_Email ON Persons(Email);
CREATE INDEX IX_Persons_Name ON Persons(LastName, FirstName);

-- User indexes
CREATE INDEX IX_Users_Username ON Users(Username);
CREATE INDEX IX_Users_Role ON Users(UserRole);

-- Customer indexes
CREATE INDEX IX_Customers_DriverLicense ON Customers(DriverLicenseNumber);

-- Vehicle indexes
CREATE INDEX IX_Vehicles_PlateNumber ON Vehicles(PlateNumber);
CREATE INDEX IX_Vehicles_Status ON Vehicles(VehicleStatus, IsAvailableForRent);
CREATE INDEX IX_Vehicles_Category ON Vehicles(CategoryID);
CREATE INDEX IX_Vehicles_Location ON Vehicles(CurrentLocationID);

-- Booking indexes
CREATE INDEX IX_RentalBookings_Customer ON RentalBookings(CustomerID);
CREATE INDEX IX_RentalBookings_Vehicle ON RentalBookings(VehicleID);
CREATE INDEX IX_RentalBookings_Dates ON RentalBookings(RentalStartDate, RentalEndDate);
CREATE INDEX IX_RentalBookings_Status ON RentalBookings(BookingStatus);
CREATE INDEX IX_RentalBookings_Locations ON RentalBookings(PickupLocationID, DropoffLocationID);

-- Transaction indexes
CREATE INDEX IX_RentalTransactions_Booking ON RentalTransactions(BookingID);
CREATE INDEX IX_RentalTransactions_Date ON RentalTransactions(PaymentDate);
```

Views for Windows Forms & Map Display

Map View - Rental Locations

```
sql
```

```sql
CREATE VIEW vw_RentalLocationsMap AS
SELECT
    rl.LocationID,
    rl.LocationName,
    rl.LocationCode,
    a.City,
    a.StateProvince,
    a.Latitude,
    a.Longitude,
    a.StreetAddress + ', ' + a.City + ', ' + a.StateProvince + ' ' + a.ZipCode AS FullAddress,
    rl.Phone,
    rl.ParkingCapacity,
    COUNT(v.VehicleID) AS VehiclesAtLocation,
    SUM(CASE WHEN v.IsAvailableForRent = 1 THEN 1 ELSE 0 END) AS AvailableVehicles,
    u.FirstName + ' ' + u.LastName AS ManagerName
FROM RentalLocations rl
    INNER JOIN Addresses a ON rl.AddressID = a.AddressID
    LEFT JOIN Vehicles v ON rl.LocationID = v.CurrentLocationID AND v.VehicleStatus != 'Retired'
    LEFT JOIN Users usr ON rl.ManagerUserID = usr.UserID
    LEFT JOIN Persons u ON usr.PersonID = u.PersonID
WHERE rl.IsActive = 1
GROUP BY rl.LocationID, rl.LocationName, rl.LocationCode, a.City, a.StateProvince,
         a.Latitude, a.Longitude, a.StreetAddress, a.ZipCode, rl.Phone,
         rl.ParkingCapacity, u.FirstName, u.LastName;
```

Map View - Available Vehicles by Location

```
sql
```

```sql
CREATE VIEW vw_AvailableVehiclesByLocation AS
SELECT
    v.VehicleID,
    v.Make + ' ' + v.Model + ' (' + CAST(v.Year AS VARCHAR) + ')' AS VehicleDisplay,
    v.PlateNumber,
    vc.CategoryName,
    ft.FuelTypeName,
    v.RentalPricePerDay,
    v.Color,
    rl.LocationID,
    rl.LocationName,
    rl.LocationCode,
    a.Latitude,
    a.Longitude,
    a.City,
    a.StateProvince,
    a.StreetAddress + ', ' + a.City + ', ' + a.StateProvince AS LocationAddress
FROM Vehicles v
    INNER JOIN VehicleCategories vc ON v.CategoryID = vc.CategoryID
    INNER JOIN FuelTypes ft ON v.FuelTypeID = ft.FuelTypeID
    INNER JOIN RentalLocations rl ON v.CurrentLocationID = rl.LocationID
    INNER JOIN Addresses a ON rl.AddressID = a.AddressID
WHERE v.IsAvailableForRent = 1
  AND v.VehicleStatus = 'Available';
```

User Management View

```
sql
```

```
CREATE VIEW vw_UserDetails AS
SELECT
    u.UserID,
    u.Username,
    u.UserRole,
    u.IsActive AS UserIsActive,
    u.LastLogin,
    p.PersonID,
    p.FirstName,
    p.LastName,
    p.FirstName + ' ' + p.LastName AS FullName,
    p.Email,
    p.Phone,
    a.City,
    a.StateProvince
FROM Users u
    INNER JOIN Persons p ON u.PersonID = p.PersonID
    LEFT JOIN Addresses a ON p.PrimaryAddressID = a.AddressID;
```

Customer Details View

sql

```sql
CREATE VIEW vw_CustomerDetails AS
SELECT
    c.CustomerID,
    c.DriverLicenseNumber,
    c.CustomerType,
    c.TotalRentals,
    c.RegistrationDate,
    p.PersonID,
    p.FirstName,
    p.LastName,
    p.FirstName + ' ' + p.LastName AS FullName,
    p.Email,
    p.Phone,
    p.DateOfBirth,
    a.StreetAddress,
    a.City,
    a.StateProvince,
    a.ZipCode,
    a.StreetAddress + ', ' + a.City + ', ' + a.StateProvince + ' ' + a.ZipCode AS FullAddress,
    c.IsActive
FROM Customers c
    INNER JOIN Persons p ON c.PersonID = p.PersonID
    LEFT JOIN Addresses a ON p.PrimaryAddressID = a.AddressID;
```

Rental Dashboard View

sql

```sql
CREATE VIEW vw_RentalDashboard AS
SELECT
    rb.BookingID,
    rb.BookingNumber,
    cust.FirstName + ' ' + cust.LastName AS CustomerName,
    cust.Phone AS CustomerPhone,
    v.Make + ' ' + v.Model AS VehicleName,
    v.PlateNumber,
    rb.RentalStartDate,
    rb.RentalEndDate,
    rb.InitialRentalDays,
    rb.InitialTotalAmount,
    pickup.LocationName AS PickupLocation,
    dropoff.LocationName AS DropoffLocation,
    rb.BookingStatus,
    CASE
        WHEN rb.BookingStatus = 'Completed' THEN 'Completed'
        WHEN rb.BookingStatus = 'Cancelled' THEN 'Cancelled'
        WHEN GETDATE() < rb.RentalStartDate THEN 'Upcoming'
        WHEN GETDATE() BETWEEN rb.RentalStartDate AND rb.RentalEndDate THEN 'Active'
        WHEN GETDATE() > rb.RentalEndDate THEN 'Overdue'
    END AS RentalStatus,
    processUser.FirstName + ' ' + processUser.LastName AS ProcessedBy
FROM RentalBookings rb
    INNER JOIN Customers c ON rb.CustomerID = c.CustomerID
    INNER JOIN Persons cust ON c.PersonID = cust.PersonID
    INNER JOIN Vehicles v ON rb.VehicleID = v.VehicleID
    INNER JOIN RentalLocations pickup ON rb.PickupLocationID = pickup.LocationID
    INNER JOIN RentalLocations dropoff ON rb.DropoffLocationID = dropoff.LocationID
    LEFT JOIN Users pu ON rb.ProcessedByUserID = pu.UserID
    LEFT JOIN Persons processUser ON pu.PersonID = processUser.PersonID;
```

Financial Summary View

```
sql
```

```
CREATE VIEW vw_FinancialSummary AS
SELECT
    rb.BookingID,
    rb.BookingNumber,
    c.FirstName + ' ' + c.LastName AS CustomerName,
    rb.InitialTotalAmount,
    ISNULL(vr.ActualTotalDueAmount, rb.InitialTotalAmount) AS TotalDue,
    ISNULL(SUM(rt.Amount), 0) AS TotalPaid,
    ISNULL(vr.ActualTotalDueAmount, rb.InitialTotalAmount) - ISNULL(SUM(rt.Amount), 0) AS Balance,
    rb.BookingStatus
FROM RentalBookings rb
    INNER JOIN Customers cu ON rb.CustomerID = cu.CustomerID
    INNER JOIN Persons c ON cu.PersonID = c.PersonID
    LEFT JOIN VehicleReturns vr ON rb.BookingID = vr.BookingID
    LEFT JOIN RentalTransactions rt ON rb.BookingID = rt.BookingID
GROUP BY rb.BookingID, rb.BookingNumber, c.FirstName, c.LastName,
         rb.InitialTotalAmount, vr.ActualTotalDueAmount, rb.BookingStatus;
```

Stored Procedures

Get Available Vehicles at Location

```
sql
```

```sql
CREATE PROCEDURE sp_GetAvailableVehiclesAtLocation
    @LocationID INT = NULL,
    @CategoryID INT = NULL
AS
BEGIN
    SELECT
        v.VehicleID,
        v.Make + ' ' + v.Model + ' (' + CAST(v.Year AS VARCHAR) + ')' AS VehicleDisplay,
        v.PlateNumber,
        v.Color,
        v.RentalPricePerDay,
        vc.CategoryName,
        ft.FuelTypeName,
        rl.LocationName
    FROM Vehicles v
        INNER JOIN VehicleCategories vc ON v.CategoryID = vc.CategoryID
        INNER JOIN FuelTypes ft ON v.FuelTypeID = ft.FuelTypeID
        LEFT JOIN RentalLocations rl ON v.CurrentLocationID = rl.LocationID
    WHERE v.IsAvailableForRent = 1
        AND v.VehicleStatus = 'Available'
        AND (@LocationID IS NULL OR v.CurrentLocationID = @LocationID)
        AND (@CategoryID IS NULL OR v.CategoryID = @CategoryID)
    ORDER BY v.RentalPricePerDay;
END;
GO
```

Create New Booking

```
sql
```

```
CREATE PROCEDURE sp_CreateBooking
    @CustomerID INT,
    @VehicleID INT,
    @RentalStartDate DATETIME2,
    @RentalEndDate DATETIME2,
    @PickupLocationID INT,
    @DropoffLocationID INT,
    @InitialMileage INT,
    @ProcessedByUserID INT,
    @BookingID INT OUTPUT
AS
BEGIN
    BEGIN TRANSACTION;

    -- Check if vehicle is available
    IF NOT EXISTS (
        SELECT 1 FROM Vehicles
        WHERE VehicleID = @VehicleID
        AND IsAvailableForRent = 1
        AND VehicleStatus = 'Available'
    )
    BEGIN
        ROLLBACK;
        THROW 50001, 'Vehicle is not available for rent', 1;
    END

    -- Get rental price
    DECLARE @RentalPricePerDay DECIMAL(8,2);
    SELECT @RentalPricePerDay = RentalPricePerDay
    FROM Vehicles
    WHERE VehicleID = @VehicleID;

    -- Create booking
    INSERT INTO RentalBookings (
        CustomerID, VehicleID, RentalStartDate, RentalEndDate,
        PickupLocationID, DropoffLocationID, InitialMileage,
        RentalPricePerDay, BookingStatus, ProcessedByUserID
    )
    VALUES (
        @CustomerID, @VehicleID, @RentalStartDate, @RentalEndDate,
        @PickupLocationID, @DropoffLocationID, @InitialMileage,
        @RentalPricePerDay, 'Confirmed', @ProcessedByUserID
    );
    SET @BookingID = SCOPE_IDENTITY();

    COMMIT;
END;
GO
```

Triggers

Update Vehicle Status on Booking

sql

```
CREATE TRIGGER trg_UpdateVehicleOnBooking
ON RentalBookings
AFTER INSERT, UPDATE
AS
BEGIN
    -- Mark vehicle as Rented when booking becomes Active
    UPDATE v
    SET VehicleStatus = 'Rented',
        IsAvailableForRent = 0,
        LastUpdated = GETDATE()
    FROM Vehicles v
        INNER JOIN inserted i ON v.VehicleID = i.VehicleID
    WHERE i.BookingStatus = 'Active';
END;
GO
```

Update Vehicle Status on Return

sql

```sql
CREATE TRIGGER trg_UpdateVehicleOnReturn
ON VehicleReturns
AFTER INSERT
AS
BEGIN
    UPDATE v
    SET CurrentMileage = i.ReturnMileage,
        CurrentLocationID = i.ActualReturnLocationID,
        VehicleStatus = 'Available',
        IsAvailableForRent = 1,
        LastUpdated = GETDATE()
    FROM Vehicles v
        INNER JOIN RentalBookings rb ON v.VehicleID = rb.VehicleID
        INNER JOIN inserted i ON rb.BookingID = i.BookingID;

    -- Update booking status
    UPDATE rb
    SET BookingStatus = 'Completed'
    FROM RentalBookings rb
        INNER JOIN inserted i ON rb.BookingID = i.BookingID;
END;
GO
```

Update Customer Total Rentals

sql

```sql
CREATE TRIGGER trg_UpdateCustomerStats
ON RentalBookings
AFTER INSERT
AS
BEGIN
    UPDATE c
    SET TotalRentals = TotalRentals + 1
    FROM Customers c
        INNER JOIN inserted i ON c.CustomerID = i.CustomerID
    WHERE i.BookingStatus IN ('Confirmed', 'Active');
END;
GO
```

Sample Data for Testing

```sql
-- Insert Fuel Types
INSERT INTO FuelTypes (FuelTypeName) VALUES
('Gasoline'), ('Diesel'), ('Electric'), ('Hybrid');

-- Insert Vehicle Categories
INSERT INTO VehicleCategories (CategoryName, PassengerCapacity) VALUES
('Economy', 4), ('Compact', 5), ('SUV', 7), ('Luxury', 5), ('Van', 8);

-- Insert Sample Address
INSERT INTO Addresses (StreetAddress, City, StateProvince, ZipCode, Country, Latitude, Longitude, AddressType)
VALUES
('123 Main St', 'Los Angeles', 'California', '90001', 'USA', 34.0522, -118.2437, 'Location'),
('456 Broadway', 'New York', 'New York', '10001', 'USA', 40.7128, -74.0060, 'Location'),
('789 Lake Shore Dr', 'Chicago', 'Illinois', '60601', 'USA', 41.8781, -87.6298, 'Location');

-- Insert Rental Locations
INSERT INTO RentalLocations (LocationName, LocationCode, AddressID, Phone, Email, ParkingCapacity)
VALUES
('Downtown LA Branch', 'LA01', 1, '555-0001', 'la@carrental.com', 50),
('Manhattan Branch', 'NY01', 2, '555-0002', 'ny@carrental.com', 30),
('Chicago Loop Branch', 'CH01', 3, '555-0003', 'ch@carrental.com', 40);
```

Key Features for CV Portfolio:

1. Person-Based Architecture

- Demonstrates normalization skills
- Single source for all user types
- Reduces data redundancy

2. Map Integration

- Latitude/Longitude in Addresses table
- Views optimized for map display
- Location-based vehicle tracking

3. Role-Based Access

- Users table for employees/admins
- Customers table for rental customers
- Proper separation of concerns

4. Business Logic

- Computed columns (BookingNumber, rental days, totals)
- Triggers for automatic updates
- Stored procedures for complex operations

5. Professional Features

- Audit trails (CreatedDate, ProcessedBy)
- Status management
- Financial tracking
- Maintenance records

6. Windows Forms Ready

- Multiple views for data binding
- Optimized for ComboBox/DataGridView
- Map-friendly data structure

This schema is perfect for showcasing in your CV - it demonstrates database design skills, normalization, business logic implementation, and real-world application development capabilities!