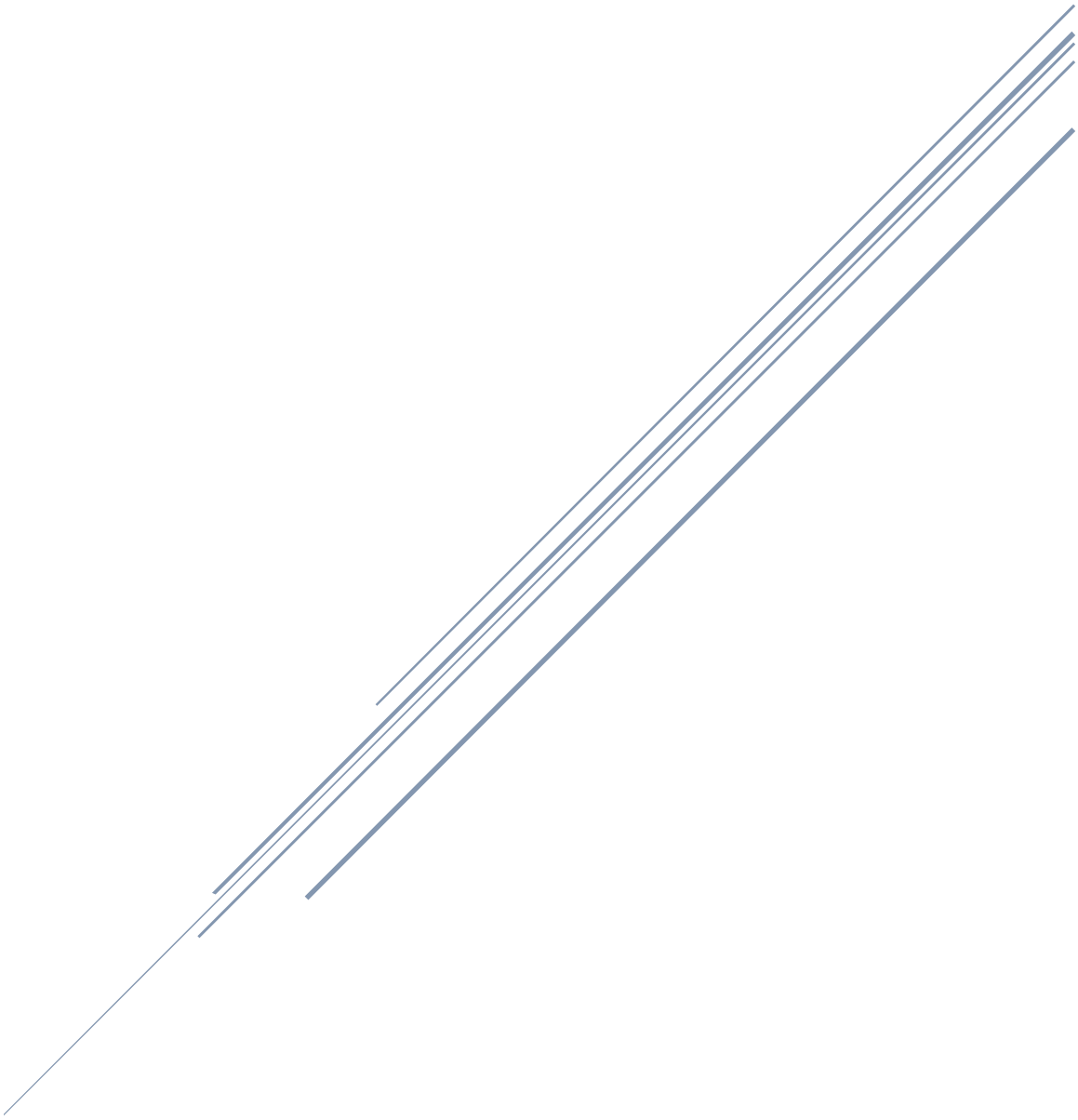
EZRENTAL DATABASE APPLICATION DESIGN & IMPLEMENTATION

Auto Rental Point-of-Sales Management System



By: Steve Carchi

EXECUTIVE SUMMARY

An overview of this document, including deliverables and expectations for the project.

- Provides an overview of the problem statement and objectives, along with the roles and responsibilities of those involved in the completion of the project.

- Business and technical requirements.

- Explanation of proposed application physical architecture, including the database management system physical architecture and the project methodology used to achieve the objectives.

- By the end, the Database design and implementation to support both the EZRental Point-of-Sales & Back-end Management Systems will be ready for interaction between the Database manager system and the user.

    - Includes Conceptual Model, Logical Model and Normalized Model
    - Data dictionary derived from the Normalized model
    - Physical Model Schema Diagram
    - Implementation and testing of database using Microsoft SQL Server

PROBLEM STATEMENT & OBJECTIVES

This section provides an overview of the problem presented and objectives of the project.

Project Objectives

- To design & implement a Client/Server Application Auto Rental Point-of-Sales Management System named EZRental POS, which includes an e-commerce site name EZRental.com.
- Basic objectives and architecture being targeted:
  - The EZRental POS System has been designed to allow customers, both retail and corporate, to reserve vehicles for renting like existing in-person or online car rental systems such as Avis, Hertz, Budget, etc.
  - The application was designed to support dozens of major cities around the world. In addition, provide a great user experience both in the physical rental agencies as well as online system with the best competitive pricing available in the market.
  - The company currently has rental agency branches in US, Canada, Mexico, United Kingdom, Japan & Australia and looking to expand further globally into other markets in Asia, Africa, and the Mediterranean.
- Analysis, Design, development, implantation, and testing of the database used by the Client/Server Application EZRental POS.

PROJECT ROLES & RESPONSIBILITIES

This section contains the roles and responsibilities for the members of the project. Their roles and individuals involved in this project are shown below.

Person	Role	Description
Mr. Rodriguez	**Program Manager & Project Manager**	▪ Owner of the project and liaison to Manage the EZRental Inc. ▪ Roles include but not limited to. 1. Owner of project responsible for the success of the project. 2. Project Management 3. Scrum Master that ensures the project stayed on time and moved in the right direction. Cleared any obstacles impeding the team's progress etc.
Consultant #1 (Mr. Rodriguez)	**Business & Database Analyst**	▪ Interviewed the stakeholders at **EZRental Inc**. And created the Business Requirements that were the foundation to the database design & implementation. ▪ Roles include but not limited to: 1. Engaged in discovery activities & interviewed the stakeholders at **EZRental Inc**. 2. From the interview and discovery ***created*** **1) ER/EER Conceptual Data Model from the business requirements & 2) Normalized Logical Model**.
Consultant #2 (Steve Carchi)	**Database Developer**	▪ Used the Normalized Logical Model created by consultant #2 to create the Data Dictionary, Physical Schema Diagram, and Implemented the Database Application for the Auto Rental System. ▪ Roles include but not limited to: 1. Used the Normalized Logical Model created by consultant #2 to do the **following 1) Created Data Dictionary tables for each logical table targeting Microsoft SQL Server Data Types** & **2) Created Physical Schema Diagram**. 2. From these two deliverables, **1) implemented the Database Application using Microsoft SQL Server** for the Auto Rental System.
Consultant #3 (Steve Carchi)	**Database Administrator**	▪ Installed the DBMS, maintained, and operated the DBMS throughout its lifetime. ▪ Roles include but not limited to: 1. As DB Admin, they **1) Setup & installed Microsoft SQL Server DBMS. 2) Microsoft SQL Server Management Studio Administrative tool**. 2. They also **3) Operated & Maintained the DBMS**.

Person	Role	Description
Consultant #4 (Mr. Rodriguez)	**Object-Oriented-Programming Architect**	▪ An Object-Oriented-Programming Architect was hired by Mr. Rodriguez that interviewed the stakeholders at **EZRental Inc**., from a client application programming prospective and in addition designed the Class/Object model based on interview and analysis results. ▪ Roles include but not limited to: 1. Engaged in discovery activities & Interviewed the stakeholders at **EZRental Inc**. 2. From the interview and discovery **1) Designed/Architected the Object-Oriented-Programming Class/Object Model**. 3. From the interview and discovery **1) Designed a high-level Graphical User-Interface (GUID) wireframe**, & **2) front-end features & functionality**.
Consultant #2 (Steve Carchi)	**Full Stack Application Developer**	▪ Object-Oriented-Programming developer which implemented the Windows Client application using **C# & .NET technologies** & 2) on the database side, implemented stored procedures and supported the databased team as needed. ▪ Roles include but not limited to: 1. As OOP developer, **1) Object-Oriented-Programming of Class/Object Model designed by Consultant #4 of the Windows Client/Server Application Client using C# & .NET technologies**. 2. In addition, **2) Database Stored Procedures and other development requirements in the Back-end DBMS**.
Consultant #2 (NA)	**Full Stack Web Developer**	▪ Object-Oriented-Programming developer & Web Developer that implemented the Web-based application using **C#, .NET technologies** and other technology & 2) on the database side, implemented stored procedures and supported the databased team as needed. ▪ Roles include but not limited to: 1. Web Developer, **1) Object-Oriented-Programming of Class/Object Model designed by Consultant #4 of the Web Client/Server Application Client using C# & .NET technologies & Other Web Technology** 2. In addition, **2) Database Stored Procedures and other development requirements in the Back-end DBMS**.

APPLICATION BUSINESS REQUIREMENTS

A business analyst interviewed EZRental Inc. to gather the necessary data required for the application and database design.

Business Requirements

About Us:

EZ-Car Rental is an auto rental company that rents vehicles such as cars, SUVs, minivans & cargo vans. In addition, specialized vehicles such as trucks, motorcycles, etc. We operate in several countries with rental agency locations in the US, Canada, Mexico, UK, Japan & Australia. In each country we operate, multiple rental agencies can exist in a city. For example, New York City has 2 rental agencies in Manhattan, one in Brooklyn and two in Queens, one in each airport. With multiple rental agencies in cities, a customer can pick up a vehicle in one location and drop it off at another.

Current Challenges:

Our current rental system is outdated, with a poor user-experience, inefficient (breaks often thus expensive to operate), does not meet our business requirements, and is not scalable (cannot be easily updated with new features). Also, very important the current system is not elastic since it does not give us the flexibility to scale-up or scale-down based on business trends and seasonal changes in the market.

We want to invest in modernizing our business with a new vehicle management system that can meet these challenges and give us: a great user-experience, meet our business new requirements, scalable, and elastic to adopt to business trends and seasonal market changes. Elasticity is very important since we are also faced with a new type of competition; small rental companies that are nimble and can quickly adopt to market changes thus able to provide new offerings that are appealing to customers thus affecting our profits. These smaller competitors are using new technologies that enable them to be nimble and elastic. Figurative speaking "*they are eating our lunch*".

We look forward to your proposed architecture & implementation of this new system. Below are our business requirements.

Our Agencies:

A rental agency is identified by a unique number *rental agency ID*, *agency name*, *address* that is composed of the following elements: *address line1, address line 2*, *city*, *state code*, *zip code* & *country*. In addition, we also need to capture the agency's *phone number*, and *email*.

Our Customers:

EZ-Car Rental offer their services to two types of customers: corporate customers & retail customers. Corporate Customers are individuals whose corporation have a contract with us and get special corporate rates for their employees. On the other hand, retail customers are consumers not associated with a company.

To run our business, the application must store the following information for both type of customers (retail & corporate):

- A *Customer ID* number which uniquely identifies the customer, *customer name* which is composed of: *first name*, *last name*.
- *Birth date*, *Age*, *Address* which includes the elements: *address line1, address line 2*, *city*, *state code*, *zip code* & *country*.
- Agency *phone number* & *email* which is required to rent. In addition, the unique *driver license number* and *driver license expiration date*.
- Another very important attribute we need to capture for every customer is the *credit card*. You cannot rent one of our vehicles without a credit card. A *credit card* includes the following components: *credit card number* that uniquely identifies the credit card, credit card *owner name*, *merchant name*, *expiration date*, *billing address* composed of *address line1, address line 2*, *city*, *state code*, *zip code* & *country*. Other attributes of credit card are *credit card balance*, *credit card limit* & *activation status* which is true if the credit card is active and can be used or false when disabled.
- Business rules related to a credit card are:
  - A customer can have many credit cards they can use to pay for rental transactions.
  - A credit card can be co-owned by many individuals such a family member or corporate entity the customer works for.

Business Requirements

Our Customers (Cont.):

For our corporate customers only, we must store the following properties: unique *company ID* (we have an ID number for each company), *company name*, company address which includes the elements: *address line1, address line 2*, *city*, *state code*, *zip code* & *country*, in addition, *company contact* which is composed of *contact name*, *contact phone number* & *contact email*. Finally, we need to store the *company's daily rental rate* or rate applied to the corporate customers rentals.

Retail customers can opt-in to enrolled in the EZPlus rewards program where they earn points every time they rent and can redeem these points for future rentals. Note that the EZPlus program is optional for retail customers & points are earned only when they rent a vehicle. In addition, retail customers are eligible for special promotional discounts or coupons they can obtain from other businesses and organizations. Therefore, data unique to a retail customer that we need to capture for the promotional discount are: unique random number *discount ID* to uniquely identify a discount, a unique *discount code* or coupon code. and *discount code description*. For the EZPlus rewards program we need to store: unique random *EZPlus ID*, the unique *Ezplus rewards code*, *EZPlus rewards earned points*. Examples of common *discount ID*, *discount code*, *discount code description*, *EZPlus ID*, *EZPlus rewards Code* and *EZPlus earned points* are:

Discount ID	*Discount Code*	*Discount Code Description*
1234..	AAA99700	**AAA Membership Discount - 25% off base rate plus 10% donated for breast cancer research.**
5678..	GOV87569	**Government Employee Discount - 30% off base rate**
9101..	*STA34156*	**State Employee Discount for 25% off base rate**
1213..	VET20551	**Veteran Discount 35% off base rate Plus 10% donation to veteran's family fund.**
Etc..	Etc..	**Etc..**

EZPlus ID	*EZPlus Rewards Code*	*EZPlus Rewards Earned Points*
1234..	EZP90098	**10000**
5678..	EZP10001	**500**
9101..	*EZP64932*	**159000**
1213..	*EZP20051*	**23000**
	Etc..	**Etc..**

In this business, we have the following rules for our customers:

- We only have two types of customers retail customer or corporate customers. No other type of customer exists.
- A customer *cannot* be a retail & corporate customer at the same time. A customer can only rent as a retail customer or as a corporate and these transactions must be separate. We don't want our customers to be able to combine both retail customer discounts, rewards program, and corporate rates at the same time.

Business Requirements (Cont.)

Our Vehicles:

EZ-Car Rental needs a system to manage their vehicles for renting, maintenance, selling, etc. Vehicles are classified into4 main types: cars, SUVs, minivans, and cargo vans. These are the vehicles most rented and available at every rental agency. Nevertheless, there are other categories of vehicles available only certain locations such as recreational vehicle, motorcycles etc. No matter what type of vehicle, all vehicle types of vehicles share the following common characteristics:

- Each vehicle is identified by the random number *vehicle ID*. In addition, each vehicle is also identified by the alpha-numeric vehicle *VIN number*. Other attributes include the *vehicle name* composed of *make*, *model*& year.
- Additional attributes are *color*, also the *licenseplate* composed of the following components: *license plate number*, *license plate state*. More attributes are *mileage*, *transmission type* (e.g., Manual, automatic, Continuously variable transmission (e.g. CVT), Semi-automatic & dual-clutch)and *seat capacity*.
- All vehicles also have a special identifier we use to track the vehicle status named *vehicle status ID*. This is a unique number that identifies the status of a vehicle, which works in conjunction with *vehicle status description* which describes the status, such as reserved, rented, available, maintenance, not available, transferred, etc. Below is the list of vehicle status IDs we are currently using and their descriptions:

Vehicle Status ID	*Vehicle Status Description*
1	**Reserved.**
2	**Rented.**
3	**Available.**
4	**Not available**
5	**Maintenance**
6	**Transferred to another agency**

In addition to these attributes shared by all vehicles, the unique characteristics for each of the 4 vehicle types available in all agencies are as follows:

- A Car is a vehicle whose *trunk capacity* measured in cubic feet volume is advertised to our customers. Customers can decide which vehicles better fits their needs based on the number of luggage they are carrying etc. For example, a luxury Mercedes E class car has a trunk capacity of 18.5cubic ft.
- An SUV has a *towing capacity* in pounds. Towing capacity is number in pounds or could also be a decimal in pounds. For example, some of our SUV have a maximum towing capacity of 3,000 pounds. Another attribute of SUV is classification if the SUV is *All-Wheel-Drive*or not which is a yes/no or true/false option.
- A Minivan has the option of having a *disability option package* or not or true, false.
- Finally, a Cargo Van, has a *cargo capacity* in cubic feet volume. For example, the typical volume of our Vans is 245 cubic feet (cu.ft.). Cargo Vans also have a *maximum payload* attribute that determines how much weight in pound it can hold. Our vans have a typical max load of 3,880 lbs.

As stated previously, there are other types of vehicles of interest that in some location we may want to store data on other than car, SUV minivans and cargo van. In addition, a reservation or rental can only be for one of these four categories of vehicles not a combination. You can only rent either a car, SUV minivans, cargo van or other for a reservation or rental, not a combination such as a car & SUV at the same time. Each reservation is unique to one vehicle.

In our business, we have the following business rules for our vehicles:

- Every vehicle is owned by one agency. The vehicle can be pick-up and dropped-off at any agency, but only one agency is the vehicle's owning agency. An agency can own many vehicles, but a vehicle can only be owned by one agency.
- A vehicle can currently be located at any agency depending on where it was dropped-off after a rental. We need to track the current agency where the vehicle is located, to arrange a transfer or a rental that will ultimately direct the vehicle to the owning agency.

Business Requirements (Cont.)

Reservation Process:

A vehicle must first be reserved before the vehicle can be rented. There is a distinction between a reservation and a rental. A reservation guarantees a vehicle will be ready for you to be pick-up and rented. A rental means a customer complied with the reservation and rented the vehicle.

We have the following rules for reserving a vehicle:

- A reservation is not made for a specific vehicle, but to a vehicle rental category. Rental category examples are economy, intermediate, full size, luxury.
- Thus, a customer makes a reservation of a vehicle rental category at a rental agency. Therefore, the reservation process involves a customer a vehicle rental category and the rental agency.

A rental category contains a list of vehicles depending on the vehicle type: Car (economy, intermediate, full size, luxury), SUV (standard, full size etc.), or Cargo Van etc. Each of these categories have a different price range. Therefore, for a vehicle rental category we need to capture the unique *vehicle rental category ID* that identifies the category of the vehicle being reserved or rented, *category name* and finally *category daily rental rate* for the category. We used a specific code for our vehicle rental category ID, category name & daily rental rate. The table below shows the ID, category names and cost we use:

Vehicle Rental Category ID	*Vehicle Rental Category Name*	*Category Daily Rental Rate*
1	Car-Economic	$113.99
2	Car-Compact	$115.99
3	Car-Intermediate	$116.67
4	Car-Standard	$119.99
5	Car-Full Size	$121.99
6	Car-Premium	$127.79
7	Car-Luxury	$139.99
8	SUV-Intermediate	$127.99
9	SUV-Standard	$128.99
10	SUV-Standard Elite	$135.99
11	SUV-Full Size	$148.99
12	SUV-Premium	$157.99
13	Minivan-Standard	$152.99
14	Van-Passenger Van (12 passengers)	$161.00
15	Van-Cargo Van	$19.95
16	Pick Up-Mid Size	$69.95
17	Pick Up-Full Size	$105.99
18	Motorcycle-Touring	$19.95
19	Motorcycle-Cruiser	$199.99
20	Motorcycle-Scooter	$79.95

We have the following business rule relate to a vehicle and a vehicle rental category:

- A vehicle is a member of a vehicle rental category.
- A vehicle rental category can have one, none or many vehicles belonging to that category at any given time, nevertheless, a vehicle can only belong to one vehicle rental category.

As stated previously, a customer makes a reservation of a vehicle rental category at a rental agency. Therefore, the reservation process requires the customer, vehicle rental category & rental agency for a reservation to be made. The following rules apply to a reservation:

- A vehicle can be reserved to be picked up at the **INDICATED** rental agency and dropped off at the **SAME** rental agency.
- A vehicle can be reserved to be picked up at the **INDICATED** rental agency and dropped off at a **DIFFERENT** rental agency.
- A reservation is made only for one pick-up rental agency, but a rental agency can have many reservations for pick-ups taking place.
- A reservation can only be for one drop-off rental agency, but a rental agency can have many reservations drop-offs taking place.

When a customer reserves a vehicle category for a specific rental agency, we wish to capture the following:

- A unique *reservation ID* to track the reservation, the *reservation pick-up rental agency* or the rental agency where the vehicle will be picked up, and the target *reservation drop-off rental agency*.
- In addition, we need *reservation pick up date*, *reservation pick up time*, *reservation drop off date* and *reservation drop off time*, also the *reservation estimated rental cost*.

Business Requirements (Cont.)

Reservation Process (Cont.):

- Finally, we need to store the unique *reservation status ID* which is a unique number we use to indicate the status of a reservation and *reservation status description* which describe each of the status such as: confirmed, cancelled, completed etc. Below is an example of the reservation status ID we use and description for each status.

Reservation Status ID	*Reservation Status Description*
1	**Confirmed.**
2	**Modified & reconfirmed.**
3	**Cancelled & Closed.**
4	**Fulfilled & Closed.**
Etc..	**Etc..**

For a reservation we must adhere to the following rules:

- A customer can make none, one or many reservations for a vehicle rental category at a rental agency.
- A rental category can be reserved by none, one or many customers at a rental agency.
- A rental agency can get many or no reservations for a vehicle rental category by a customer.
- A reservation can only have one pick-up rental agency location, but a rental agency can have many reservation pick-ups happening.
- Each reservation has a drop-off rental agency (may be different than pick-up rental agency). A reservation can only have one drop-off rental agency location, but a rental agency can have many reservation drop-offs taking place.

The Rental Process:

Once a vehicle has been reserved, the vehicle can be rented (picked up/dropped off) as per the scheduled of the reservation agreement. A rental means a customer complied and fulfilled the reservation and rented the vehicle.

For the rental process, the following rules apply:

- A customer rents a vehicle at a rental agency. This means the rental process requires the customer, vehicle, and & rental agency for a rental to be complete.
- During the rental process we may have any of the following scenarios:

1. A vehicle can be picked up at the **SAME** rental agency as indicated by the reservation and dropped off at the **SAME** rental agency.
2. Or a vehicle can be picked up at the **SAME** rental agency as indicated by the reservation and dropped off at **ANOTHER** rental agency.
3. Or a vehicle can be picked up at **ANOTHER** rental agency other than what was indicated by the reservation and dropped off at **SAME** rental agency of the reservation.
4. Or finally, a vehicle can be picked up at **ANOTHER** rental agency other than what was indicated by the reservation and dropped off at **ANOTHER** rental agency of the reservation.

- ❖ Note that for scenarios 3 & 4, we cannot guarantee that the vehicle rental category of the reservation will be available at the agency other than what was agreed in the reservation. We will do our best to accommodate the change during these scenarios or find another vehicle that will be closed to the original reservation.

- A rental can only be for one pick-up rental agency, but a rental agency can have many rental pick-ups taking place.
- A rental can only be to one drop-off rental agency, but a rental agency can have many rental drop-offs taking place.

When a customer rents a vehicle at the rental agency, we need to capture the following information about the rental:

- The *rental agreement ID* that uniquely identifies the rental transaction, *rental pick up date*, *rental pick up time*, *rental drop off date* and *rental drop off time*, *rental pick up odometer value*, *rental drop off odometer value* & *rental total cost* which can be calculated based on selected fuel option, insurance option, vehicle rental category price and other factors.

Business Requirements (Cont.)

The Rental Process (Cont.):

- In addition to the above, customers receive a vehicle with a full tank of gas and customers have the option to return the car on a full tank of gas they purchase or pay a charge to return the car as is, therefore, we need to capture the unique *rental fuel option ID*, *rental fuel option description* and *rental fuel option additional cost*. We currently use the following fuel option IDs, descriptions, and cost:

Rental Fuel Option ID	*Rental Fuel Option Description*	*Rental Fuel Option Additional Cost*
1	**Return with a full tank or on return, pay for gas that is missing.**	**Calculated during car return and based on the current price of a gallon of gas. Price will vary.**
2	**Pay for full tank in advanced at time of rental, return car empty. No refund for unused gas.**	**Calculated during time of car rental and based on current price of a gallon of gas. Price will vary.**

- Also, we give customer options for car insurance & protection, therefore we need to capture the unique *insurance option ID*, *insurance option description* and *insurance option additional cost*. We currently use the following insurance option IDs, descriptions, and cost:

Rental Insurance Option ID	*Rental Insurance Option Description*	*Rental Insurance Option Additional Cost per Day*
1	**No insurance. Opt-out.**	**$0.00**
2	**Collision Damage Waiver Max - Agency will pay for damage, lost or stolen vehicle.**	**$49.99**
3	**Collision Damage Waiver 3000 - Agency will pay for first $3,000 of loss or damage, renter pays all loss & damage after $3,000.**	**$39.99**
4	**Lability Extended Protection – Agency provides renter with third party liability protection up to $1 Million per accident for bodily injury or death or property damage to others.**	**$89.99**
5	**Roadside Assistance Plus – 24/7 roadside assistance, replacement for lost keys, flat tire service, fuel delivery, etc.**	**$15.99**

- Other attributes required for the rental that we need to capture are the unique *rental status ID* & *rental status description*. We currently use the following rental status IDs & descriptions:

Rental Status ID	*Rental Status Description*
1	**Picked up as scheduled.**
2	**Dropped off as scheduled.**
3	**Returned late**
4	**In progress.**
5	**Roadside assistance in progress.**
7	**Unknown**

Business Requirements (Cont.)

The Rental Process (Cont.):

- Finally, we need to capture the *rental deposit* for a rental. The rental deposit value is calculated based on the **rental period + 25% of the rental period** for any damage or other charges. This deposit is refunded to the customer's credit card when the vehicle is returned in the condition in which it was rented.

We need to be able to associate a reservation to a rental and vice versa, therefore we maintain the following additional business rules for our rental & reservation:

- A reservation is made for a rental and the opposite holds true; a rental is based on a reservation.
- But NOT all rentals are based on a reservation. We allow a customer to walk into a rental agency and rent a vehicle without a reservation.
- When a reservation is made for a rental, then it must be for only one rental, and a rental can be for a reservation but not mandatory since a customer can walk into an agency and rent a vehicle without a reservation.

Our Employees:

EZ-Car Rental employees consist of customer service agents who interact with our customer to reserve and rent vehicles. In addition, we have auto specialists who work in our services centers servicing our vehicles. In addition, drivers to transport our vehicles from one agency to another and maintenance personnel who maintain our agencies and finally our business team that handles the day-to-day business activities in our agencies and other roles. For now, we are only interested in storing the following data for all these types of employees:

- An *Employee ID* which uniquely identifies the employee, *employee name* which is composed of: *first name*, *last name*, also *employee address* which includes the components: *address line1, address line 2*, *city*, *state code*, *zip code* & *country*. Also, *employee phone*, *employee job title* and *employee email.*

Security & Access:

To access our systems proper security and authentication is required. Only authorized users can have access our agencies Point-0f-Sales & Back-End Management systems. In addition to our **EZRental.com** portal by our customers. Therefore, due to security and regulatory compliance purpose, we want to separate the employee access data from the customer access data by using two separate user accounts:

- Employee user accounts
- Customer user accounts

Security Access for Employees to Computer Systems in our Agencies (Employee User Accounts):
For our authorized employees & customer service employees to access the agencies Point-0f-Sales & Back-End Management systems they need to log in by entering a username & password for access to the application. This means every employee owns an employee user account.

An employee user account should store the user *employee user account ID* a unique identifier alpha-numeric string that identifies the employee user account, *employee username* another unique alpha-numeric that identifies each individual user, and finally the *employee password* alpha-numeric that is known only to the user, An employee can own one employee user account only, and an employee user account can only be owned by one employee only since the user account represents the identify of that one employee.

Security Access for our Customers who register for our EZ-CarRental.com web site (Customer User Accounts):
Customer who accesses our online portal to reserve and rent our vehicles also need a username and password to access our system, therefore each customer owns a customer user account.

A customer user account should store the user *customer user account ID* a unique alpha-numeric string identifier that identifies the customer user account, *customer username* another unique alpha-numeric value that identifies each customer, and finally, the *customer password* that is an alpha-numeric known only to the customer. A customer can own one customer user account only, and a customer user account can only be owned by one customer. For a period, we will need to register customers into our business but the **EZRental.com** web portal may be incomplete, therefore creating a customer user account for a new customer can be optional. We will force the creation of customer user accounts when they login to our portal for the
first time.

Business Requirements (Cont.)

Vehicle Transportation:

We need to know where our vehicles are located at all times, such as at the Rental Agency that owns the vehicle, another Rental Agency that does not own the vehicle, being transported from one Rental Agency to another as a result of a vehicle transfer after a rental to the owning rental agency, being transported as a new delivery to a Rental Agency from our distribution center, being transported for maintenance, or currently being rented by a customer. Vehicles need to be tracked or location status known. At this time, we are only interested in tracking when a vehicle is transported from one Rental Agency to another Rental Agency under the following scenarios:

- Vehicle can be located at a Rental Agency that does not own the vehicle after a rental dropping off at a different location than the picked up owning Rental Agency, thus vehicle eventually needs to be transported and delivered to the owning agency.
- Another non-owning Rental Agency requests support from other Rental Agency(s) for loans of vehicle(s) to borrow due to an unexpected busy period and requesting agency is short on inventory. After the first agency is done with the loaner vehicles, these vehicles need to be returned to the borrowed owning Rental Agency(s).
- In our current process & systems we currently use the following reason IDs and reason descriptions:

Transport Reason ID	*Transport Reason Description*
1	Rental Drop off at different location
2	Vehicle Loaned to another Agency
3	Pick up from Distribution Center
4	Drop off to Distribution Center
5	Vehicle sent for maintenance
7	Unknown

Note that transportation to and from Rental Agency is executed by an employee who is part of a transportation team or drivers. Therefore, when an employee executes a transport request of a vehicle to and from Rental Agencies, we need to capture the following information:

- *Transport pickup agency ID, Transport drop-off agency ID, Driver departure date, driver departure time, vehicle pick up date, vehicle pick up time, transport completed arrival date, transport completed arrival time, estimated arrival date, estimated arrival time*, & actual *transport time to completion*.
- In addition, we need to know at any time the transport status and transport status description of the transfer, such as: transfer completed, on route to pick up location, on route from pick up location, etc. Currently we track a transportation event using the following ID and description:

Transport Status ID	*Transport Status Description*
1	Transport completed
2	On route to pick up location.
3	On route from pick up location
4	At pickup location. In progress (Loading etc.)
5	Pickup location delay
7	Unknown

The goal again is to be able to know where our vehicles are always located and status.

Conclusion:

The business data listed in this business requirements document is what we need to capture for our business to operate. As our business evolve, additional data will be required. We will address these new requirements in future versions of the application. For example, invoice processing & employee management at our rental agencies are features on our roadmap. Therefore, our expectations is that the design is modular and scalable for future growth.

APPLICATION DEVELOPMENT& TECHNICAL REQUIREMENTS

An Application Analyst/Architect interviewed EZRental Inc., the project Business Decision Makers (BDMs), stakeholders and Information System Technical Decision Makers (TDMs) to compile the requirements below.

Application Development & Technical Requirements

Introduction & Current Challenges

As described in the Business Requirements, the current rental system is outdated, with a poor user-experience, breaks often thus expensive to operate, does not meet our business requirements, and is not scalable so it cannot be easily updated with new features etc. Also, not elastic since it does not give us the flexibility to scale-up or scale-down based on business trends and seasonal changes in the market. We want to invest in modernizing our business with a new vehicle management system that can meet these challenges and give us a great user-experience, meet new business requirements, scalable, and elastic to adopt to business trends and seasonal market changes.

We have an outdated IT infrastructure in our datacenter and there is a current initiative to modernize our datacenter and also leverage cloud technology in a hybrid environment to save on cost, streamline our operations and drive innovation.

We look forward to your proposed architecture & implementation of this new system that will meet these requirements. Next sections contain the results of our application development & technical requirements.

Rental Agencies Application & Technical Requirements:

The rental agencies are location where our customers both Retail & Corporate will engage our *Customer Service Representatives* to engage in rental/return activities in addition to other transactions such as registering, searching & updating customer information etc. Therefore, the application in the rental agencies is vital to the user-experience for both our *Customer Service Representatives* as well as our *Customers*.

We are forecasting that is some locations such as major city centers and airports, there will be many customers engaging throughout the day thus increasing the risk of a poor customer experience in addition to the work overload and poor experience for our *Customer Service Representatives*. We want our *Customers* to be serviced quickly and efficiently with a great experience, and our *Customer Service Representatives* to be able to process each *Customer* easily and effectively. With these criteria in mind, the application at our rental agencies must adhere to the following requirements:

Rental Agency Application Architecture Requirements:

Below are the requirements for the application used in our rental agencies by our customer service representatives, inventory team, service personnel and other employees working in our agencies:

1. Client application processing, transaction and response must be fast to minimize service time for a customer.
2. All transaction processing should be done in the user's computer or desktop for fast processing and response.
3. Application Architecture must be reusable and scalable to support future updates and new feature enhancements, without a long development lifecycle.
4. Depending on the architecture NYC-Tech Solutions Inc., decides for the application in the rental agencies (Desktop client or Web client), the primary Application Development Platform we use is ***C# & .NET technologies***. For any Web related development, we support JavaScript, React, NodeJs and other standard Web Technologies. We have aligned ***C#/.NET*** & **Web** developers that have been assigned to assist, support and update the application once NYCTech consultants complete the project and development of this system.
5. Rental Agency Desktop Application Security Authentication System – Proper security and authentication must be implemented to make sure only authorized customer service representative and other rental office employees can access the Point-0f-Sales with appropriate conditional access.

Application Development & Technical Requirements (Cont.)

Rental Agency Application Features and Functionalities Requirements:

The list of features and functionalities that we have compiled for the rental agencies' application are listed in the table below:

No.	Feature	Functionalities
1	**EZRental** Rental Agency Point-of-Sales (POS) System	▪ Car Rental, Car Return, New Customer Registration & Search Customer Information, Customer Update, Customer Deletion, Customer Listing operations etc.
2	**EZRental** Rental Agency Back-Office Vehicle Inventory Management System	▪ Back-office system meant for employees to perform bulk IN-MEMORY inventory processing or management tasks on vehicles such as adding vehicles to the system, searching for vehicles, updating vehicles etc. ▪ This system is NOT meant for Point-of-Sales, but for the inventory management employees who need to search, add, remove etc., a large/bulk number of vehicles or employees during a session. ▪ Back-office vehicle Management features – Allows inventory personnel and employees to bulk-manage Cars, SUVs, Mini-Vans, Cargo Vans to be searched, added, removed, printed, listed etc.
3	**EZRental** Rental Agency Back-Office Credit Card Management System	▪ The EZRental Credit Card Management System is a Back-office system meant for the Credit Card Department Employees to manage Credit Card Information. These uses can Search, Add, Edit & Delete credit card information in the database
4	**EZRental** Rental Agency Back-Office Employee & Customer User Account Management System	▪ The EZRental Customer & Employee User Account Management System is a Back-end system meant for IT ADMINISTRATOR Employees to manage both Employee & Customer USER ACCOUNTS.
5	**EZRental** Rental Agency Desktop Application Security Authentication System	▪ Proper security and authentication must be implemented to make sure only authorized employees can access the Point-0f-Sales, Back-End Management system or any other access to the applications.

Rental Agency Application Graphical User Interface Requirements:

- Graphical User-Interface should be fast rendering and user-friendly workflow.
- Visual screens or forms should be rich in color and appearance and navigation flow should be flexible and easy.
- The following UI controls or data field need to be pre-populated in GUI Screens:

  - **Addresses**

    - Any forms/UI which contains addresses, the STATE & COUNTRY fields should be automatically populated with a list of STATES or COUNTRIES, so the user does not have to manually enter a state or a country and simply select from drop-down list etc.

  - **Discount Codes:**
    - UI screens with customer's DISCOUNT CODE fields should be prepopulated with discount codes. The idea is the user should be able to select the discount to apply to a customer entry from a drop-down list/Combo Box etc. Note that this may or may not include the Discount Code Description on the UI screen as well.
    - Also note that the DISCOUNT CODE VALUES are generated by our Marketing Team and need to be pre-populated in the database before a code can be used. Therefore, the discount codes are prepopulated in the database.
    - Currently, when the Marketing Team generates a new code, they make the request to the database administrator to manually enter an update any new Discount Codes.
    - In the future, we want the application to have the necessary features for the Marketing Team to be able to manage the discount codes. This is not an immediate requirement out of the gate but should be targeted as part of a future upgrade.

Rental Agency Application Graphical User Interface Requirements (Cont.):

- **EZPlus Rewards Codes:**
  - The EZPlus Reward UI screens with customer's EZPLUS REWARDS CODE fields should be prepopulated with the EZPlus Rewards code for the customer is being applied to. The idea is the user should be able to select the EZPLUS REWARD CODE to apply to a customer entry from a drop-down list/Combo Box etc., or be handled by the back-end database.
  - **Important:** The EZPLUS REWARDS CODE VALUES are NOT generated by a business entity in our organization, but AUTOMATICALLY GENERATED by the application on the fly when registering a new customer. This is a different approach compared to the DISCOUNT CODE which are generated by Marketing Team. In this case, the EZPlus Rewards Code values are generated by the application and available via the UI screen to be used or some other method of generation.
  - To finalize this requirement, the idea is the EZPlus Rewards Code should be automatically generated and either appear in the UI Screen or automatically generated in the database.

- **Company Name:**
  - UI screens with corporate customer's COMPANY NAME fields should be prepopulated with the list of corporations that are members of our corporate program, which enables our users to avoid having to manually enter the company name. Note that this may or may not include the Company ID in the UI Screen which is a unique number with business value that we assign to each company.
  - Note that the company names. Company ids and other company data are managed by our Corporate Sales Team and need to be pre-populated in the database before any corporate customer processing can be made. Therefore, the company information is prepopulated in the database.
  - Currently, when the Corporate Sales Team adds a new corporation or company into the program, they make the request to the database administrator to manually enter and add the new company to the database.
  - In the future we want the application to have the necessary features for the Corporate Sales Team to have the functionality to manage the data of our corporate companies via the application. This is not an immediate requirement out of the gate but should be targeted as part of a future upgrade.
  -

- **Vehicle Status:**
  - UI screens for vehicle inventory management, VEHICLE STATUS field should be prepopulated with the list of vehicle status. Based on the business requirements, the current list of vehicle status is listed in table below:

Vehicle Status ID	*Vehicle Status Description*
1	**Reserved.**
2	**Rented.**
3	**Available.**
4	**Not available**
5	**Maintenance**
6	**Transferred to another agency**

  - Currently populating the database with a vehicle status record is handled manually by the database administrator. In the future we would like the application to have the necessary features for our business to be able to manage the vehicle status data. This is not an immediate requirement out of the gate but should be targeted as part of a future upgrade.

- **Rental Agency:**

  - UI screens that required adding or managing a RENTAL AGENCY field should be prepopulated with the list of rental agencies in our company.
  - Currently populating the database with a rental agency record is handled manually by the database administrator. In the future we would like the application to have the necessary features for our business to be able to manage the rental agency data. This is not an immediate requirement out of the gate but should be targeted as part of a future upgrade.

Rental Agency Application Graphical User Interface Requirements (Cont.):

- **Vehicle Rental Category:**
  - UI screens that require the use of the VEHICLE RENTAL CATEGORY fields, must be prepopulated with the list of vehicle rental categories. Based on the business requirements, the current list of vehicle rental categories is as follows:

Vehicle Rental Category ID	*Vehicle Rental Category Name*	*Category Daily Rental Rate*
1	Car-Economic	$113.99
2	Car-Compact	$115.99
3	Car-Intermediate	$116.67
4	Car-Standard	$119.99
5	Car-Full Size	$121.99
6	Car-Premium	$127.79
7	Car-Luxury	$139.99
8	SUV-Intermediate	$127.99
9	SUV-Standard	$128.99
10	SUV-Standard Elite	$135.99
11	SUV-Full Size	$148.99
12	SUV-Premium	$157.99
13	Minivan-Standard	$152.99
14	Van-Passenger Van (12 passengers)	$161.00
15	Van-Cargo Van	$19.95
16	Pick Up-Mid Size	$69.95
17	Pick Up-Full Size	$105.99
18	Motorcycle-Touring	$19.95
19	Motorcycle-Cruiser	$199.99
20	Motorcycle-Scooter	$79.95

  - Currently populating the database with vehicle rental category records is handled manually by the database administrator. In the future we would like the application to have the necessary features for our business to be able to manage the vehicle rental categories data. This is not an immediate requirement out of the gate but should be targeted as part of a future upgrade.

- **Reservation Status:**

  - UI screens that require the use of the RESERVATION STATUS field, must be prepopulated with the list of reservation status data. Based on the business requirements, the current list of reservation status is as follows:

Reservation Status ID	*Reservation Status Description*
1	Confirmed.
2	Modified & reconfirmed.
3	Cancelled & Closed.
4	Fulfilled & Closed.
Etc..	Etc..

  - Currently populating the database with a reservation status record is handled manually by the database administrator. In the future we would like the application to have the necessary features for our business to be able to manage the reservation status data. This is not an immediate requirement out of the gate but should be targeted as part of a future upgrade.

Rental Agency Application Graphical User Interface Requirements (Cont.):

- **Rental Status:**
  - UI screens that require the use of the RENTAL STATUS field, must be prepopulated with the list of rental status data. Based on the business requirements, the current list of rental status is as follows:

Rental Status ID	*Rental Status Description*
1	**Picked up as scheduled.**
2	**Dropped off as scheduled.**
3	**Returned late**
4	**In progress.**
5	**Roadside assistance in progress.**
7	**Unknown**

  - Currently populating the database with a rental status record is handled manually by the database administrator. In the future we would like the application to have the necessary features for our business to be able to manage the rental status data. This is not an immediate requirement out of the gate but should be targeted as part of a future upgrade.

- **Rental Fuel Option:**

  - UI screens that require the use of the RENTAL FUEL OPTION field, must be prepopulated with the list of rental fuel options data. Based on the business requirements, the current list of rental fuel option is as follows:

Rental Fuel Option ID	*Rental Fuel Option Description*	*Rental Fuel Option Additional Cost*
1	**Return with a full tank or on return, pay for gas that is missing.**	**Calculated during car return and based on the current price of a gallon of gas. Price will vary.**
2	**Pay for full tank in advanced at time of rental, return car empty. No refund for unused gas.**	**Calculated during time of car rental and based on current price of a gallon of gas. Price will vary.**

  - Currently populating the database with a rental fuel option record is handled manually by the database administrator. In the future we would like the application to have the necessary features for our business to be able to manage the rental fuel option data. This is not an immediate requirement out of the gate but should be targeted as part of a future upgrade.

Application Development & Technical Requirements

- **Rental Insurance Option:**

  - UI screens that require the use of the RENTAL INSURANCE OPTION field, must be prepopulated with the list of rental insurance options data. Based on the business requirements, the current list of rental insurance option is as follows:

Rental Insurance Option ID	*Rental Insurance Option Description*	*Rental Insurance Option Additional Cost per Day*
1	**No insurance. Opt-out.**	**$0.00**
2	**Collision Damage Waiver Max - Agency will pay for damage, lost or stolen vehicle.**	**$49.99**
3	**Collision Damage Waiver 3000 - Agency will pay for first $3,000 of loss or damage, renter pays all loss & damage after $3,000.**	**$39.99**
4	**Lability Extended Protection – Agency provides renter with third party liability protection up to $1 Million per accident for bodily injury or death or property damage to others.**	**$89.99**
5	**Roadside Assistance Plus – 24/7 roadside assistance, replacement for lost keys, flat tire service, fuel delivery, etc.**	**$15.99**

  - Currently populating the database with a rental insurance option record is handled manually by the database administrator. In the future we would like the application to have the necessary features for our business to be able to manage the rental insurance option data. This is not an immediate requirement out of the gate but should be targeted as part of a future upgrade.

Application Development & Technical Requirements (Cont.)

Customer Facing Self-Service Web-Portal Application Architecture Requirements:

We now address architecture requirements for the application used in customers via the public internet to make reservations to rent a vehicle, modify their personal account, profile etc.:

1. Customer will use a secure and standard Web Application via a Browser to access our self-service portal in the internet. We need a website to support all customer self-service related transactions.
2. Web Application Architecture must be reusable and scalable to support future updates and new feature enhancements, without a long development lifecycle.
3. For this web development, we support ***JavaScript***, ***React***, ***NodeJS*** and other standard Web Technologies. In addition, the primary Application Development Platform we use is ***C# & .NET technologies***. We have aligned ***C# & .NET*** & **Web** developers that have been assigned to assist, support, operated and update the application once NYCTech consultants complete the project and development of this system.
4. Web Portal Security Authentication System – Proper security and authentication must be implemented to make sure only the customer can access the **EZRental.com** website for his or her profile home page.

No.	Feature	Functionalities
1	**EZRental.com** Customer Web Portal	▪ Front-end WEB INTERFACE SCREENS & features used by customers via our web portal EZRentalCar.com to reserve a vehicle for rental and manage their account online. ▪ Features include: Search & reserve a car for rental, register as a new customer, search/view their account information, update their account etc.
2	**EZRental.com** Customer Web Portal Application Security Authentication System	▪ Proper security and authentication must be implemented to make sure only our customer can access the web portal to use the application.

Web Portal Application Web Pages User Interface Requirements:

The web pages graphical UI requirements are listed below:

- The GUI requirements for the web pages are like those functionalities of the Rental Agency Application that are found on the web site for example Search & reserve a car for rental, register as a new customer, search/view their account information, update their account etc.
- The design and graphics of the application should be appealing to customers and a smooth and fluent workflow.
- The following UI controls or data field need to be pre-populated in GUI Screens:

  - **Addresses**

    - Any web-page UI which contains addresses, the STATE & COUNTRY fields should be automatically populated with a list of STATES or COUNTRIES, so the user does not have to manually enter a state or a country and simply select from drop-down list etc.

  - **Discount Codes:**

    - Web pages with customer's DISCOUNT CODE fields should be a text box that allows the customer to ADD/APPLY the discount codes to redeem the coupon.

Rental Agency Application Graphical User Interface Requirements (Cont.):

- **EZPlus Rewards Codes:**

  - The EZPlus Reward web page screens with customer's EZPLUS REWARDS CODE fields should be prepopulated with the EZPlus Rewards code for the customer is being applied to. The idea is the user should be able to select the EZPLUS REWARD CODE to apply to a customer entry from a drop-down list/Combo Box etc., or be handled by the back-end database.
  - **Important:** The EZPLUS REWARDS CODE VALUES are NOT generated by a business entity in our organization, but AUTOMATICALLY GENERATED by the application on the fly when registering a new customer. The EZPlus Rewards Code values are generated by the application and available via the UI screen to be used or some other method of generation.
  - To finalize this requirement, the idea is the EZPlus Rewards Code should be automatically generated and either appear in the UI Screen or automatically generated in the database.

- **Rental Agency:**

  - Web pages that required adding a RENTAL AGENCY field should be prepopulated with the list of rental agencies in our company.

- **Vehicle Rental Category:**

  - Web pages that require the use of the VEHICLE RENTAL CATEGORY fields, must be prepopulated with the list of vehicle rental categories. Based on the business requirements, the current list of vehicle rental categories is as follows:

Vehicle Rental Category ID	*Vehicle Rental Category Name*	*Category Daily Rental Rate*
1	Car-Economic	$113.99
2	Car-Compact	$115.99
3	Car-Intermediate	$116.67
4	Car-Standard	$119.99
5	Car-Full Size	$121.99
6	Car-Premium	$127.79
7	Car-Luxury	$139.99
8	SUV-Intermediate	$127.99
9	SUV-Standard	$128.99
10	SUV-Standard Elite	$135.99
11	SUV-Full Size	$148.99
12	SUV-Premium	$157.99
13	Minivan-Standard	$152.99
14	Van-Passenger Van (12 passengers)	$161.00
15	Van-Cargo Van	$19.95
16	Pick Up-Mid Size	$69.95
17	Pick Up-Full Size	$105.99
18	Motorcycle-Touring	$19.95
19	Motorcycle-Cruiser	$199.99
20	Motorcycle-Scooter	$79.95

Application Development & Technical Requirements

- **Transport Reason:**

  - UI screens that require the use of the **Transport Reason** field must be *prepopulated* with the list of transport reasons data from **DATABASE**. Based on the business requirements, the current list of transport reasons are as follows:

Transport Reason ID	*Transport Reason Description*
1	Rental Drop off at different location
2	Vehicle Loaned to another Agency
3	Pick up from Distribution Center
4	Drop off to Distribution Center
5	Vehicle sent for maintenance
7	Unknown

  - Currently populating the database with a Transport Reason record is handled manually by the database administrator. In the future we would like the application to have the necessary features for our business to be able to manage the transport reason data. This is not an immediate requirement out of the gate but should be targeted as part of a future upgrade.

- **Transport Status:**

  - UI screens that require the use of the **Transport Status** field must be *prepopulated* with the list of transport status data from **DATABASE**. Based on the business requirements, the current list of transport status options are as follows:

Transport Status ID	*Transport Status Description*
1	Transport completed
2	On route to pick up location.
3	On route from pick up location
4	At pickup location. In progress (Loading etc.)
5	Pickup location delay
7	Unknown

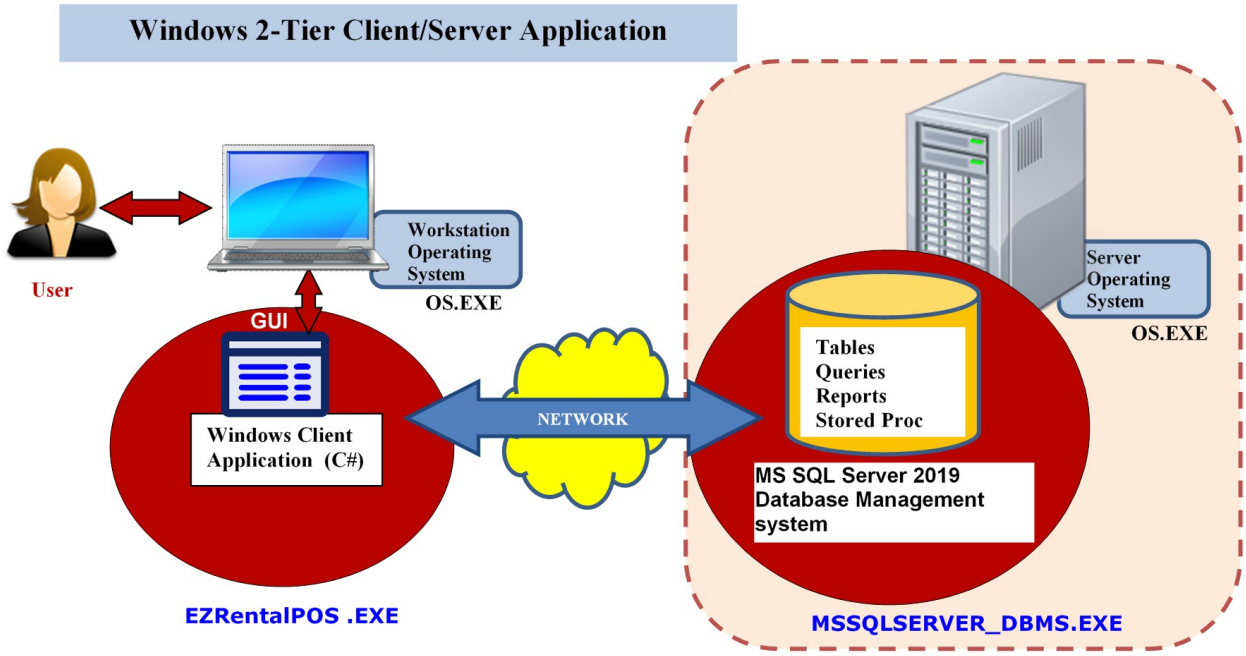
  - Currently populating the database with a transport status record is handled manually by the database administrator. In the future we would like the application to have the necessary features for our business to be able to manage the transport status data. This is not an immediate requirement out of the gate but should be targeted as part of a future upgrade.

APPLICATION PHYSICAL & TECHNICAL ARCHITECTURE

The targeted applications architecture and components for the final implemented application are the Two-Tiered Windows-Client Client/Server Application and Three-Tired Web-based Client/Server which both share a Database Tier.

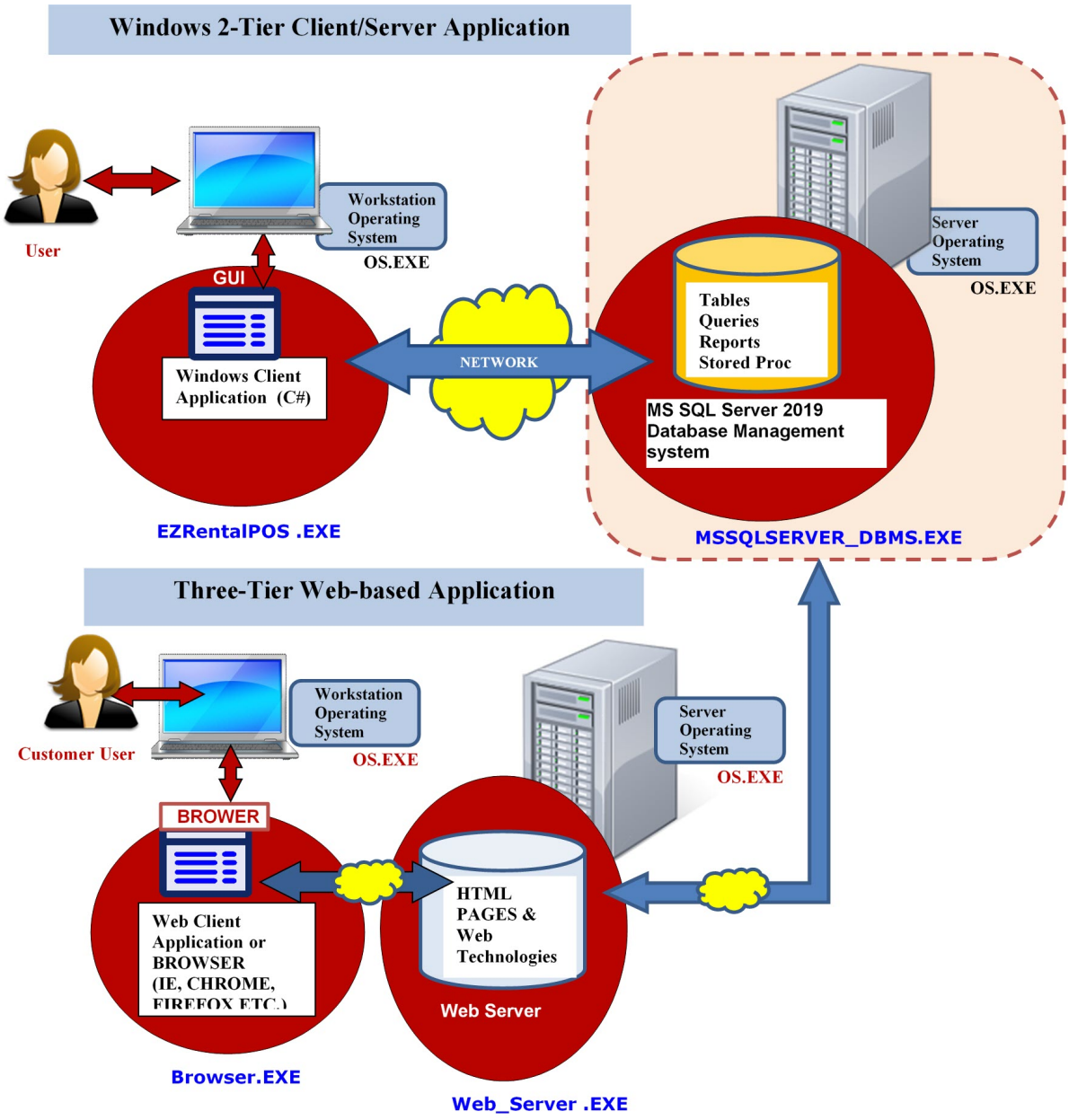
▪ **Two-Tiered Windows-Client Client/Server Application** – Front-line workers such as customer service desk in store branches, airports etc., in addition to other support personnel such as service centers employees, inventory etc., are to use this Windows-based client application for speed and performance.

Below is a pictorial diagram of this Two-Tiered Windows-Client client/server architecture used for the final software product:



▪ **Three-Tiered Web-based Client/Server** – This Web Application named EZRentalCar.com, targeted for customers who will rent a car online, in addition to the day-to-day activities of our business and office workers personnel via a Browser Application.

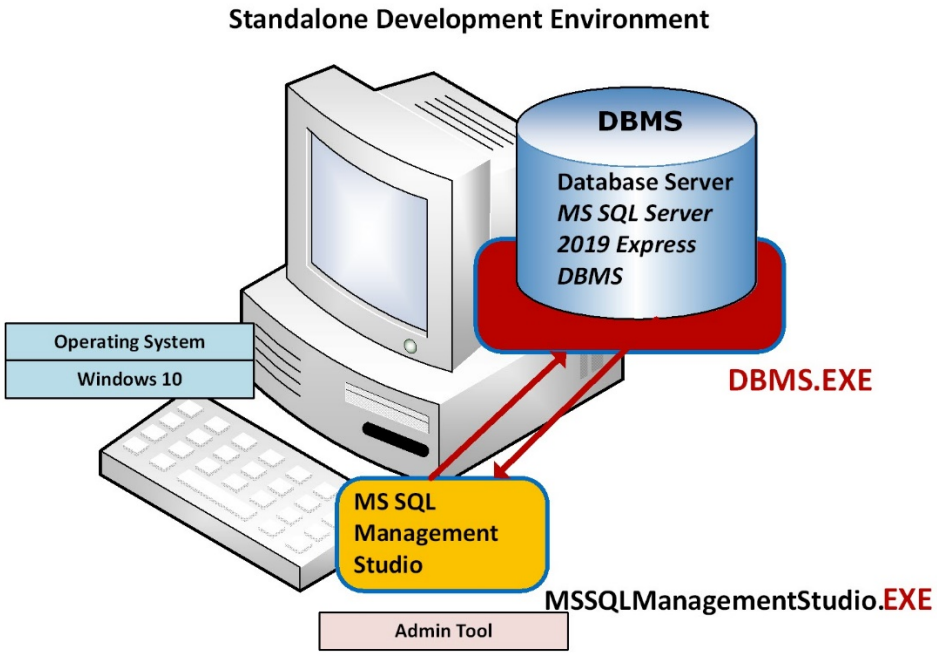
Below is a pictorial diagram of this Three-Tiered Web-based Client/Server architecture and shows both Tiers connected to the same **Database Tier** used for the final software product:



DATABASE MANAGEMENT PHYSICAL ARCHITECTURE

This section shows the planned Database Management System & Development Environment Overview. As stated before, this Database Tier will be used for both the two-tier and three- tiered architectures.

Database Tier – The Database Management System (DBMS) used is Microsoft SQL Server 2019 Express Edition since this is the standard DBMS used at EZRental Inc, along with Microsoft SQL Server Management Studio.



PROJECT MANAGEMENT METHODOLOGY

This section explains the planned project methodologies used to develop the project. A combination of the Waterfall Project Management Methodology & Agile Project Management Methodology was used for the Auto Rental Management System.

Project Management Methodology

A structured method, approach, process, technique, tools etc., to accomplish a project successfully, flexible to unexpected changes and delivered within the expected timelines.

Waterfall Methodology Phases

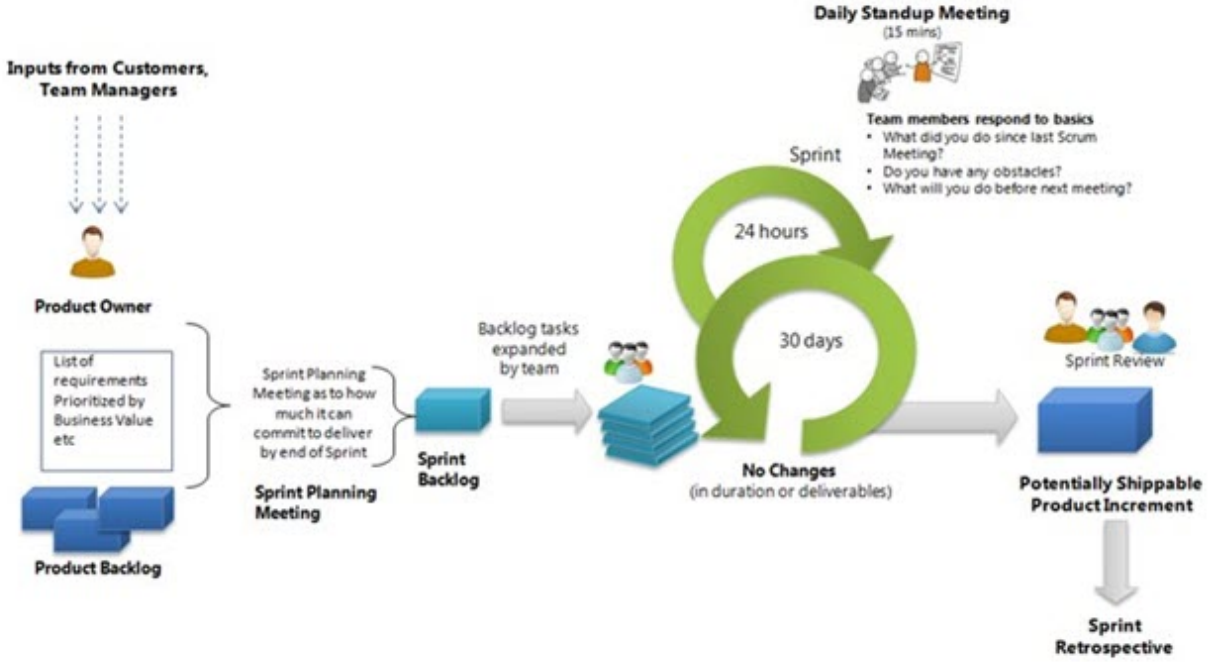
1. Project requirements are discussed and listed
2. Project is broken into structured method of phases.
3. Input to project first phase is the list of features (e.g., 20 features in new app business needed).
4. Each phase is executed sequentially until project completes at the expected timelines.
5. Output of project is the expected input implemented successfully (e.g., with 20 working features).

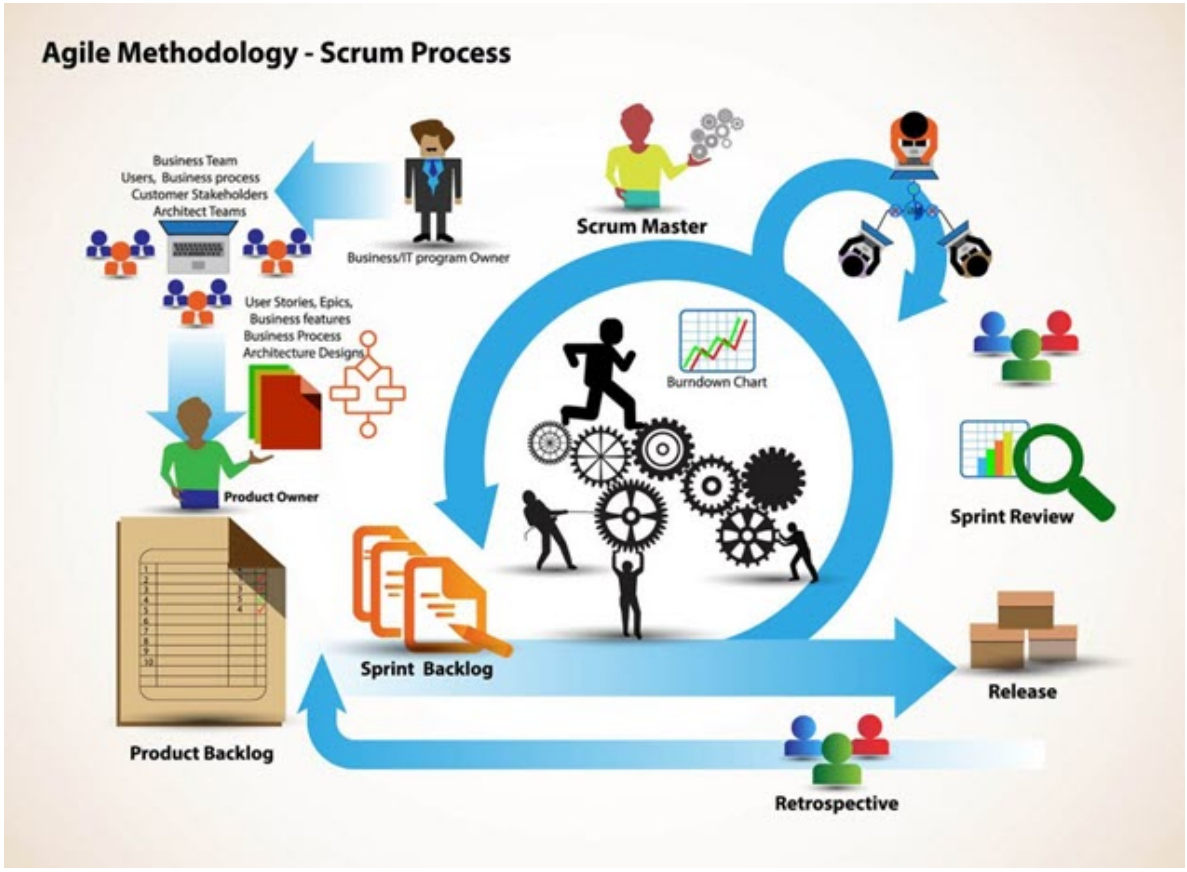
What is Agile Methodology

Agile is the evolution of Waterfall. Another Project Management Methodology that considers unexpected changes during the lifecycle of a project to ensure the project is delivered within the expected timelines.

Like the Waterfall Methodology, project requirements are discussed, listed, and broken into structured phases. Input for the project is the list of features, such as 20 features needed in the new app. This is like Waterfall but there is a strong focus on the list of features called backlog. However, the project is driven by subsets of the feature list not the entire backlog. The total list of features is divided into blocks of features to implement in increments. This is very different to Waterfall, since in Agile only subsets of the backlog or feature lists are implemented and delivered to users in increments, instead of the full list. Each phase is executed sequentially but repeated for each subset of the feature list. The repeated blocks of features are called sprints, and the output of each sprint is a subset list of features implemented and delivered to users. Only when all sprints (repetition, execution, and delivered product) and all the features in the backlog are completed within the expected timelines is the project considered finished.

Agile Scrum Methodology





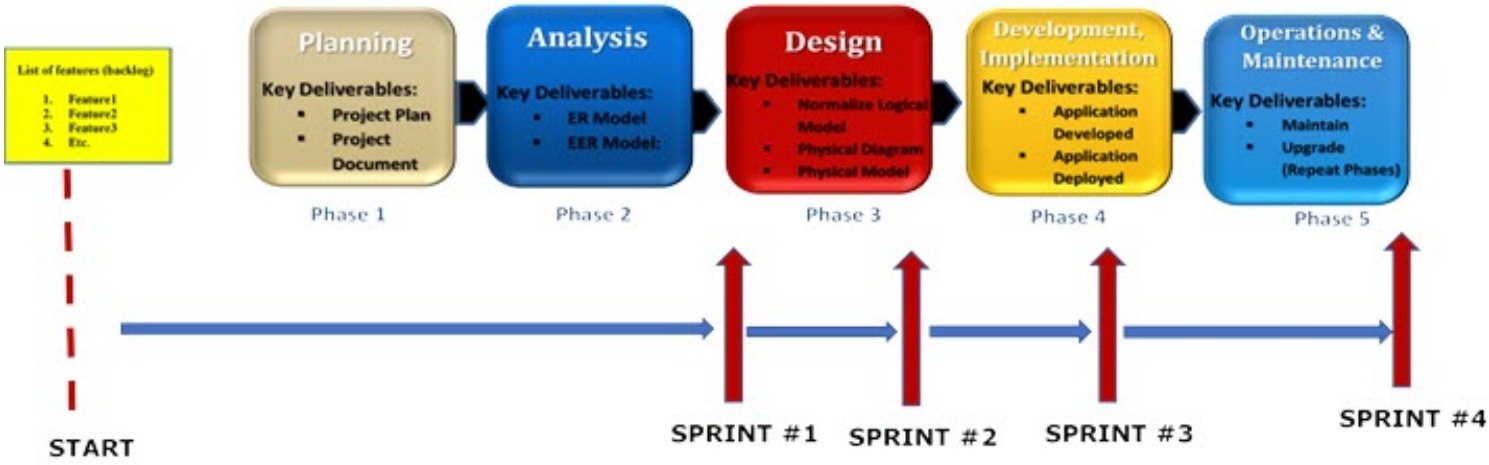
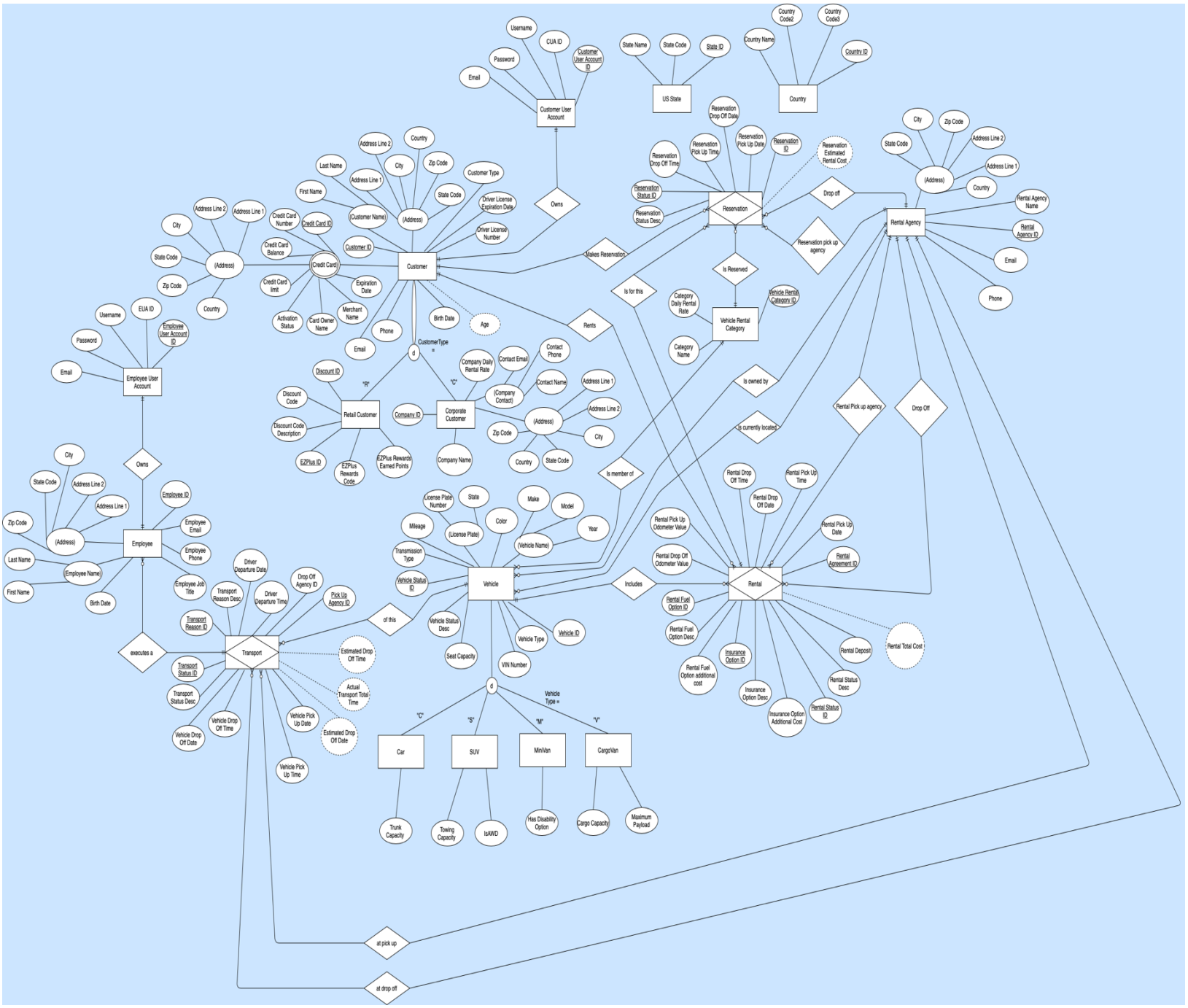


Illustration of Waterfall and Agile Methodologies used in conjunction

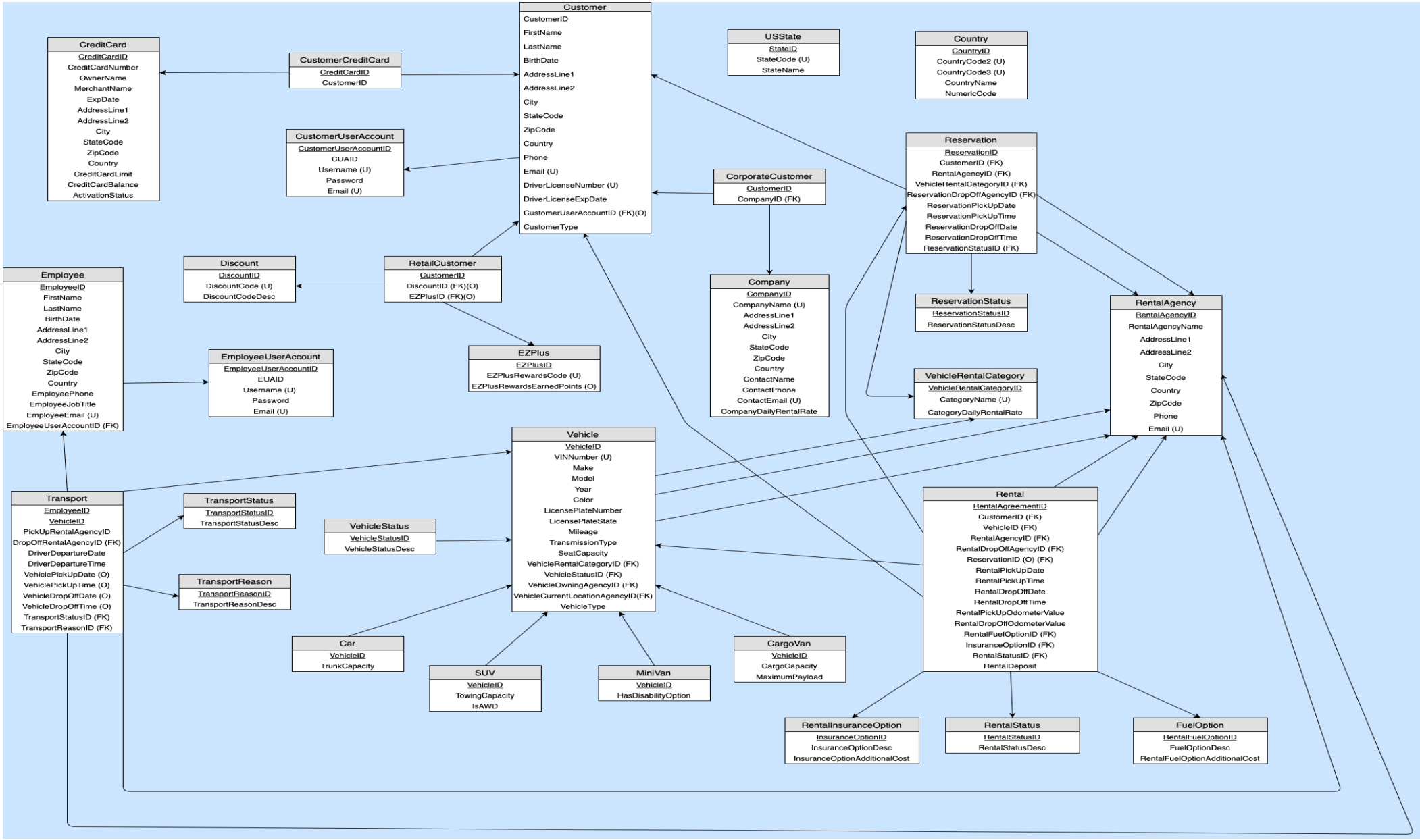
ER/EER Conceptual Model

This section shows the Extended Entity-Relationship (EER) Conceptual Model for the Auto Rental Management System. It was derived from the interview the Business Analyst gathered from the stakeholders at EZRental Inc. This model was created with all the Entities & Relationships based on the data needed in the database and how they relate to each other.



NORMALIZED LOGICAL MODEL

The normalized logical model below was derived from the EER Model. All entities are included with their relationships, attributes, primary and foreign keys.



PHYSICAL MODEL DATA DICTIONARY

The normalized logical model was used to design the Data Dictionary in this section. The dictionary presents all the metadata for each entity.

CUSTOMER						
Attribute/Column Name	**Data Type**	**MS SQL Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
CustomerID	Numeric	INT	Y	Default size of INT data type	NOT NULL IDENTITY(1111,1) PRIMARY KEY	Auto-generated Integer Identity PK starting at 1111
FirstName	String	NVARCHAR(X)	Y	50	NOT NULL	First name of customer
LastName	String	NVARCHAR(X)	Y	50	NOT NULL	Last name of customer
BirthDate	Date	DATE	Y	YYYY/MM/DD	NOT NULL	Date of Birth
AddressLine1	String	NVARCHAR(X)	Y	50	NOT NULL	House number & street part 1
AddressLine2	String	NVARCHAR(X)	Y	50	NOT NULL	House number & street part 2
City	String	NVARCHAR(X)	Y	60	NOT NULL	City name
StateCode	Character	CHAR(X)	Y	2	NOT NULL	State code. US state only
ZipCode	String	VARCHAR(X)	Y	12	NOT NULL	Zip Code
Country	String	VARCHAR(X)	Y	60	NOT NULL	Country name
Phone	String	VARCHAR(X)	Y	20	NOT NULL	Phone – international scope
Email	String	VARCHAR(X)	Y	50	NOT NULL UNIQUE	Email address. International scope
DriverLicenseNumber	String	VARCHAR(X)	Y	25	NOT NULL UNIQUE	Driver license number. International scope
DriverLicenseExpDate	Date	DATE	Y	YYYY/MM/DD	NOT NULL	US license exp date
CustomerUserAccountID	Numeric	INT	N	Default size of INT data type	NULL FOREIGN KEY	Customer User Account ID; FK of CustomerUserAccount table
CustomerType	Character	CHAR(X)	Y	1	NOT NULL	Value of Retail or Corporate Customer (R or C)

RETAILCUSTOMER						
Attribute/Column Name	**General Data Type Name**	**MS SQL Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
CustomerID	Numeric	INT	Y	Default size of INT data type	NOT NULL PRIMARY KEY	Customer ID and PK, FK of Customer table
DiscountID	Numeric	INT	N	Default size of INT data type	NULL FOREIGN KEY	ID of the discount, FK of Discount table
EZPlusID	Numeric	INT	N	Default size of INT data type	NULL FOREIGN KEY	ID of EZ plus rewards, FK of EZPlus table

COPORATECUSTOMER						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
CustomerID	Numeric	INT	Y	Default size of INT data type	NOT NULL PRIMARY KEY	Customer ID and PK, FK of Customer table
CompanyID	Numeric	INT	Y	9999999	NOT NULL FOREIGN KEY CHECK(CompanyID BETWEEN 1 AND 9999999)	Id of a company customer. FK of Company table

DISCOUNT						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
DiscountID	Numeric	INT	Y	Default size of INT data type	NOT NULL IDENTITY(1111,1) PRIMARY KEY	Auto-generated Integer Identity PK starting at 1111
DiscountCode	Character	CHAR(X)	Y	8	NOT NULL UNIQUE	Code for discount with 3 letters then 5 numbers
DiscountCodeDesc	String	VARCHAR(X)	Y	150	NOT NULL	Description of discount code

CREDITCARD						
Attribute/Column Name	**General Data Type Name**	**MS SQL Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
CreditCardID	Numeric	INT	Y	Default size of INT data type	NOT NULL IDENTITY(1,1) PRIMARY KEY	Auto-generated Integer Identity PK starting at 1
CreditCardNumber	String	VARCHAR(X)	Y	128	NOT NULL	Credit card number of owner card
OwnerName	String	NVARCHAR(X)	Y	80	NOT NULL	Name of credit card owner, first + last
MerchantName	String	VARCHAR(X)	Y	20	NOT NULL	Name of merchant
ExpDate	Date	DATE	Y	YYYY/MM/DD	NOT NULL	Credit card expiration date
AddressLine1	String	NVARCHAR(X)	Y	50	NOT NULL	House number & street part 1
AddressLine2	String	NVARCHAR(X)	Y	50	NOT NULL	House number & street part 2
City	String	NVARCHAR(X)	Y	60	NOT NULL	City name
StateCode	Character	CHAR(X)	Y	2	NOT NULL	US state code only
ZipCode	String	VARCHAR(X)	Y	12	NOT NULL	Zip Code
Country	String	VARCHAR(X)	Y	60	NOT NULL	Country column with international scope
CreditCardLimit	String	DECIMAL(X,Y)	Y	X=9 Y=2	NOT NULL	CC cash limit. X = total number of digits, Y = decimal places
CreditCardBalance	String	DECIMAL(X,Y)	Y	X=9 Y=2	NOT NULL	Credit card balance
ActivationStatus	Exact Numeric	BIT	Y	1	NOT NULL	Is card activated? 1 true, 0 false

COMPANY						
Attribute/Column Name	**General Data Type Name**	**MS SQL Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
CompanyID	Numeric	INT	Y	9999999	NOT NULL PRIMARY KEY CHECK (CompanyID BETWEEN 1 AND 9999999)	Unique identifier for a company instance.
CompanyName	String	NVARCHAR(X)	Y	50	NOT NULL UNIQUE	Name of company
AddressLine1	String	NVARCHAR(X)	Y	50	NOT NULL	House number & street part 1
AddressLine2	String	NVARCHAR(X)	Y	50	NOT NULL	House number & street part 2
City	String	NVARCHAR(X)	Y	60	NOT NULL	City Name
StateCode	Character	CHAR(X)	Y	2	NOT NULL	US state code, only
ZipCode	String	VARCHAR(X)	Y	12	NOT NULL	Zip Code
Country	String	VARCHAR(X)	Y	60	NOT NULL	Country
ContactName	String	NVARCHAR(X)	Y	100	NOT NULL	First and last name of contact
ContactPhone	String	VARCHAR(X)	Y	20	NOT NULL	Contact Phone - international scope
ContactEmail	String	VARCHAR(X)	Y	75	NOT NULL UNIQUE	Contact Email address, International scope
CompanyDailyRentalRate	Numeric	DECIMAL(X,Y)	Y	X=6 Y=2	NOT NULL	Company daily rental rate

CUSTOMERCREDITCARD						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
CreditCardID	Numeric	INT	Y	Default size of INT data type	NOT NULL PRIMARY KEY	ID of credit card, composite PK combined with CustomerID, FK of credit card
CustomerID	Numeric	INT	Y	Default size of INT data type	NOT NULL PRIMARY KEY	ID of customer, composite PK combined with CreditCardID, FK of Customer table

CUSTOMERUSERACCOUNT						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
CustomerUserAccountID	Numeric	INT	Y	Default size of INT data type	NOT NULL IDENTITY(1,1) PRIMARY KEY	Auto-generated Integer Identity PK starting at 1
CUAID	String	UNIQUEIDENTIFIER	Y	GUID-Auto generated 36 characters long	NOT NULL DEFAULT NEWID()	Customer user account ID
Username	String	VARCHAR(X)	Y	50	NOT NULL UNIQUE	Customer username
Password	String	VARCHAR(X)	Y	60	NOT NULL	Customer password
Email	String	VARCHAR(X)	Y	75	NOT NULL UNIQUE	Email address, international scope

EZPLUS						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
EZPlusID	Numeric	INT	Y	Default size of INT data type	NOT NULL IDENTITY(1111,1) PRIMARY KEY	Auto-generated Integer Identity PK starting at 1111
EZPlusRewardsCode	Character	CHAR(X)	Y	8	NOT NULL UNIQUE	Code for EZplus Rewards with 3 letters then 5 numbers
EZPlusRewardsEarnedPoints	Numeric	INT	N	Default size of INT data type	NULL	Number of points rewarded per code

VEHICLESTATUS						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
VehicleStatusID	Numeric	TINYINT	Y	9	NOT NULL PRIMARY KEY CHECK (VehicleStatusID BETWEEN 1 AND 9)	Unique identifier for a company instance
VehicleStatusDesc	String	VARCHAR(X)	Y	50	NOT NULL	Description of vehicle status

VEHICLE						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
VehicleID	Numeric	INT	Y	Default size of INT data type	NOT NULL IDENTITY(1111,1) PRIMARY KEY	Auto-generated Integer Identity PK starting at 1111
VINNumber	Character	CHAR(X)	Y	17	NOT NULL UNIQUE	Unique vehicle Identification Number
Make	String	VARCHAR(X)	Y	40	NOT NULL	Make of vehicle
Model	String	VARCHAR(X)	Y	40	NOT NULL	Model of vehicle
Year	Numeric	SMALLINT	Y	Default size SMALLINT	NOT NULL	Year of vehicle
Color	String	VARCHAR(X)	Y	20	NOT NULL	Color of vehicle
LicensePlateNumber	String	NVARCHAR(X)	Y	15	NOT NULL	License plate number
LicensePlateState	Character	CHAR(X)	Y	2	NOT NULL	Plate issued - US state name only
Mileage	Numeric	INT	Y	300000	NOT NULL CHECK (Mileage BETWEEN 1 AND 300000)	Total vehicle mileage
TransmissionType	String	VARCHAR(X)	Y	35	NOT NULL	Vehicle transmission type
SeatCapacity	Numeric	TINYINT	Y	Default size TINYINT	NOT NULL	Vehicle seat capacity
VehicleRentalCategoryID	Numeric	TINYINT	Y	25	NOT NULL FOREIGN KEY CHECK(VehicleRentalCategoryID BETWEEN 1 AND 25)	Id of vehicle category, FK of VehicleRentalCategory table
VehicleStatusID	Numeric	TINYINT	Y	9	NOT NULL FOREIGN KEY CHECK(VehicleStatusID BETWEEN 1 AND 9)	Id of vehicle status, FK of VehicleStatus table
VehicleOwningAgencyID	Numeric	INT	Y	Default size of INT data type	NOT NULL FOREIGN KEY	Id of rental agency that owns vehicle, FK of Rental Agency table
VehicleCurrentLocationAgencyID	Numeric	INT	Y	Default size of INT data type	NOT NULL FOREIGN KEY	Id of agency that currently has vehicle, FK of RentalAgency table
VehicleType	Character	CHAR(X)	Y	1	NOT NULL	Vehicle type Value = C, S, M, or V

VEHICLERENTALCATEGORY						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
VehicleRentalCategoryID	Numeric	TINYINT	Y	25	NOT NULL PRIMARY KEY CHECK (VehicleRentalCategory BETWEEN 1 AND 25)	Unique identifier for a vehicle rental category instance
CategoryName	String	VARCHAR (X)	Y	35	NOT NULL UNIQUE	Name of the vehicle rental category
CategoryDailyRentalRate	Numeric	DECIMAL (X,Y)	Y	X=5 Y=2	NOT NULL	Vehicle category daily rental rate. Maximum rate currently of 199.99

MINIVAN						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
VehicleID	Numeric	INT	Y	Default size of INT	NOT NULL PRIMARY KEY	Id of vehicle, FK of Vehicle table
HasDisabilityOption	Numeric	BIT	Y	1	NOT NULL	Does Van offer disability option? 1 or 0

CAR						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
VehicleID	Numeric	INT	Y	Default size of INT	NOT NULL PRIMARY KEY	Id of vehicle, FK of Vehicle table
TrunkCapacity	Numeric	Decimal (X,Y)	Y	X=4 Y=1	NOT NULL	Capacity in cubic ft.

SUV						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
VehicleID	Numeric	INT	Y	Default size of INT	NOT NULL PRIMARY KEY	Id of vehicle, FK of Vehicle table
TowingCapacity	Numeric	SMALLINT	Y	Default size SMALLINT	NOT NULL	Towing Capacity in lbs.
IsAWD	Numeric	BIT	Y	1	NOT NULL	Is Vehicle All wheel Drive? 1 or 0

CARGOVAN						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
VehicleID	Numeric	INT	Y	Default size of INT	NOT NULL PRIMARY KEY	Id of vehicle, FK of Vehicle table
CargoCapacity	Numeric	DECIMAL (X,Y)	Y	X=4 Y=1	NOT NULL	Cargo capacity in cubic ft.
MaximumPayload	Numeric	SMALLINT	Y	Default size SMALLINT	NOT NULL	Maximum Payload in lbs.

RESERVATION						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required ?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
ReservationID	Numeric	INT	Y	Default size of INT data type	NOT NULL IDENTITY(1,1) PRIMARY KEY	Auto-generated Integer Identity PK starting at 1
CustomerID	Numeric	INT	Y	Default size of INT data type	NOT NULL FOREIGN KEY	Customer Id, FK of Customer table
RentalAgencyID	Numeric	INT	Y	Default size of INT data type	NOT NULL FOREIGN KEY	Rental Agency ID, FK of Rental Agency
VehicleRentalCategoryID	Numeric	TINYINT	Y	25	NOT NULL FOREIGN KEY CHECK(VehicleRentalCategoryID BETWEEN 1 AND 25)	Id of Vehicle Rental Category, FK of VehicleRentalCategory table
ReservationDropOffAgency ID	Numeric	INT	Y	Default size of INT data type	NOT NULL FOREIGN KEY	Id of agency where vehicle reserved will be dropped off, FK of RentalAgency table
ReservationPickUplDate	Date	DATE	Y	YYYY/MM/DD	NOT NULL	Reservation pick up date
ReservationPickUpTime	Numeric	TIME(X)	Y	0 HH:MM:SS	NOT NULL	Pick up time of reservation
ReservationDropOffDate	Date	DATE	Y	YYYY/MM/DD	NOT NULL	Reservation drop off date
ReservationDropOffTime	Numeric	TIME(X)	Y	0 HH:MM:SS	NOT NULL	Reservation drop off time
ReservationStatusID	Numeric	TINYINT	Y	9	NOT NULL FOREIGN KEY CHECK(ReservationStatusID BETWEEN 1 AND 9)	Reservation status ID, FK of ReservationStatus table)

RENTAL						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
RentalAgreementID	Numeric	INT	Y	Default size of INT data type	NOT NULL IDENTITY(1,1) PRIMARY KEY	Auto-generated Integer Identity PK starting at 1
CustomerID	Numeric	INT	Y	Default size of INT data type	NOT NULL FOREIGN KEY	Customer Id FK of Customer table
VehicleID	Numeric	INT	Y	Default size of INT	NOT NULL FOREIGN KEY	Id of vehicle FK of Vehicle table
RentalAgencyID	Numeric	INT	Y	Default size of INT data type	NOT NULL FOREIGN KEY	Id of rental agency, FK of RentalAgency table
RentalDropOffAgencyID	Numeric	INT	Y	Default size of INT data type	NOT NULL FOREIGN KEY	Agency ID where vehicle rented will be dropped off, FK of RentalAgency table
ReservationID	Numeric	INT	N	Default size of INT data type	NULL FOREIGN KEY	Reservation ID, FK of Reservation table
RentalPickUpDate	Date	DATE	Y	YYYY/MM/DD	NOT NULL	Rental pick up date
RentalPickUpTime	Numeric	TIME(0)	Y	0 HH:MM:SS	NOT NULL	Pick Up time of rental
RentalDropOffDate	Date	DATE	Y	YYYY/MM/DD	NOT NULL	Rental drop off date
RentalDropOffTime	Numeric	TIME(0)	Y	0 HH:MM:SS	NOT NULL	Rental drop off time
RentalPickUpOdometerValue	Numeric	INT	Y	300000	NOT NULL CHECK (RentalPickUp OdometerValue BETWEEN 0 AND 300000)	Odometer value upon rental pick up
RentalDropOffOdometerValue	**Numeric**	INT	Y	300000	NOT NULL (RentalDropOff OdometerValue BETWEEN 0 AND 300000)	Odometer value upon rental drop off
RentalFuelOptionID	Numeric	TINYINT	Y	5	NOT NULL FOREIGN KEY CHECK (RentaFuelOptionID BETWEEN 1 AND 5)	Id of rental fuel option. (Foreign key of FuelOption)
InsuranceOptionID	Numeric	TINYINT	Y	8	NOT NULL FOREIGN KEY CHECK(InsuranceOptionID BETWEEN 1 AND 8)	Id of insurance option. (Foreign key of RentalInsurance Option table)
RentalStatusID	Numeric	TINYINT	Y	10	NOT NULL FOREIGN KEY CHECK(RentalStatusID BETWEEN 1 AND 10)	Id of rental status. FK of RentalStatus table
RentalDeposit	Numeric	DECIMAL (X,Y)	Y	X=7 Y=2	NOT NULL	Deposit amount. Rental period + 25% of the rental

RESERVATIONSTATUS						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
ReservationStatusID	Numeric	TINYINT	Y	9	NOT NULL PRIMARY KEY CHECK (ReservationStatusID BETWEEN 1 AND 9)	Unique identifier for a reservation status instance.
ReservationStatusDesc	String	VARCHAR(X)	Y	30	NOT NULL	Description of reservation status

RENTALINSURANCEOPTION						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
InsuranceOptionID	Numeric	TINYINT	Y	8	NOT NULL PRIMARY KEY CHECK (InsuranceOptionID BETWEEN 1 AND 8)	Unique identifier for an insurance option instance.
InsuranceOptionDesc	String	VARCHAR (MAX)	Y	Default MAX size	NOT NULL	Rental insurance option description
InsuranceOptionAdditionalCost	Numeric	DECIMAL (X,Y)	Y	X=5 Y=2	NOT NULL	Rental insurance cost per day

RENTALSTATUS						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
RentalStatusID	Numeric	TINYINT	Y	10	NOT NULL Primary Key CHECK (RentalStatusID BETWEEN 1 AND 10)	The ID number of the rental status.
RentalStatusDesc	String	VARCHAR(X)	Y	40	NOT NULL	Description of rental status

FUELOPTION						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
RentalFuelOptionID	Numeric	TINYINT	Y	5	NOT NULL PRIMARY KEY CHECK (RentaFuel OptionID BETWEEN 1 AND 5)	The ID number of rental fuel option
FuelOptionDesc	String	VARCHAR (X)	Y	100	NOT NULL	Description of fuel option
RentalFuelOptionAdditionalCost	String	VARCHAR (X)	Y	100	NOT NULL	Additional fuel cost calculation description

EMPLOYEEUSERACCOUNT						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
EmployeeUserAccountID	Numeric	INT	Y	Default size of INT data type	NOT NULL IDENTITY(1,1) PRIMARY KEY	Auto-generated Integer Identity PK starting at 1
EUAID	String	UNIQUEIDENTIFIER	Y	GUID-Auto generated 36 characters long	NOT NULL DEFAULT NEWID()	Employee user account ID
Username	String	VARCHAR(X)	Y	50	NOT NULL UNIQUE	Customer username
Password	String	VARCHAR(X)	Y	60	NOT NULL	Customer password
Email	String	VARCHAR (X)	Y	75	NOT NULL UNIQUE	Email address. International scope

EMPLOYEE						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
EmployeeID	Numeric	INT	Y	Default size of INT data type	NOT NULL IDENTITY(1111,1) PRIMARY KEY	Auto-generated Integer Identity primary key starting at 1111
FirstName	String	NVARCHAR(X)	Y	50	NOT NULL	First name of employee
LastName	String	NVARCHAR(X)	Y	50	NOT NULL	Last name of employee
BirthDate	Date	DATE	Y	YYYY/MM/DD	NOT NULL	Date of Birth
AddressLine1	String	NVARCHAR(X)	Y	50	NOT NULL	House number & street part 1
AddressLine2	String	NVARCHAR(X)	Y	50	NOT NULL	House number & street part 2
City	String	NVARCHAR(X)	Y	60	NOT NULL	City name
StateCode	Character	CHAR(X)	Y	2	NOT NULL	State code. US state only
ZipCode	String	VARCHAR(X)	Y	12	NOT NULL	Zip Code
Country	String	VARCHAR(X)	Y	60	NOT NULL	Country column with international scope
EmployeePhone	String	VARCHAR(X)	Y	20	NOT NULL	Employee Phone – international scope
EmployeeJobTitle	String	VARCHAR(X)	Y	50	NOT NULL	Employee job title
EmployeeEmail	String	VARCHAR(X)	Y	75	NOT NULL UNIQUE	Employee Email address. International scope
EmployeeUserAccountID	Numeric	INT	Y	Default size of INT data type	NOT NULL FOREIGN KEY	Employee User Account ID; FK of EmployeeUserAccount table

RENTALAGENCY						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
RentalAgencyID	Numeric	INT	Y	Default size of INT data type	NOT NULL IDENTITY(1,1) PRIMARY KEY	Auto-generated Integer Identity primary key starting at 1
RentalAgencyName	String	NVARCHAR(X)	Y	100	NOT NULL	Rental Agency Name
AddressLine1	String	NVARCHAR(X)	Y	50	NOT NULL	House number & street part 1
AddressLine2	String	NVARCHAR(X)	Y	50	NOT NULL	House number & street part 2
City	String	NVARCHAR(X)	Y	60	NOT NULL	City Name
StateCode	Character	CHAR(X)	Y	2	NOT NULL	US state code, only
Country	String	VARCHAR(X)	Y	60	NOT NULL	Country column with international scope
ZipCode	String	VARCHAR(X)	Y	12	NOT NULL	Zip Code
Phone	String	VARCHAR(X)	Y	20	NOT NULL	Agency Phone – international scope
Email	String	VARCHAR(X)	Y	75	NOT NULL UNIQUE	Agency Email address. International scope

TRANSPORTSTATUS						
Attribute/Column Name	**General Data Type Name**	**MS SQL Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
TransportStatusID	Numeric	TINYINT	Y	9	NOT NULL PRIMARY KEY CHECK (TransportStatusID BETWEEN 1 AND 9)	Transport Status ID
TransportStatusDesc	String	VARCHAR(X)	Y	60	NOT NULL	Transport status description

TRANSPORTREASON						
Attribute/Column Name	**General Data Type Name**	**MS SQL Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
TransportReasonID	Numeric	TINYINT	Y	9	NOT NULL PRIMARY KEY CHECK (TransportReasonID BETWEEN 1 AND 9)	Transport reason ID
TransportReasonDesc	String	VARCHAR(X)	Y	60	NOT NULL	Transport reason description

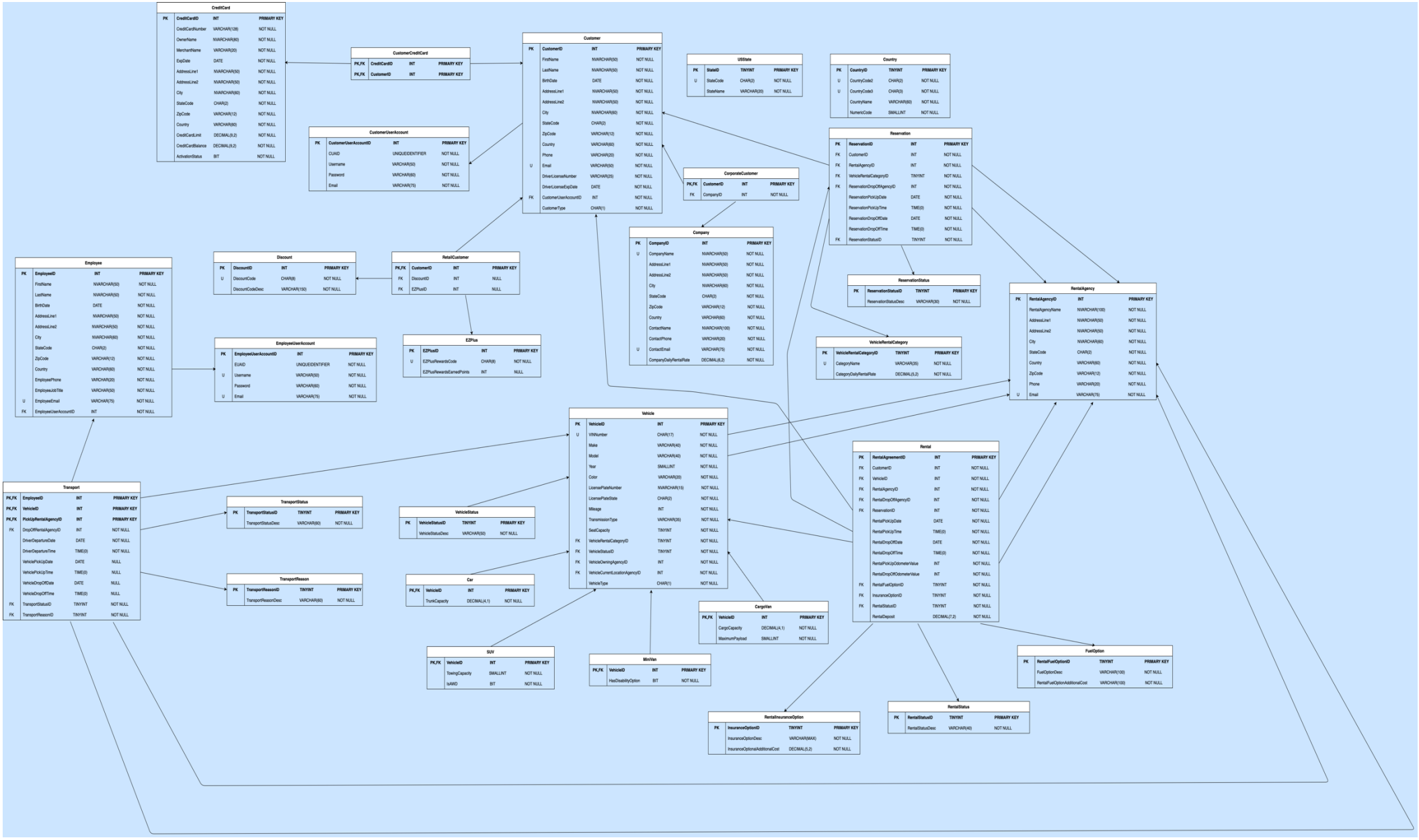
TRANSPORT						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
EmployeeID	Numeric	INT	Y	Default size of INT data type	NOT NULL PRIMARY KEY	Employee ID, composite key, FK of Employee table
VehicleID	Numeric	INT	Y	Default size of INT data type	NOT NULL PRIMARY KEY	Vehicle ID, composite key, FK of Vehicle table
PickUpRentalAgencyID	Numeric	INT	Y	Default size of INT data type	NOT NULL PRIMARY KEY	Pick up rental agency ID, composite key, FK of RentalAgency table
DropOffRentalAgencyID	Numeric	INT	Y	Default size of INT data type	NOT NULL FOREIGN KEY	Drop off rental agency ID, FK of RentalAgency table
DriverDepartureDate	Date	DATE	Y	YYYY/MM/DD	NOT NULL	Driver departure date
DriverDepartureTime	Numeric	TIME(X)	Y	0 HH:MM:SS	NOT NULL	Driver departure time
VehiclePickUpDate	Date	DATE	N	YYYY/MM/DD	NULL	Vehicle pick up date
VehiclePickUpTime	Numeric	TIME(X)	N	0 HH:MM:SS	NULL	Vehicle pick up time
VehicleDropOffDate	Date	DATE	N	YYYY/MM/DD	NULL	Vehicle drop off date
VehicleDropOffTime	Numeric	TIME(X)	N	0 HH:MM:SS	NULL	Vehicle drop off time
TransportStatusID	Numeric	TINYINT	Y	9	NOT NULL FOREIGN KEY CHECK (TransportStatusID BETWEEN 1 AND 9)	Transport Status ID, FK of TransportStatus table
TransportReasonID	Numeric	TINYINT	Y	9	NOT NULL FOREIGN KEY CHECK (TransportReasonID BETWEEN 1 AND 9)	Transport reason ID, FK of TransportReason table

COUNTRY						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
CountryID	Numeric	TINYINT	Y	Default size of TINYINT	NOT NULL PRIMARY KEY	The ID number of a country.
CountryCode2	Character	CHAR(X)	Y	2	NOT NULL UNIQUE	Country code 2
CountryCode3	Character	CHAR(X)	Y	3	NOT NULL UNIQUE	Country code 3
CountryName	String	VARCHAR(X)	Y	60	NOT NULL	Country name
NumericCode	Numeric	SMALLINT	Y	999	NOT NULL CHECK(NumericCode BETWEEN 1 AND 999)	Country numeric code

USSTATE						
Attribute/Column Name	**General Data Type Name**	**MS SQL Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
StateID	Numeric	TINYINT	Y	51	NOT NULL PRIMARY KEY CHECK (StateID BETWEEN 1 AND 51)	The ID number of a US state.
StateCode	Character	CHAR(X)	Y	2	NOT NULL UNIQUE	Unique US state code
StateName	String	VARCHAR (X)	Y	20	NOT NULL	US state name only

PHYSICAL MODEL SCHEMA DIAGRAM

The Physical Model is a combination of both the Normalized Logical Model and the Data Dictionary.



DEVELOPMENT & IMPLEMENTATION

Using the physical model schema diagram, the script below was created, which contains all the data definition language (DDL) statements required to create the database and its entities. As previously stead, Microsoft SQL Express and Microsoft SQL Management Studio were used to create this script.

```
--CREATE THE DATABASE
CREATE DATABASE EZRentalDB;

--SET DEFAULT DATABASE TO USE
use EZRentalDB;

--CREATE CREDITCARD TABLE
CREATE TABLE CreditCard
(
    CreditCardID        INT             NOT NULL IDENTITY(1,1),
    CreditCardNumber    VARCHAR(128)    NOT NULL,
    OwnerName           NVARCHAR(80)    NOT NULL,
    MerchantName        VARCHAR(20)     NOT NULL,
    ExpDate             DATE            NOT NULL,
    AddressLine1        NVARCHAR(50)    NOT NULL,
    AddressLine2        NVARCHAR(50)    NOT NULL,
    City                NVARCHAR(60)    NOT NULL,
    StateCode           CHAR(2)         NOT NULL,
    ZipCode             VARCHAR(12)     NOT NULL,
    Country             VARCHAR(60)     NOT NULL,
    CreditCardLimit     DECIMAL(9,2)    NOT NULL,
    CreditCardBalance   DECIMAL(9,2)    NOT NULL,
    ActivationStatus    BIT             NOT NULL,

    PRIMARY KEY(CreditCardID)
);

--CREATE EMPLOYEE USER ACCOUNT TABLE
CREATE TABLE EmployeeUserAccount
(
    EmployeeUserAccountID   INT                     NOT NULL IDENTITY(1,1),
    EUAID                   UNIQUEIDENTIFIER        NOT NULL DEFAULT NEWID(),
    Username                VARCHAR(50)             NOT NULL UNIQUE,
    Password                VARCHAR(60)             NOT NULL,
    Email                   VARCHAR(75)             NOT NULL UNIQUE,

    PRIMARY KEY(EmployeeUserAccountID)
);
```

```sql
--CREATE EMPLOYEE TABLE
CREATE TABLE Employee
(
    EmployeeID              INT                 NOT NULL IDENTITY(1111,1),
    FirstName               NVARCHAR(50)        NOT NULL,
    LastName                NVARCHAR(50)        NOT NULL,
    BirthDate               DATE                NOT NULL,
    AddressLine1            NVARCHAR(50)        NOT NULL,
    AddressLine2            NVARCHAR(50)        NOT NULL,
    City                    NVARCHAR(60)        NOT NULL,
    StateCode               CHAR(2)             NOT NULL,
    ZipCode                 VARCHAR(12)         NOT NULL,
    Country                 VARCHAR(60)         NOT NULL,
    EmployeePhone           VARCHAR(20)         NOT NULL,
    EmployeeJobTitle        VARCHAR(50)         NOT NULL,
    EmployeeEmail           VARCHAR(75)         NOT NULL UNIQUE,
    EmployeeUserAccountID   INT                 NOT NULL,

    PRIMARY KEY(EmployeeID),

    CONSTRAINT FK_E_EmployeeUserAccountID
    FOREIGN KEY (EmployeeUserAccountID)
    REFERENCES EmployeeUserAccount(EmployeeUserAccountID)
);

--CREATE CUSTOMER USER ACCOUNT TABLE
CREATE TABLE CustomerUserAccount
(
    CustomerUserAccountID   INT                 NOT NULL IDENTITY(1,1),
    CUAID                   UNIQUEIDENTIFIER    NOT NULL DEFAULT NEWID(),
    Username                VARCHAR(50)         NOT NULL UNIQUE,
    Password                VARCHAR(60)         NOT NULL,
    Email                   VARCHAR(75)         NOT NULL UNIQUE,

    PRIMARY KEY(CustomerUserAccountID)
);

--CREATE CUSTOMER TABLE
CREATE TABLE Customer
(
    CustomerID              INT                     NOT NULL IDENTITY(1111,1),
    FirstName               NVARCHAR(50)            NOT NULL,
    LastName                NVARCHAR(50)            NOT NULL,
    BirthDate               DATE                    NOT NULL,
    AddressLine1            NVARCHAR(50)            NOT NULL,
    AddressLine2            NVARCHAR(50)            NOT NULL,
    City                    NVARCHAR(60)            NOT NULL,
    StateCode               CHAR(2)                 NOT NULL,
    ZipCode                 VARCHAR(12)             NOT NULL,
    Country                 VARCHAR(60)             NOT NULL,
    Phone                   VARCHAR(20)             NOT NULL,
    Email                   VARCHAR(50)             NOT NULL UNIQUE,
    DriverLicenseNumber     VARCHAR(25)             NOT NULL UNIQUE,
    DriverLicenseExpDate    DATE                    NOT NULL,
    CustomerUserAccountID   INT                     NULL,
    CustomerType            CHAR(1)                 NOT NULL,

    PRIMARY KEY(CustomerID),
```

```
    CONSTRAINT FK_C_CustomerUserAccountID
    FOREIGN KEY (CustomerUserAccountID)
    REFERENCES CustomerUserAccount(CustomerUserAccountID)
);

--CREATE CUSTOMER_CREDITCARD TABLE
CREATE TABLE CustomerCreditCard
(
    CreditCardID        INT         NOT NULL,
    CustomerID          INT         NOT NULL,

    PRIMARY KEY (CreditCardID, CustomerID),

    CONSTRAINT FK_CCC_CreditCardID
    FOREIGN KEY (CreditCardID)
    REFERENCES CreditCard(CreditCardID),

    CONSTRAINT FK_CCC_CustomerID
    FOREIGN KEY (CustomerID)
    REFERENCES Customer(CustomerID)
);

--CREATE DISCOUNT TABLE
CREATE TABLE Discount
(
    DiscountID          INT             NOT NULL IDENTITY(1111,1),
    DiscountCode        CHAR(8)         NOT NULL UNIQUE,
    DiscountCodeDesc    VARCHAR(150)    NOT NULL,

    PRIMARY KEY (DiscountID)
);

--CREATE EZPLUS TABLE
CREATE TABLE EZPlus
(
    EZPlusID            INT             NOT NULL IDENTITY(1111,1),
    EZPlusRewardsCode   CHAR(8)         NOT NULL UNIQUE,
    EZPlusEarnedPoints  INT             NULL,

    PRIMARY KEY (EZPlusID)
);

--CREATE RETAIL CUSTOMER TABLE
CREATE TABLE RetailCustomer
(
    CustomerID  INT     NOT NULL,
    DiscountID  INT     NULL,
    EZPlusID    INT     NULL,

    PRIMARY KEY (CustomerID),

    CONSTRAINT FK_RC_CustomerID
    FOREIGN KEY (CustomerID)
    REFERENCES Customer(CustomerID),

    CONSTRAINT FK_RC_DiscountID
    FOREIGN KEY (DiscountID)
    REFERENCES Discount(DiscountID),
```

```
    CONSTRAINT FK_RC_EZPlusID
    FOREIGN KEY (EZPlusID)
    REFERENCES EZPlus(EZPlusID)
);

--CREATE COMPANY TABLE
CREATE TABLE Company
(
    CompanyID               INT             NOT NULL CHECK(CompanyID BETWEEN
                                            1 AND 9999999),
    CompanyName             NVARCHAR(50)    NOT NULL UNIQUE,
    AddressLine1            NVARCHAR(50)    NOT NULL,
    AddressLine2            NVARCHAR(50)    NOT NULL,
    City                    NVARCHAR(60)    NOT NULL,
    StateCode               CHAR(2)         NOT NULL,
    ZipCode                 VARCHAR(12)     NOT NULL,
    Country                 VARCHAR(60)     NOT NULL,
    ContactName             NVARCHAR(100)   NOT NULL,
    ContactPhone            VARCHAR(20)     NOT NULL,
    ContactEmail            VARCHAR(75)     NOT NULL UNIQUE,
    CompanyDailyRentalRate  DECIMAL(6,2)    NOT NULL,

    PRIMARY KEY (CompanyID)
);

--CREATE CORPORATE CUSTOMER TABLE
CREATE TABLE CorporateCustomer
(
    CustomerID      INT     NOT NULL,
    CompanyID       INT     NOT NULL,

    PRIMARY KEY (CustomerID),

    CONSTRAINT  FK_CC_CustomerID
    FOREIGN KEY (CustomerID)
    REFERENCES Customer(CustomerID),

    CONSTRAINT  FK_CC_CompanyID
    FOREIGN KEY (CompanyID)
    REFERENCES Company(CompanyID)
);

--CREATE VEHICLE STATUS TABLE
CREATE TABLE VehicleStatus
(
    VehicleStatusID     TINYINT         NOT NULL CHECK(VehicleStatusID
                                        BETWEEN 1 AND 9),
    VehicleStatusDesc   VARCHAR(50)     NOT NULL,

    PRIMARY KEY (VehicleStatusID)
);
```

```
--CREATE VEHICLE RENTAL CATEGORY
CREATE TABLE VehicleRentalCategory
(
    VehicleRentalCategoryID     TINYINT         NOT NULL CHECK(VehicleRental
                                                CategoryID BETWEEN 1 AND 25),
    CategoryName                VARCHAR(35)     NOT NULL UNIQUE,
    CategoryDailyRentalRate     DECIMAL(5,2)    NOT NULL,

    PRIMARY KEY (VehicleRentalCategoryID)
);

--CREATE RENTAL AGENCY TABLE
CREATE TABLE RentalAgency
(
    RentalAgencyID      INT             NOT NULL IDENTITY(1,1),
    RentalAgencyName    NVARCHAR(100)   NOT NULL,
    AddressLine1        NVARCHAR(50)    NOT NULL,
    AddressLine2        NVARCHAR(50)    NOT NULL,
    City                NVARCHAR(60)    NOT NULL,
    StateCode           CHAR(2)         NOT NULL,
    Country             VARCHAR(60)     NOT NULL,
    ZipCode             VARCHAR(12)     NOT NULL,
    Phone               VARCHAR(20)     NOT NULL,
    Email               VARCHAR(75)     NOT NULL UNIQUE,

    PRIMARY KEY (RentalAgencyID)
);

--CREATE RENTAL INSURANCE OPTION TABLE
CREATE TABLE RentalInsuranceOption
(
    InsuranceOptionID               TINYINT         NOT NULL CHECK(Insurance
                                                    OptionID BETWEEN 1 AND 8),
    InsuranceOptionDesc             VARCHAR(MAX)    NOT NULL,
    InsuranceOptionAdditionalCost   DECIMAL(5,2)    NOT NULL,

    PRIMARY KEY (InsuranceOptionID)
);

--CREATE RENTAL STATUS TABLE
CREATE TABLE RentalStatus
(
    RentalStatusID      TINYINT         NOT NULL CHECK(RentalStatusID BETWEEN
                                        1 AND 10),
    RentalStatusDesc    VARCHAR(40)     NOT NULL,

    PRIMARY KEY (RentalStatusID)
);

--CREATE FUEL OPTION TABLE
CREATE TABLE FuelOption
(
    RentalFuelOptionID              TINYINT         NOT NULL CHECK(RentalFuel
                                                    OptionID BETWEEN 1 AND 5),
    FuelOptionDesc                  VARCHAR(100)    NOT NULL,
    RentalFuelOptionAdditionalCost  VARCHAR(100)    NOT NULL,

    PRIMARY KEY (RentalFuelOptionID)
);
```

```
--CREATE RESERVATION STATUS TABLE
CREATE TABLE ReservationStatus
(
    ReservationStatusID     TINYINT          NOT NULL CHECK(ReservationStatus
                                             ID BETWEEN 1 AND 9),
    ReservationStatusDesc   VARCHAR(30)      NOT NULL,

    PRIMARY KEY (ReservationStatusID)
);

--CREATE VEHICLE TABLE
CREATE TABLE Vehicle
(
    VehicleID                       INT              NOT NULL
                                                     IDENTITY(1111,1),
    VINNumber                       CHAR(17)         NOT NULL UNIQUE,
    Make                            VARCHAR(40)      NOT NULL,
    Model                           VARCHAR(40)      NOT NULL,
    Year                            SMALLINT         NOT NULL,
    Color                           VARCHAR(20)      NOT NULL,
    LicensePlateNumber              VARCHAR(15)      NOT NULL,
    LicensePlateState               CHAR(2)          NOT NULL,
    Mileage                         INT              NOT NULL CHECK (Mileage
                                                     BETWEEN 1 AND 300000),
    TransmissionType                VARCHAR(35)      NOT NULL,
    SeatCapacity                    TINYINT          NOT NULL,
    VehicleRentalCategoryID         TINYINT          NOT NULL CHECK(Vehicle
                                                     RentalCategoryID
                                                     BETWEEN 1 AND 25),
    VehicleStatusID                 TINYINT          NOT NULL CHECK(Vehicle
                                                     StatusID BETWEEN
                                                     1 AND 9),
    VehicleOwningAgencyID           INT              NOT NULL,
    VehicleCurrentLocationAgencyID  INT              NOT NULL,
    VehicleType                     CHAR(1)          NOT NULL,

    PRIMARY KEY (VehicleID),

    CONSTRAINT FK_V_VehicleRentalCategoryID
    FOREIGN KEY (VehicleRentalCategoryID)
    REFERENCES VehicleRentalCategory(VehicleRentalCategoryID),

    CONSTRAINT FK_V_VehicleStatusID
    FOREIGN KEY (VehicleStatusID)
    REFERENCES VehicleStatus(VehicleStatusID),

    CONSTRAINT FK_V_VehicleOwningAgencyID
    FOREIGN KEY (VehicleOwningAgencyID)
    REFERENCES RentalAgency(RentalAgencyID),

    CONSTRAINT FK_V_VehicleCurrentLocationAgencyID
    FOREIGN KEY (VehicleCurrentLocationAgencyID)
    REFERENCES RentalAgency(RentalAgencyID)
);
```

```sql
--CREATE CAR TABLE
CREATE TABLE Car
(
    VehicleID       INT             NOT NULL,
    TrunkCapacity   Decimal(4,1)    NOT NULL,

    PRIMARY KEY (VehicleID),



    CONSTRAINT FK_C_VehicleID
    FOREIGN KEY (VehicleID)
    REFERENCES Vehicle(VehicleID)
);

--CREATE SUV TABLE
CREATE TABLE Suv
(
    VehicleID       INT         NOT NULL,
    TowingCapacity  SMALLINT    NOT NULL,
    IsAWD           BIT         NOT NULL,

    PRIMARY KEY (VehicleID),

    CONSTRAINT FK_S_VehicleID
    FOREIGN KEY (VehicleID)
    REFERENCES Vehicle(VehicleID)
);

--CREATE MINIVAN TABLE
CREATE TABLE Minivan
(
    VehicleID           INT     NOT NULL,
    HasDiabilityOption  BIT     NOT NULL,

    PRIMARY KEY (VehicleID),

    CONSTRAINT FK_M_VehicleID
    FOREIGN KEY (VehicleID)
    REFERENCES Vehicle(VehicleID)
);

--CREATE CARGOVAN TABLE
CREATE TABLE Cargovan
(
    VehicleID       INT             NOT NULL,
    CargoCapacity   DECIMAL(4,1)    NOT NULL,
    MaximumPayload  SMALLINT        NOT NULL,

    PRIMARY KEY (VehicleID),

    CONSTRAINT FK_CV_VehicleID
    FOREIGN KEY (VehicleID)
    REFERENCES Vehicle(VehicleID)
);
```

```
--CREATE RESERVATION TABLE
CREATE TABLE Reservation
(
    ReservationID               INT         NOT NULL IDENTITY(1,1),
    CustomerID                  INT         NOT NULL,
    RentalAgencyID              INT         NOT NULL,
    VehicleRentalCategoryID     TINYINT     NOT NULL CHECK(VehicleRental
                                            CategoryID BETWEEN 1 AND 25),
    ReservationDropOffAgencyID  INT         NOT NULL,
    ReservationPickUpDate       DATE        NOT NULL,
    ReservationPickUpTime       TIME(0)     NOT NULL,
    ReservationDropOffDate      DATE        NOT NULL,
    ReservationDropOffTime      TIME(0)     NOT NULL,
    ReservationStatusID         TINYINT     NOT NULL CHECK(ReservationStatus
                                            ID BETWEEN 1 AND 9),

    PRIMARY KEY (ReservationID),

    CONSTRAINT FK_RES_CustomerID
    FOREIGN KEY (CustomerID)
    REFERENCES Customer(CustomerID),

    CONSTRAINT FK_RES_ReservationAgencyID
    FOREIGN KEY (RentalAgencyID)
    REFERENCES RentalAgency(RentalAgencyID),

    CONSTRAINT FK_RES_VehicleRentalCategoryID
    FOREIGN KEY (VehicleRentalCategoryID)
    REFERENCES VehicleRentalCategory(VehicleRentalCategoryID),

    CONSTRAINT FK_RES_ReservationDropOffAgencyID
    FOREIGN KEY (ReservationDropOffAgencyID)
    REFERENCES RentalAgency(RentalAgencyID),

    CONSTRAINT FK_RES_ReservationStatusID
    FOREIGN KEY (ReservationStatusID)
    REFERENCES ReservationStatus(ReservationStatusID)
);

--CREATE RENTAL TABLE
CREATE TABLE Rental
(
    RentalAgreementID           INT             NOT NULL IDENTITY(1,1),
    CustomerID                  INT             NOT NULL,
    VehicleID                   INT             NOT NULL,
    RentalAgencyID              INT             NOT NULL,
    RentalDropOffAgencyID       INT             NOT NULL,
    ReservationID               INT             NULL,
    RentalPickUpDate            DATE            NOT NULL,
    RentalPickUpTime            TIME(0)         NOT NULL,
    RentalDropOffDate           DATE            NOT NULL,
    RentalDropOffTime           TIME(0)         NOT NULL,
    RentalPickUpOdometerValue   INT             NOT NULL CHECK(RentalPickUp
                                                OdometerValue BETWEEN
                                                0 AND 300000),
    RentalDropOffOdometerValue  INT             NOT NULL CHECK(RentalDropOff
                                                OdometerValue BETWEEN
                                                0 AND 300000),
```

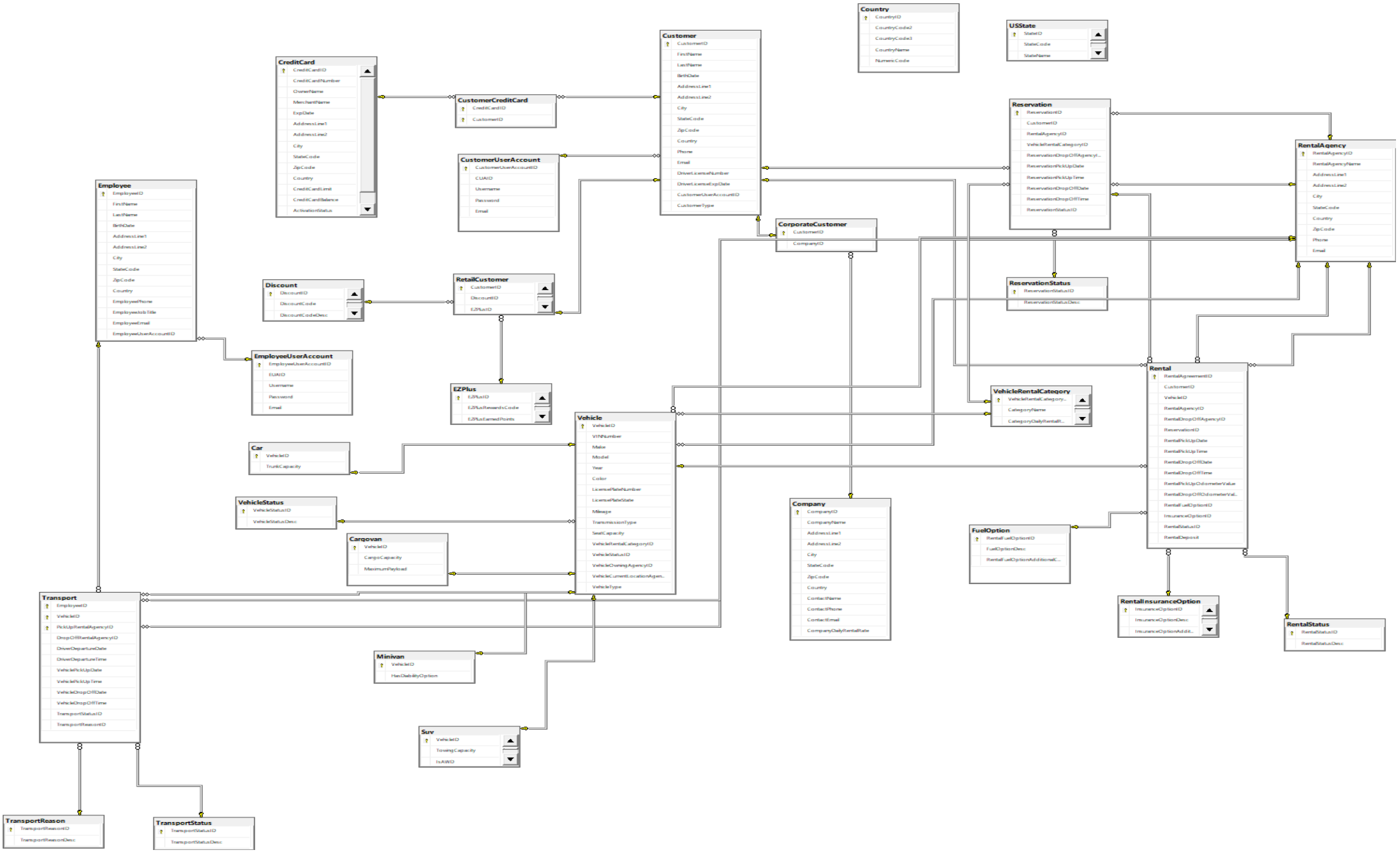
```
    RentalFuelOptionID          TINYINT         NOT NULL CHECK(RentalFuel
                                                OptionID BETWEEN 1 AND 5),
    InsuranceOptionID           TINYINT         NOT NULL CHECK(Insurance
                                                OptionID BETWEEN 1 AND 8),
    RentalStatusID              TINYINT         NOT NULL CHECK(RentalStatusID
                                                BETWEEN 1 AND 10),
    RentalDeposit               DECIMAL(7,2)    NOT NULL,

    PRIMARY KEY (RentalAgreementID),

    CONSTRAINT FK_R_CustomerID
    FOREIGN KEY (CustomerID)
    REFERENCES Customer(CustomerID),

    CONSTRAINT FK_R_VehicleID
    FOREIGN KEY (VehicleID)
    REFERENCES Vehicle(VehicleID),

    CONSTRAINT FK_R_RentalAgencyID
    FOREIGN KEY (RentalAgencyID)
    REFERENCES RentalAgency(RentalAgencyID),

    CONSTRAINT FK_R_RentalDropOffAgencyID
    FOREIGN KEY (RentalDropOffAgencyID)
    REFERENCES RentalAgency(RentalAgencyID),

    CONSTRAINT FK_R_ReservationID
    FOREIGN KEY (ReservationID)
    REFERENCES Reservation(ReservationID),

    CONSTRAINT FK_R_RentalFuelOptionID
    FOREIGN KEY (RentalFuelOptionID)
    REFERENCES FuelOption(RentalFuelOptionID),

    CONSTRAINT FK_R_InsuranceOptionID
    FOREIGN KEY (InsuranceOptionID)
    REFERENCES RentalInsuranceOption(InsuranceOptionID),

    CONSTRAINT FK_R_RentalStatusID
    FOREIGN KEY (RentalStatusID)
    REFERENCES RentalStatus(RentalStatusID)
);

--CREATE TransportStatus TABLE
CREATE TABLE TransportStatus(
    TransportStatusID       TINYINT         NOT NULL CHECK(TransportStatusID
                                            BETWEEN 1 AND 9),
    TransportStatusDesc     VARCHAR(60)     NOT NULL,

    PRIMARY KEY (TransportStatusID)
);
```

```
--CREATE TransportReason TABLE
CREATE TABLE TransportReason(
    TransportReasonID       TINYINT         NOT NULL CHECK(TransportReasonID
                                            BETWEEN 1 AND 9),
    TransportReasonDesc     VARCHAR(60)     NOT NULL

    PRIMARY KEY (TransportReasonID)
);

--CREATE Transport TABLE
CREATE TABLE Transport(
    EmployeeID              INT         NOT NULL,
    VehicleID               INT         NOT NULL,
    PickUpRentalAgencyID    INT         NOT NULL,
    DropOffRentalAgencyID   INT         NOT NULL,
    DriverDepartureDate     DATE        NOT NULL,
    DriverDepartureTime     TIME(0)     NOT NULL,
    VehiclePickUpDate       DATE        NOT NULL,
    VehiclePickUpTime       TIME(0)     NOT NULL,
    VehicleDropOffDate      DATE        NOT NULL,
    VehicleDropOffTime      TIME(0)     NOT NULL,
    TransportStatusID       TINYINT     NOT NULL CHECK(TransportStatusID
                                        BETWEEN 1 AND 9),
    TransportReasonID       TINYINT     NOT NULL CHECK(TransportReasonID
                                        BETWEEN 1 AND 9),

    PRIMARY KEY (EmployeeID, VehicleID, PickUpRentalAgencyID),

    CONSTRAINT FK_T_EmployeeID
    FOREIGN KEY (EmployeeID)
    REFERENCES Employee(EmployeeID),

    CONSTRAINT FK_T_VehicleID
    FOREIGN KEY (VehicleID)
    REFERENCES Vehicle(VehicleID),

    CONSTRAINT FK_T_PickUpRentalAgencyID
    FOREIGN KEY (PickUpRentalAgencyID)
    REFERENCES RentalAgency(RentalAgencyID),

    CONSTRAINT FK_T_DropOffRentalAgencyID
    FOREIGN KEY (DropOffRentalAgencyID)
    REFERENCES RentalAgency(RentalAgencyID),

    CONSTRAINT FK_T_TransportStatusID
    FOREIGN KEY (TransportStatusID)
    REFERENCES TransportStatus(TransportStatusID),

    CONSTRAINT FK_T_TransportReasonID
    FOREIGN KEY (TransportReasonID)
    REFERENCES TransportReason(TransportReasonID)
);
```

```sql
--CREATE COUNTRY TABLE
CREATE TABLE Country
(
    CountryID       TINYINT         NOT NULL,
    CountryCode2    CHAR(2)         NOT NULL UNIQUE,
    CountryCode3    CHAR(3)         NOT NULL UNIQUE,
    CountryName     VARCHAR(60)     NOT NULL,
    NumericCode     SMALLINT        NOT NULL CHECK(NumericCode
                                    BETWEEN 1 AND 999),

    PRIMARY KEY (CountryID)
);

--CREATE USSTATE TABLE
CREATE TABLE USState
(
    StateID     TINYINT         NOT NULL CHECK(StateID BETWEEN 1 AND 51),
    StateCode   CHAR(2)         NOT NULL UNIQUE,
    StateName   VARCHAR(20)     NOT NULL,

    PRIMARY KEY (StateID)
);
```

Development & Implementation Physical Schema Diagram

The diagram below was generated from Microsoft SQL Management Studio after the execution of the DDL script above.



DATABASE DEVELOPMENT & IMPLEMENTATION UNIT TESTING

This section shows the script used to test a select group of entities, using Data Manipulation Language statements (DML) (Insert, Select, Update and Delete). The table group consist of binary relationships between tables, an associative table relationship, and a Supertype/Subtype Parent with multiple children were selected for testing.

INSERT STATEMENTS

CREDITCARD TABLE

Populated the CreditCard table which was the first parent to the Associative Child table – CustomerCreditCard

```
INSERT INTO CreditCard (CreditCardNumber, OwnerName, MerchantName, ExpDate, AddressLine1,  AddressLine2, City,
StateCode, ZipCode, Country, CreditCardBalance, CreditCardLimit, ActivationStatus)
VALUES('$2a$10$TCfAFGJUvr6Ia8JUMCLt9Ob.vnsde5nt1A1LGHuOj7JsXGDz1Q6kq', N'Eric Smith', 'Visa', '2023-07-01',
N'55 Water St.', N'Apt 26', N'New York', 'NY', '10041', 'USA', '200.00', '2000.00', 1);

INSERT INTO CreditCard (CreditCardNumber, OwnerName, MerchantName, ExpDate, AddressLine1, AddressLine2, City,
StateCode, ZipCode, Country, CreditCardBalance, CreditCardLimit, ActivationStatus)
VALUES('$2a$10$v1NOvP4rwR2BsxFkWj4oFutGdwfVJMXIbBR7TSW116AR4p7F.PjEC', N'Sandy Yu', 'Visa', '2024-03-03',
N'37 Main St.', N'Apt 2B', N'Queens', 'NY', '11355', 'USA', '100.00', '1000.00', 1);

INSERT INTO CreditCard (CreditCardNumber, OwnerName, MerchantName, ExpDate, AddressLine1, AddressLine2, City,
StateCode, ZipCode, Country, CreditCardBalance, CreditCardLimit, ActivationStatus)
VALUES('$2a$10$YfLAC4wGFI10aJ0Im/K8We1/GajZjLr58Ugjn0ncyjHCFZA7F/3xu', N'Rafael Núñez', 'Mastercard', '2024-
08-04', N'10 Beverly', N'Apt 1', N'Hollywood', 'CA', '40171', 'USA', '200.00', '2000.00', 1);

INSERT INTO CreditCard (CreditCardNumber, OwnerName, MerchantName, ExpDate, AddressLine1, AddressLine2, City,
StateCode, ZipCode, Country, CreditCardBalance, CreditCardLimit, ActivationStatus)
VALUES('$2a$10$ynPptjzhrEXD2YDxCHTPGOqi5qNxLU/qELAqx6O0GmMnJz/KmwkoS', N'Tim Tucker', 'Visa', '2024-11-27',
N'65 Pine St.', N'Suite 306', N'Brooklyn', 'NY', '11473', 'USA', '100.00', '1000.00', 1);

INSERT INTO CreditCard (CreditCardNumber, OwnerName, MerchantName, ExpDate, AddressLine1, AddressLine2, City,
StateCode, ZipCode, Country, CreditCardBalance, CreditCardLimit, ActivationStatus)
VALUES('$2a$10$o.UAHkjwVw1T/mUpvjEUMOqsx5GAZcV1vR7VM1EmFo7de9SBa5LQ6', N'Sam Williams', 'Mastercard',
'2025-01-14', N'77-18 108 St.', N'Apt 2', N'Queens', 'NY', '11368', 'USA', '200.00', '2000.00', 1);
```

CreditCard table populated with 5 rows after executing Insert statements.

	CreditCardID	CreditCardNumber	OwnerName	MerchantName	ExpDate	AddressLine1	AddressLine2	City	StateCode	ZipCode	Country	CreditCardLimit	CreditCardBalance	ActivationStatus
1	1	$2a$10$TCfAFGJUvr6la8JUMCLt9Ob.vnsde5nt1A1LGHuOj7JsX...	Eric Smith	Visa	2023-07-01	55 Water St.	Apt 26	New York	NY	10041	USA	2000.00	200.00	1
2	2	$2a$10$v1NOvP4rwR2BsxFkWj4oFutGdwfVJMXIbBR7TSW116...	Sandy Yu	Visa	2024-03-03	37 Main St.	Apt 2B	Queens	NY	11355	USA	1000.00	100.00	1
3	3	$2a$10$YfLAC4wGFI10aJ0Im/K8We1/GajZjLr58Ugjn0ncyjHCFZ...	Rafael Núñez	Mastercard	2024-08-04	10 Beverly	Apt 1	Hollywood	CA	40171	USA	2000.00	200.00	1
4	4	$2a$10$ynPptjzhrEXD2YDxCHTPGOqi5qNxLU/qELAqx6O0GmM...	Tim Tucker	Visa	2024-11-27	65 Pine St.	Suite 306	Brooklyn	NY	11473	USA	1000.00	100.00	1
5	5	$2a$10$o.UAHkjwVw1T/mUpvjEUMOqsx5GAZcV1vR7VM1EmF...	Sam Williams	Mastercard	2025-01-14	77-18 108 St.	Apt 2	Queens	NY	11368	USA	2000.00	200.00	1

CUSTOMER TABLE

Populated the Customer table which was the second parent to the Associative Child table – CustomerCreditCard

```
INSERT INTO Customer (FirstName, LastName, BirthDate, AddressLine1, AddressLine2, City, StateCode, ZipCode, Country,
Phone, Email, DriverLicenseNumber, DriverLicenseExpDate, CustomerType)
VALUES(N'Eric', N'Smith', '1983-07-01', N'55 Water St.', N'Apt 26', N'New York', 'NY',
'10041', 'USA', '646-211-1131', 'e.smith@gmail.com', '111111111', '2030-07-01', 'R');

INSERT INTO Customer (FirstName, LastName, BirthDate, AddressLine1, AddressLine2, City, StateCode, ZipCode, Country,
Phone, Email, DriverLicenseNumber, DriverLicenseExpDate, CustomerType)
VALUES(N'Sandy', N'Yu', '1992-03-03', N'37 Main St.', N'Apt 2B', N'Queens', 'NY',
'11355', 'USA', '573-222-3122', 's.yu@gmail.com', '222222222', '2030-03-03', 'R');

INSERT INTO Customer (FirstName, LastName, BirthDate, AddressLine1, AddressLine2, City, StateCode, ZipCode, Country,
Phone, Email, DriverLicenseNumber, DriverLicenseExpDate, CustomerType)
VALUES(N'Rafael', N'Nunez', '1987-08-04', N'10 Beverly', N'Apt 1', N'Hollywood', 'CA',
'40171', 'USA', '983-333-3333', 'r.nunez@gmail', 'C3333333', '2030-08-04', 'R');

INSERT INTO Customer (FirstName, LastName, BirthDate, AddressLine1, AddressLine2, City, StateCode, ZipCode, Country,
Phone, Email, DriverLicenseNumber, DriverLicenseExpDate, CustomerType)
VALUES(N'Tim', N'Tucker', '1980-11-27', N'65 Pine St.', N'Suite 306', N'Brooklyn', 'NY',
'11473', 'USA', '189-444-4444', 't.tucker@gmail.com', '444444444', '2030-11-27', 'R');

INSERT INTO Customer (FirstName, LastName, BirthDate, AddressLine1, AddressLine2, City, StateCode, ZipCode, Country,
Phone, Email, DriverLicenseNumber, DriverLicenseExpDate, CustomerType)
VALUES(N'Sam', N'Williams', '2025-01-14', N'77-18 108 St.', N'Apt 2', N'Queens', 'NY',
'11368', 'USA', '298-555-5555', 's.williams@gmail.com', '555555555', '2030-01-14', 'R');
```

Customer table populated with 5 rows after executing Insert statements. CustomerID was automatically populated by Identity constraint and CustomerUserAccountID is set to NULL.

	CustomerID	FirstName	LastName	BirthDate	AddressLine1	AddressLine2	City	StateCode	ZipCode	Country	Phone	Email	DriverLicenseNumber	DriverLicenseExpDate	CustomerUserAccountID	CustomerType
1	1111	Eric	Smith	1983-07-01	55 Water St.	Apt 26	New York	NY	10041	USA	646-211-1131	e.smith@gmail.com	111111111	2030-07-01	NULL	R
2	1112	Sandy	Yu	1992-03-03	37 Main St.	Apt 2B	Queens	NY	11355	USA	573-222-3122	s.yu@gmail.com	222222222	2030-03-03	NULL	R
3	1113	Rafael	Núñez	1987-08-04	10 Beverly	Apt 1	Hollywood	CA	40171	USA	983-333-3333	r.nunez@gmail	C3333333	2030-08-04	NULL	R
4	1114	Tim	Tucker	1980-11-27	65 Pine St.	Suite 306	Brooklyn	NY	11473	USA	189-444-4444	t.tucker@gmail.com	444444444	2030-11-27	NULL	R
5	1115	Sam	Williams	2025-01-14	77-18 108 St.	Apt 2	Queens	NY	11368	USA	298-555-5555	s.williams@gmail.com	555555555	2030-01-14	NULL	R

CUSTOMERCREDITCARD TABLE

Populated CustomerCreditcard which is the associative child of both CreditCard and Customer parent tables.

```
INSERT INTO CustomerCreditCard(CustomerID, CreditCardID)
VALUES(1111, 1);

INSERT INTO CustomerCreditCard(CustomerID, CreditCardID)
VALUES(1112, 2);

INSERT INTO CustomerCreditCard(CustomerID, CreditCardID)
VALUES(1113, 3);

INSERT INTO CustomerCreditCard(CustomerID, CreditCardID)
VALUES(1114, 4);

INSERT INTO CustomerCreditCard(CustomerID, CreditCardID)
VALUES(1115, 5);
```

CustomerCreditCard table populated with 5 rows after executing Insert statements which kept consistent with data from the parent tables, Customer and CreditCard.

	CreditCardID	CustomerID
1	1	1111
2	2	1112
3	3	1113
4	4	1114
5	5	1115

RETAILCUSTOMER TABLE

Populated the Supertype/Subtype parent and child tables. The Supertype parent is the Customer table and the Subtype child tables are the Retail Customer and the Corporate Customer. The Customer table is populated first and then RetailCustomer child table.

```
INSERT INTO RetailCustomer(CustomerID, DiscountID, EZPlusID)
VALUES(1111, 2, 3);

INSERT INTO RetailCustomer(CustomerID, DiscountID, EZPlusID)
VALUES(1112, 4, 2);

INSERT INTO RetailCustomer(CustomerID, DiscountID, EZPlusID)
VALUES(1113, NULL, 1);

INSERT INTO RetailCustomer(CustomerID, DiscountID, EZPlusID)
VALUES(1114, 4, NULL);

INSERT INTO RetailCustomer(CustomerID, DiscountID, EZPlusID)
VALUES(1115, 1, 4);
```

RetailCustomer table populated with 5 rows after executing Insert statements. Customer table had to be populated with retail customers before executing these insert statements. DiscountID and EZPlusID allow NULL values.

	CustomerID	DiscountID	EZPlusID
1	1111	1112	1113
2	1112	1114	1112
3	1113	NULL	1111
4	1114	1114	NULL
5	1115	1111	1114

COMPANY TABLE

Populated the Company table which has a binary relationship to the CorporateCustomer table.

```
INSERT INTO Company (CompanyID, CompanyName, AddressLine1, AddressLine2, City, StateCode, ZipCode, Country,
ContactName, ContactPhone, ContactEmail, CompanyDailyRentalRate)
VALUES(1111, N'VTech', N'1 Tech Dr.', N'Suite 306', N'Silicon Valley', 'CA', '30331', 'USA', N'Martin Ødegaard', '718-111-
1111', ' o.martin@outlook.com ', 100.00);

INSERT INTO Company (CompanyID, CompanyName, AddressLine1, AddressLine2, City, StateCode, ZipCode, Country,
ContactName, ContactPhone, ContactEmail, CompanyDailyRentalRate)
VALUES(1112, N'Yellow LLC', N'380 Park Ave.', N'Fl 52', N'New York', 'NY', '10101', 'USA', N'David Silva', '718-222-2222',
'd.silva@outlook.com', 100.00);

INSERT INTO Company (CompanyID, CompanyName, AddressLine1, AddressLine2, City, StateCode, ZipCode, Country,
ContactName, ContactPhone, ContactEmail, CompanyDailyRentalRate)
VALUES(1113, N'Together INC', N'420 Santa St', N'Fl 1', N'Santa Fe', 'CA', '34001', 'USA', N'Marco Alonso', '718-333-3333',
'm.alonso@outlook.com', 100.00);

INSERT INTO Company (CompanyID, CompanyName, AddressLine1, AddressLine2, City, StateCode, ZipCode, Country,
ContactName, ContactPhone, ContactEmail, CompanyDailyRentalRate)
VALUES(1114, N'Starfire Inc.', N'541 Flatbush Ave', N'Fl 10', N'Brooklyn', 'NY', '11235', 'USA', N'Iker Casillas', '718-444-
4444', 'i.casillas@outlook.com', 100.00);

INSERT INTO Company (CompanyID, CompanyName, AddressLine1, AddressLine2, City, StateCode, ZipCode, Country,
ContactName, ContactPhone, ContactEmail, CompanyDailyRentalRate)
VALUES(1115, N'Ionized LLC', N'315 W15th St.', N'Fl 4', N'New York', 'NY', '10014', 'USA', N'Luka Modrić', '718-555-5555',
'l.modric@outlook.com', 100.00);
```

Company table populated with 5 rows after executing Insert statements.

	CompanyID	CompanyName	AddressLine1	AddressLine2	City	StateCode	ZipCode	Country	ContactName	ContactPhone	ContactEmail	CompanyDailyRentalRate
1	1111	VTech	1 Tech Dr.	Suite 306	Silicon Valley	CA	30331	USA	Martin Ødegaard	718-111-1111	o.martin@outlook.com	100.00
2	1112	Yellow LLC	380 Park Ave.	Fl 52	New York	NY	10101	USA	David Silva	718-222-2222	d.silva@outlook.com	100.00
3	1113	Together INC	420 Santa St	Fl 1	Santa Fe	CA	34001	USA	Marco Alonso	718-333-3333	m.alonso@outlook.com	100.00
4	1114	Starfire Inc.	541 Flatbush Ave	Fl 10	Brooklyn	NY	11235	USA	Iker Casillas	718-444-4444	i.casillas@outlook.com	100.00
5	1115	Ionized LLC	315 W15th St.	Fl 4	New York	NY	10014	USA	Luka Modrić	718-555-5555	l.modric@outlook.com	100.00

CORPORATECUSTOMER TABLE

Populated the Supertype/Subtype parent and child tables. The Supertype parent is the Customer table and the Subtype child tables are the Retail Customer and the Corporate Customer. The Customer table is populated first and then CorporateCustomer child table (note that extra Corporate Customers were added in the script below).

```
INSERT INTO Customer (FirstName, LastName, BirthDate, AddressLine1, AddressLine2, City, StateCode, ZipCode, Country,
Phone, Email, DriverLicenseNumber, DriverLicenseExpDate, CustomerType)
VALUES(N'Alexander', N'Sørloth', '1979-09-29', N'806 Park Ave.', N'Apt 8C', N'New York', 'NY',
'10041', 'USA', '781-981-6666', 's.alexander@gmail.com', '666666666', '2030-09-29','C');

INSERT INTO CorporateCustomer(CustomerID,CompanyID)
VALUES(1116, 1111);

INSERT INTO Customer (FirstName, LastName, BirthDate, AddressLine1, AddressLine2, City, StateCode, ZipCode, Country,
Phone, Email, DriverLicenseNumber, DriverLicenseExpDate, CustomerType)
VALUES(N'Samuel', N'Anderson', '1993-06-07', N'307 Jensen Ave.', N'Apt 1', N'Hoboken', 'NJ',
'07197', 'USA', '712-981-7777', 's.anderson@gmail.com', 'J77777777777777', '2030-04-17','C');

INSERT INTO CorporateCustomer(CustomerID,CompanyID)
VALUES(1117, 1112);

INSERT INTO Customer (FirstName, LastName, BirthDate, AddressLine1, AddressLine2, City, StateCode, ZipCode, Country,
Phone, Email, DriverLicenseNumber, DriverLicenseExpDate, CustomerType)
VALUES(N'Ben', N'Thomas', '1987-04-17', N'99 Court St.', N'Apt 3', N'Bronx', 'NY',
'11777', 'USA', '981-941-8913', 'b.thomas@gmail.com', '888888888', '2030-04-17','C');

INSERT INTO CorporateCustomer(CustomerID,CompanyID)
VALUES(1118, 1113);

INSERT INTO Customer (FirstName, LastName, BirthDate, AddressLine1, AddressLine2, City, StateCode, ZipCode, Country,
Phone, Email, DriverLicenseNumber, DriverLicenseExpDate, CustomerType)
VALUES(N'Jack', N'Brown', '1991-03-19', N'10 Burn Rd.', N'Apt 1', N'North Bergen', 'NJ',
'21714', 'USA', '472-981-1298', 'j.browngmail.com', 'J99999999999999', '2030-03-19','C');

INSERT INTO CorporateCustomer(CustomerID,CompanyID)
VALUES(1119, 1114);

INSERT INTO Customer (FirstName, LastName, BirthDate, AddressLine1, AddressLine2, City, StateCode, ZipCode, Country,
Phone, Email, DriverLicenseNumber, DriverLicenseExpDate, CustomerType)
VALUES(N'Robert', N'Sánchez', '1990-02-01', N'109-10 Prent Ave.', N'Apt 2', N'Staten Island', 'NY',
'11001', 'USA', '421-214-5621', 'r.sanchez@gmail.com', '123456789', '2030-02-01','C');

INSERT INTO CorporateCustomer(CustomerID,CompanyID)
VALUES(1120, 1115);
```

CorporateCustomer table populated with 5 rows after executing Insert statements.

	CustomerID	CompanyID
1	1116	1111
2	1117	1112
3	1118	1113
4	1119	1114
5	1120	1115

DISCOUNT TABLE

Populated the Discount table which has a binary relationship to the RetailCustomer table.

```
INSERT INTO Discount(DiscountCode, DiscountCodeDesc)
VALUES('AAA99700', 'AAA Membership Discount - 25% off base rate plus 10% donated for breast cancer research.');

INSERT INTO Discount(DiscountCode, DiscountCodeDesc)
VALUES('GOV87569', 'Government Employee Discount - 30% off base rate');

INSERT INTO Discount(DiscountCode, DiscountCodeDesc)
VALUES('STA34156', 'State Employee Discount for 25% off base rate');

INSERT INTO Discount(DiscountCode, DiscountCodeDesc)
VALUES('VET20551', 'Veteran Discount 35% off base rate Plus 10% donation to veteran's family fund.');

INSERT INTO Discount(DiscountCode, DiscountCodeDesc)
VALUES('CNY98765', 'Cuny Student Discount for 10% off base rate');
```

Discount table populated with 5 rows after executing Insert statements.

	DiscountID	DiscountCode	DiscountCodeDesc
1	1111	AAA99700	AAA Membership Discount - 25% off base rate plus...
2	1112	GOV87569	Government Employee Discount - 30% off base rate
3	1113	STA34156	State Employee Discount for 25% off base rate
4	1114	VET20551	Veteran Discount 35% off base rate plus 10% dona...
5	1115	CNY98765	Cuny Student Discount for 10% off base rate

EZPLUS TABLE

Populated the EZPlus table which has a binary relationship to the RetailCustomer table.

```
INSERT INTO EZPlus(EZPlusRewardsCode,EZPlusEarnedPoints)
VALUES('EZP90098', 10000);

INSERT INTO EZPlus(EZPlusRewardsCode,EZPlusEarnedPoints)
VALUES('EZP10001', 500);

INSERT INTO EZPlus(EZPlusRewardsCode,EZPlusEarnedPoints)
VALUES('EZP64932', 159000);

INSERT INTO EZPlus(EZPlusRewardsCode,EZPlusEarnedPoints)
VALUES('EZP20051', 23000);
```

EZPlus table populated with 4 rows with the existing Rewards Program, after executing Insert statements.

	EZPlusID	EZPlusRewardsCode	EZPlusEarnedPoints
1	1111	EZP90098	10000
2	1112	EZP10001	500
3	1113	EZP64932	159000
4	1114	EZP20051	23000

CUSTOMERUSERACCOUNT TABLE

Populated the CustomerUserAccount table which has a binary relationship to the Customer table.

```
INSERT INTO CustomerUserAccount(Username, Password, Email)
VALUES('e.smith', '$2a$10$4GN6hLCo4SHGSp8ZWTupcOdBjK631RJPwWJsnh65Zf3//jECfgL3q', 'e.smith@gmail.com');

INSERT INTO CustomerUserAccount(Username, Password, Email)
VALUES('r.nunez', '$2a$10$TfKhfo.dZIwCCZlfmBmu2.lFBdwYK3w76xDEhQDwx0REO.oEKAr2u', 'r.nunez@gmail.com');

INSERT INTO CustomerUserAccount(Username, Password, Email)
VALUES('t.tucker', '$2a$10$uGYdmhOHOywN7hq7T3ptFusN3/151Ril.skpC..enuBsO0PkdKwi2', 't.tucker@gmail.com');

INSERT INTO CustomerUserAccount(Username, Password, Email)
VALUES('s.anderson','$2a$10$shdPNTAv0U9Yo8ejYw4mUeEC7K2/QnhqcvcN0NnOK1qLMT2ztfZEq',
's.anderson@gmail.com');

INSERT INTO CustomerUserAccount(Username, Password, Email)
VALUES('r.sanchez', '$2a$10$onAU6A2wojxVDR33lHS9E.ZtlYa7E7gzq2JquoPziY6sBrjhDq1AW', 'r.sanchez@gmail.com');
```

CustomerUserAccount table populated with 5 rows after executing Insert statements. CustomerUserAccountID was automatically populated due to the Uniqueidentifier constraint.

	CustomerUserAccountID	CUAID	Username	Password	Email
1	1	4427CC33-1663-449F-8CDC-32BF6F1D72A1	e.smith	$2a$10$4GN6hLCo4SHGSp8ZWTupcOdBjK631RJPwWJsnh65Z...	e.smith@gmail.com
2	2	E753F5BE-EB43-44A7-BCAD-C3E60699FC2C	r.nunez	$2a$10$TfKhfo.dZlwCCZlfmBmu2.IFBdwYK3w76xDEhQDwx0RE...	r.nunez@gmail.com
3	3	901FF43A-E520-4F93-AA1A-7C9E58532670	t.tucker	$2a$10$uGYdmhOHOywN7hq7T3ptFusN3/151Ril.skpC..enuBsO...	t.tucker@gmail.com
4	4	4EE6D363-E68C-4E6E-ADA0-2E5DE4D30811	s.anderson	$2a$10$shdPNTAv0U9Yo8ejYw4mUeEC7K2/QnhqcvcN0NnOK1...	s.anderson@gmail.com
5	5	EBD749AE-F178-4AEF-8DB1-D368E8B835E6	r.sanchez	$2a$10$onAU6A2wojxVDR33lHS9E.ZtlYa7E7gzq2JquoPziY6sBrj...	r.sanchez@gmail.com

SELECT STATEMENTS

SELECT STATEMENT 1

Select Statement on Customer table from the table group, to return only one record with all columns which is based on the Primary Key - CustomerID.

```
SELECT * FROM Customer
WHERE CustomerID = 1115;
```

The query returned the record with all its columns from the Customer table which has an ID number of 1115.

	CustomerID	FirstName	LastName	BirthDate	AddressLine1	AddressLine2	City	StateCode	ZipCode	Country	Phone	Email	DriverLicenseNumber	DriverLicenseExpDate	CustomerUserAccountID	CustomerType
1	1115	Sam	Williams	2025-01-14	77-18 108 St.	Apt 2	Queens	NY	11368	USA	298-555-5555	s.williams@gmail.com	555555555	2030-01-14	NULL	R

SELECT STATEMENT 2

Select Statement on Company table which returned multiple records and includes all columns based on criteria other than the primary key.

```
SELECT * FROM Company
WHERE StateCode = 'CA' AND Country = 'USA';
```

The query returned the records which have, “CA” in StateCode column and “USA” in the country column, with all the columns.

	CompanyID	CompanyName	AddressLine1	AddressLine2	City	StateCode	ZipCode	Country	ContactName	ContactPhone	ContactEmail	CompanyDailyRentalRate
1	1111	VTech	1 Tech Dr.	Suite 306	Silicon Valley	CA	30331	USA	Martin Ødegaard	718-111-1111	o.martin@outlook.com	100.00
2	1113	Together INC	420 Santa St	Fl 1	Santa Fe	CA	34001	USA	Marco Alonso	718-333-3333	m.alonso@outlook.com	100.00

SELECT STATEMENT 3

Select statement on Customer, CreditCard, and CustomerCreditCard tables which have an associative entity relationship and returned one or multiple records with columns from each table.

Business Scenario: What credit card merchant and exp date does the customer Tim Tucker have and its balance? Include first name, last name, merchant name, exp date, birth date, email, and balance.

```
SELECT C.FirstName, C.LastName, C.BirthDate, C.Email, CC.MerchantName, CC.CreditCardBalance, CC.ExpDate
FROM Customer AS C
INNER JOIN CustomerCreditCard AS CCC
ON C.CustomerID = CCC.CustomerID
INNER JOIN CreditCard AS CC
ON CCC.CreditCardID = CC.CreditCardID
WHERE OwnerName = 'Tim Tucker';
```

The query returned the specified columns from each table which are joined Customer to CustomerCreditCard and CustomerCreditCard to CreditCard, where the OwnerName is “Tim Tucker”. Aliases were added to simplify statement.

	FirstName	LastName	BirthDate	Email	MerchantName	CreditCardBalance	ExpDate
1	Tim	Tucker	1980-11-27	t.tucker@gmail.com	Visa	100.00	2024-11-27

UPDATE STATEMENTS

UPDATE STATEMENT 1

A select statement was executed to display the content on the record before updating.

```
SELECT * FROM CreditCard WHERE CreditCardID = 5
```

100 %

Results | Messages

	CreditCardID	CreditCardNumber	OwnerName	MerchantName	ExpDate	AddressLine1	AddressLine2	City	StateCode	ZipCode	Country	CreditCardLimit	CreditCardBalance	ActivationStatus
1	5	$2a$10$o.UAHkjwVw1T/mUpvjEUMOqsx5GAZcV1vR...	Sam Williams	Mastercard	2025-01-14	77-18 108 St.	Apt 2	Queens	NY	11368	USA	2000.00	200.00	1

The following UPDATE statement was used to update the content of the record.

```
UPDATE  CreditCard
SET                 OwnerName = N'Sam Williams',
                    MerchantName= 'Visa',
                    ExpDate = '01/14/2030',
                    AddressLine1 = N'106-10 34thave',
                    AddressLine2 = N'Fl 3',
                    City = 'Corona',
                    StateCode = 'NY',
                    ZipCode= '21438',
                    Country = 'USA',
                    CreditCardBalance = 400.00,
                    CreditCardLimit = 3000.00,
                    ActivationStatus = 0
WHERE CreditCardNumber = 5
```

Executed the Select statement again to verify that the columns were updated.

```
SELECT * FROM CreditCard WHERE CreditCardID = 5
```

100 %

Results | Messages

	CreditCardID	CreditCardNumber	OwnerName	MerchantName	ExpDate	AddressLine1	AddressLine2	City	StateCode	ZipCode	Country	CreditCardLimit	CreditCardBalance	ActivationStatus
1	5	$2a$10$o.UAHkjwVw1T/mUpvjEUMOqs...	Sam Williams	Visa	2030-01-14	106-10 34th Ave	Fl 3	Corona	NY	21438	USA	3000.00	400.00	0

UPDATE STATEMENT 2

A SELECT statement was executed to display record before updating.

```
SELECT * FROM CorporateCustomer WHERE CustomerID = 1116
```

100 %

Results | Messages

	CustomerID	CompanyID
1	1116	1111

The following UPDATE statement was used to update CorporateCustomer.

```
UPDATE  CorporateCustomer
SET     CompanyID = 1112
WHERE   CustomerID = 1116;
```

Executed the select statement again to verify the column had updated. The company ID now becomes the same as another record in the table but since it's a composite key, the combination of the Customer ID and Company ID, make it unique.

```
SELECT * FROM CorporateCustomer WHERE CustomerID = 1116
```

100 %

Results | Messages

	CustomerID	CompanyID
1	1116	1112

DELETE STATEMENTS

DELETE STATEMENT 1

A SELECT statement was executed to display content of the Company table before deletion.

	CompanyID	CompanyName	AddressLine1	AddressLine2	City	StateCode	ZipCode	Country	ContactName	ContactPhone	ContactEmail	CompanyDailyRentalRate
1	1111	VTech	1 Tech Dr.	Suite 306	Silicon Valley	CA	30331	USA	Martin Ødegaard	718-111-1111	o.martin@outlook.com	100.00
2	1112	Yellow LLC	380 Park Ave.	Fl 52	New York	NY	10101	USA	David Silva	718-222-2222	d.silva@outlook.com	100.00
3	1113	Together INC	420 Santa St	Fl 1	Santa Fe	CA	34001	USA	Marco Alonso	718-333-3333	m.alonso@outlook.com	100.00
4	1114	Starfire Inc.	541 Flatbush Ave	Fl 10	Brooklyn	NY	11235	USA	Iker Casillas	718-444-4444	i.casillas@outlook.com	100.00
5	1115	Ionized LLC	315 W15th St.	Fl 4	New York	NY	10014	USA	Luka Modrić	718-555-5555	l.modric@outlook.com	100.00

The following delete statement was executed to remove Together INC from the Company table. First the column from child table CorporateCustomer had to be deleted first before this was done.

```
DELETE
FROM Company
WHERE CompanyID = 1113;
```

Executed the Select statement again to verify the record was removed from the table.

```
SELECT * FROM Company;
```

100 %

Results | Messages

	CompanyID	CompanyName	AddressLine1	AddressLine2	City	StateCode	ZipCode	Country	ContactName	ContactPhone	ContactEmail	CompanyDailyRentalRate
1	1111	VTech	1 Tech Dr.	Suite 306	Silicon Valley	CA	30331	USA	Martin Ødegaard	718-111-1111	o.martin@outlook.com	100.00
2	1112	Yellow LLC	380 Park Ave.	Fl 52	New York	NY	10101	USA	David Silva	718-222-2222	d.silva@outlook.com	100.00
3	1114	Starfire Inc.	541 Flatbush Ave	Fl 10	Brooklyn	NY	11235	USA	Iker Casillas	718-444-4444	i.casillas@outlook.com	100.00
4	1115	Ionized LLC	315 W15th St.	Fl 4	New York	NY	10014	USA	Luka Modrić	718-555-5555	l.modric@outlook.com	100.00

DELETE STATEMENT 2

A SELECT statement was executed to display the contents of CustomerCreditCard table.

```
SELECT * FROM CustomerCreditCard;
```

100 %

Results | Messages

	CreditCardID	CustomerID
1	1	1111
2	2	1112
3	3	1113
4	4	1114
5	5	1115

The following delete statement was executed to Remove one record from the CustomerCreditCard table.

```
DELETE
FROM CustomerCreditCard
WHERE CustomerID = 1115;
```

Executed the select statement again to verify the record was removed from the table.

```
SELECT * FROM CustomerCreditCard;
```

100 %

Results | Messages

	CreditCardID	CustomerID
1	1	1111
2	2	1112
3	3	1113
4	4	1114

CONCLUSION

A summary of the necessary steps taken to develop and implement the project and all its features.

- Provided an overview of the problem statement and objectives, along with the roles and responsibilities of those involved in the completion of the project.

- Stated business and technical requirements.

- Gave explanation of proposed application physical architecture including the database management system physical architecture and the project methodology used to achieve the objectives.

- The Database design and implementation to support both the EZRental Point-of-Sales and Back-end Management Systems is shown ready for interaction between the DBMS and the user.

    - Analyzed requirements to derive the database and conceptualized the business requirements using an EER Conceptual Model
    - Designed the Logical Model & Normalized model
    - Designed the Data dictionary derived from the Normalized model
    - Designed the Physical Model Schema Diagram
    - Developed, implemented, and tested the database based on the Physical Model Schema Diagram, using Microsoft SQL Server