EZRENTAL AUTO POS MANAGEMENT SYSTEM DATABSE DESIGN AND IMPLEMENTATION

By: Mohamed Shohatee

Table of Contents

PROJECT 1 – EZRental Auto Rental POS Management System Database Design and Implementation

Executive Summary

EZRental Inc. has hired **NYC-Tech Solutions Inc.** to design and implement an ***Auto Rental Point-of-Sales Management System Application*** that will include business modules:

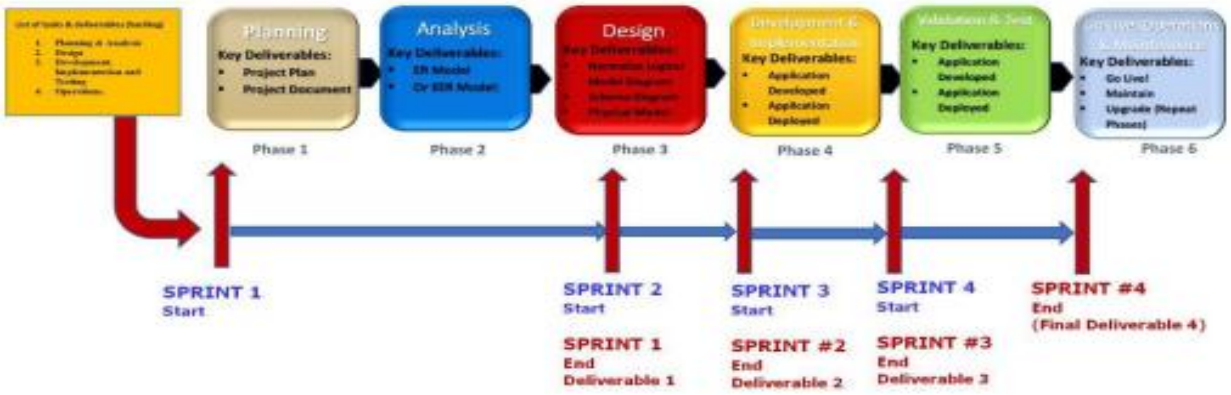
1. **EZRental Point-of-Sales(POS)**
    1. This is intended for ***Customer Service Representatives*** and other ***employees*** in the ***rental agencies***, such as Maintenance Personnel, Vehicle Inventory Team, Transport Drivers etc.
2. A **Corporate INTRANET Website** named **EZRentalCorp.com**
    1. This website is intended for business employees in the corporate offices and Rental Agencies
3. A **e-commerce INTERNET Website** named **EZRental.com**
    1. This website is intended for customers to make and manage reservations via the public internet

Problem Statement & Objectives

EZRental Inc., has hired **NYC-Tech Solutions INC**., to design & implement a suite of **Auto Rental Point-of-Sales Management System Application**. This application has an **EZRental Point-of-Sales(POS) system** that would be used by ***Customer Service Representatives*** and other ***employees*** in ***the rental agencies*** such as Maintenance Personnel, Vehicle Inventory Team, Transport Drivers etc. The application will have **a Corporate INTRANET Website** named **EZRentalCorp.com** intended for business employees in the corporate offices and Rental Agencies. As well as an e-commerce Internet website named **EZRental.com** for customers to make and manage reservations via the internet. The company currently has rental agency branches in US, Canada, United Kingdom, Japan & Australia and looking to expand further globally into Asia, Africa, and the Mediterranean.

The design of this application is to allow customers(retail and corporate) to reserve vehicles for renting in person or online like other car rental systems such as **Avis**, **Hertz**, **Budget**, etc. The application will provide the required functionalities for our Customer Service representatives and other front-line workers in our rental agencies, as well as provide features for business users in our corporate offices who need to create reports, perform analytics and anything related to management of the reservations and rental of our vehicles via our INTERNET PORTAL. These features will allow customers to make and manage vehicle reservations, profile, account etc. via the public internet and is designed to support dozens of major cities in the world providing a great user experience both in the rental agencies and online with the best competitive pricing available in the market.

Project Management Methodology

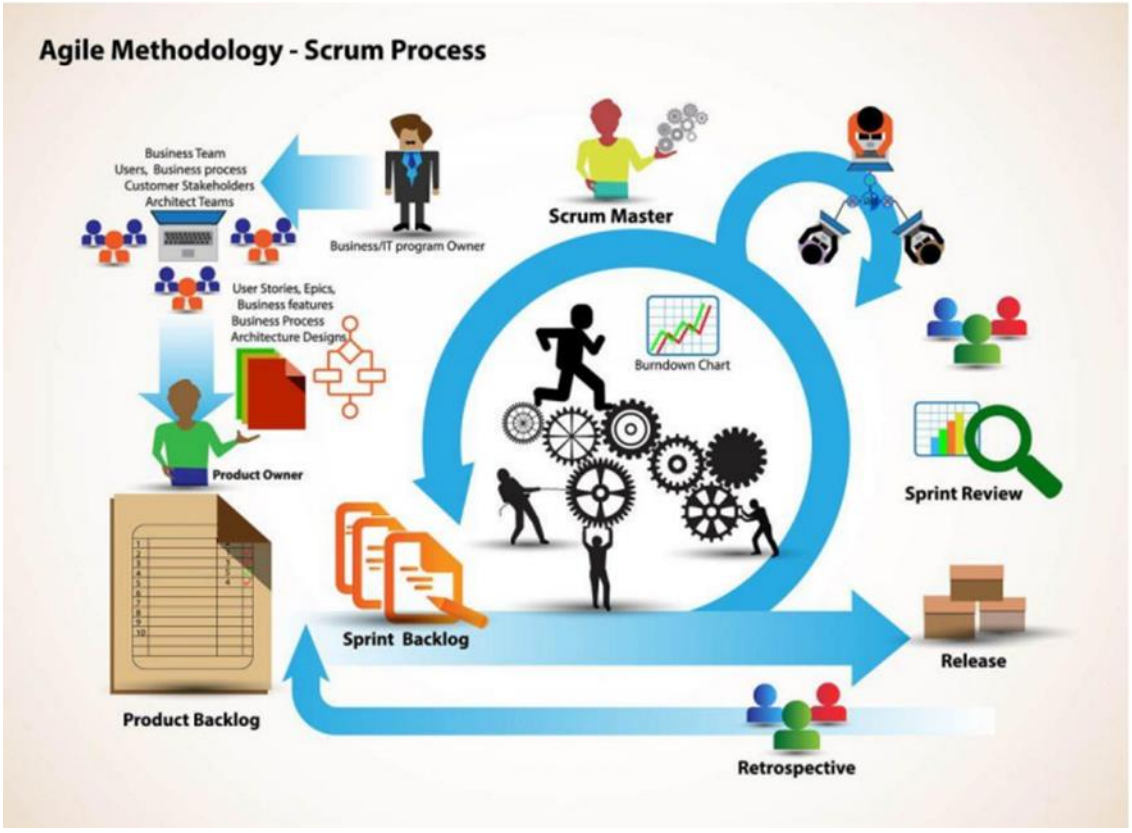


The methodology used is a combination of the Waterfall project management methodology and agile project management methodology.

WATERFALL Project Management Overview	AGILE Project Management Overview
-- A sequential project management design process where a project is divided into several phases -- Each stage is executed in sequence. -- A developer cannot move on to the next phase until the current phase is completed. -- Only when all stages or phases are completed is the project considered done.	-- This methodology consists of an incremental approach to implementing a Project/Application by implementing subsets of FEATURES into pieces. -- The entire Project/Application is DIVIDED and EXECUTED in increments or subset called a SPRINT

Agile Sprint #	Waterfall Phase	Output Deliverable
Sprint #1	Planning	1. Create Project Document 2. Business Requirements
	Analysis	1. ER/EER Conceptual Model Diagram
	Design Phase(Part1)	1. Normalized Logical Model Diagram
Sprint #2	Design Phase(Part 2)	1. Data Dictionary matrix 2. Physical Schema Design Diagram
Sprint #3	Development & Implementation	1. Database application developed & implemented
Sprint #4	Validation & Testing	1. Unit Testing 2. Integration testing
	Operations	1. Operations or Keep running. 2. Keeping the lights on

Picture Illustration of Agile Project Management Methodology



Application Business requirements

- Interviews were held with **EZRental Inc.,** **Project & Business Stakeholders** to ***gather and compile*** a list of **business data needed for the application** also known as *business requirements* which are the **foundation of the database design**.
- The following below contain the ***Business Requirements*** *captured* by the **Business Analyst**:

Business Requirements

About Us:

EZ-Car Rental is an auto rental company that rents vehicles such as cars, SUVs, minivans & cargo vans to customers. In addition, other specialized vehicles such as trucks, motorcycles, boats, mobile homes, etc. We operate in several countries with rental agency locations in the US, Canada, Mexico, UK, Japan & Australia. Within each country we own and operate rental agencies located in cities, regions and state. For example, New York City has 2 rental agencies in Manhattan, one in Brooklyn and two in Queens located at each airport. With multiple rental agencies in cities, states etc., 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). Another very important shortcoming of the current system, is the lack of elasticity since it does not give us the flexibility to scale-up or scale-down resources during 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 delivers a great user-experience, meet our new business requirements, scalable, and elastic to adopt to business trends and seasonal market changes. Elasticity is very important since recently we have been 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 ***rental agency ID*** number, ***agency name***, ***address*** that is composed of the following elements: ***address line1***, ***address line 2 (***which is optional and used for apartment number, suite or any additional address information required), **city**, ***state code*** (which is the two-character code for a state in the US), ***zip code*** & ***country***. In addition, we also need to capture the agency's ***phone number***, and ***email*** which is unique for all agencies as all emails are.

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 to use our services with special corporate rate for their employee's rental services. On the other hand, **Retail Customers** are consumers not associated with a company and engaging in personal rental.

All Customers (Retail & Corporate Customers)

To run our business, the application must store the following customer information for both types of **customer** (retail & corporate) so this data is common to both types of customers:

- 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 (***which is optional and used for apartment number, suite or any additional address information required), ***city***, ***state code*** (which is the two-character code for a state in the US), ***zip code*** & ***country***.
- Customer ***phone number*** & ***email*** (unique like all emails and required to rent).
- In addition, a driver license is required to reserve and rent a vehicle. Therefore, we need to capture the unique ***driver license number (an alpha numeric character string containing numbers & characters)***, ***driver license expiration Date*** and ***driver license state***. In addition, note the following business rule on the business importance of the ***driver license number***:

1. ***The driver license number is used throughout the business to identify a customer for searching, reporting etc.***
2. ***Therefore, the driver license number is the unique ID for a customer to be identified and managed from a business perspective.***

Business Requirements

Our Customers (Cont.):

- A very important attribute we need to capture for every customer is the **credit card**. A credit card includes the following attributes: ***credit card number*** that uniquely identifies the credit card and is a 16-character number digits, ***credit card owner name***, ***credit card issuing company name*** (such as American Express, Visa, MasterCard, Capital One, etc.), ***merchant Code & merchant name*** which is the credit card payment processing company that acts as an intermediary between our business and the customers' credit card companies or bank. The merchant handles the interaction between the purchase of a rental and the credit card company etc., validating credit card transaction. This merchant Code & Name attributes have business meaning and used throughout the business using a digit code for ***merchant Code*** and the name of the merchant associated with the code or ***merchant name***. We currently use the following ***merchant code*** and ***merchant names*** throughout the world to handle our credit card processing:

Merchant Code	*Merchant Name*
1	Stax by Fattmerchant
2	Helcim
3	Dharma Merchant Services
4	Payment Depot
5	National Processing
6	Block
7	Intuit Quickbooks
8	PayPal
9	Stripe
10	Flagship Merchant Services
11	Clover

- Other attributes of credit card are ***expiration date***, ***billing address*** composed of ***address line1***, ***address line 2 (***which is optional and used for apartment number, suite or any additional address information required), ***city***, ***state code*** (which is the two-character code for a state in the US), ***zip code*** & ***country***.
- In addition, ***credit card limit***, ***credit card balance*** & ***activation status*** which is true if the credit card is active and can be used or false when disabled.
- During the interview with business steakholders we captured the following ***Business Rules*** related to a credit card:

1. ***You cannot reserve or rent one of our vehicles without a credit card***
2. ***A customer can have many credit cards they can use to pay for rental transactions.***
3. ***A credit card can be owned by the one customer or co-owned by other individuals such a family member or corporate entity the customer works for. Therefore, many customers can own the same credit card and a credit card can be owned by many customers.***

Business Requirements

Our Customers (Cont.):

Corporate Customers

Corporate Customers are customers who are renting vehicle during business travel and their company have a contract with **EZRental Inc**. These companies get special corporate rate for their employee's rental services. Therefore, for our **corporate customers only**, we must store the following attributes/properties: unique ***company ID*** (we have a unique ID number for each company doing business with us), ***company name***, ***company address*** which contains the elements: ***address line1***, ***address line 2*** (which is optional and used for apartment number, suite or any additional address information required), ***city***, ***state code***, ***zip code*** (which is the two-character code for a state in the US) & ***country***, in addition, ***company contact*** which is composed of ***company representative name***, ***contact phone number*** & ***contact email*** (unique as all email addresses). And finally, we need to store the ***company discount percentage rate*** which is the discounted percentage applied to a corporate customers rental. The company Discount percentage rate is store in the database as a decimal percentage value, for example 20% is stored as 0.20, 30% as 0.30. 50% as 0.50 etc. This discount percentage (0.0x) is applied to the **Vehicle Rental Categories** which determines the price of each category to determine the total discount. Therefore, when a corporate customer rents a vehicle from a vehicle category (such as economic, compact, standard etc.), this discount percentage is applied to each of the categories during the rental/reservation process. Note that every company has a different percentage rating depending on their contract with **EZ-Rentals Inc**. For example, some companies have 20% discount towards their rentals, which would be stored as 0.20 in the database, some have 30% (0.30) etc. Vehicle Rental Categories are discussed in more details later in these requirements.

Retail Customers

Retail Customers can (but don't have to) leverage promotional **discounts** or coupons obtain from other businesses, internet, magazine, organizations, etc., to save money on their rentals. Therefore, data unique to a retail customer that we need to capture for the promotional discount is unique random number ***discount ID*** which uniquely identifies a discount, a unique ***discount code*** or the coupon code itself used to redeem the coupon, which is an alphanumeric code **10-characters** long. This code is generated by our marketing team and published to magazines, newspapers, internet e-commerce sites, etc. Finally, the last attribute is ***discount code description*** or description of the discount. Examples of currently used ***discount ID***, ***discount code***, ***discount code description*** are shown in table below:

Discount ID	*Discount Code*	*Discount Code Description*
1234..	AAA9970054	**AAA Membership Discount - 25% off base rate plus 10% donated for breast cancer research.**
5678..	GOV8756921	**Government Employee Discount - 30% off base rate**
9101..	*STA3415632*	**State Employee Discount for 25% off base rate**
1213..	VET2055179	**Veteran Discount 35% off base rate Plus 10% donation to veteran's family fund.**
Etc..	Etc..	**Etc..**

Retail customers can opt-in to enrolled in the **EZPlus Rewards Program** where they earn points every time they rent and these points can be redeemed for future rentals. Note that the **EZPlus Rewards Program** is **optional** for retail customers & points are earned only when they rent vehicles. For the **EZPlus Rewards Program** we need to store unique random number ***EZPlus ID***, the unique ***Ezplus rewards code*** which is the code used in the business when managing the **EZPlus Rewards Program**. This random code is generated and assigned to a **Retail Customer** by the client application. The number starts with the 3-characters EZP and a 10-digit number e.g., **EZP9999999999**, and the final attribute is the ***EZPlus rewards earned points***, which is an integer that indicates the number of rewards points earned that accumulated after all the rentals and can be used to save on future rentals. **Examples of currently used *EZPlus ID*, *EZPlus rewards Code* and *EZPlus earned points* that we currently use are:**

EZPlus ID	*EZPlus Rewards Code*	*EZPlus Rewards Earned Points*
1234..	EZP9009854637	**10000**
5678..	EZP1000192461	**500**
9101..	EZP6493238865	**159000**
1213..	EZP2005135627	**23000**
	Etc..	**Etc..**

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

3. ***We only have two types of customers retail customer or corporate customers. No other type of customer exists.***

4. ***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 into 4 main types: **CAR**, **SUV**, **MINIVAN**, and **CARGO VAN**. These are the vehicles most rented and available at every rental agency. Nevertheless, there are other categories of vehicles available only certain rental agency locations such as **RECREATIONAL VEHICLES**, **MOTORCYCLES**, **MOBILE HOMES**, etc. No matter what type of vehicle being rented, all vehicle types 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***. Note the following business rule on a ***vehicle VIN number***:

  1. ***The vehicle VIN number is used throughout the business to identify a vehicle for searching, reporting etc.***
  2. ***Therefore, the vehicle VIN number is the unique ID for a vehicle to be identified and managed from a business perspective.***

- Other attributes include the ***vehicle name*** composed of ***make***, ***model*** & **year**. Additional attributes are ***color***, also the ***license plate*** composed of the following components: ***license plate number***, ***license plate state***.
- More attributes are ***mileage***, ***transmission type*** of the vehicle. The Transmission Type attribute has business value thus used in reports and in the business processes. The values used for ***transmission type*** and a ***transmission type description*** as follows:

Transmission Type	*Transmission Type Description*
1	Manual Transmission
2	Automatic Transmission
3	Continuously Variable Transmission (e.g., CVT).
4	Semi-automatic Transmission
5	Dual-clutch Transmission
6	Transaxle Transmission

- ***seat capacity*** attribute, which is the number of seats in the vehicle. Vehicles such as *cars* have a seat capacity of 5 passengers (2 in front and 3 in the back), *SUVs* have 7 or 8 passengers. Cargo Vans have only 2 passenger seat capacity, Minivan have 8 to 9 passengers, special vehicles such as passenger van hold 12 passenger seat capacity, a shuttles bus can hold 16 to 20 passengers, mini-buses 30 to 40 passengers and large busses can hold 70 passengers.

- All vehicles also have a special code and description that 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 represented by the ***Vehicle Status ID***, 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	Available
2	Reserved
3	Rented
4	Not available
5	Maintenance (Not available)
6	Dropped off and located at another agency
7	In Transport to Owning Agency
8	No Longer available for rental

Business Requirements (Cont.)

Our Vehicles (Cont.):

In addition to these attributes shared by all vehicles, there are 4 main categories of vehicle which share unique characteristics than the other types of vehicles found in our agencies. These 4 types 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 trunk capacity and number of luggage they are carrying etc. For example, a *luxury Mercedes E class* car has a trunk capacity of 18.5 cubic ft., which has a large trunk capacity.
- An **SUV** is a vehicle with a ***towing capacity*** attribute in pounds. Towing capacity is a single number in pound or could also be a decimal number in pounds. For example, some of our SUV have a maximum towing capacity of 3,000 pounds etc. Another attribute of SUV is an attribute classification if the SUV is ***All-Wheel-Drive***, which stores a Boolean value of **YES/NO** or **TRUE/FALSE**.
- A **Minivan** has the option of ***having a disability package***, which is also a Boolean value of **YES/NO** 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 cargo vans have typically a maximum payload 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.
- Note that the following Business Rules were identified by the business stakeholders on the vehicles:

1. ***A reservation/rental can only be for one of these four categories of Vehicles or other vehicle types, not a combination.***

2. ***This means, 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.***

Below are additional business rules for our vehicles and agency ownership:

1. ***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.***

2. ***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.***

Reservation Process:

A vehicle must be reserved if a customer wants to guarantee the vehicle will be available for rental. 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. On the other hand, a customer can walk into an agency and rent without reservation but only vehicles that are available at the time and not reserved.

We have the following business rules for reserving a vehicle reservation:

1. ***A reservation is NOT made for a specific vehicle, but to a vehicle rental category. Rental category examples are economy, intermediate, full size, luxury.***

2. ***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*** where the vehicle will be picked up.

Business Requirements (Cont.)

Reservation Process (Cont.):

A **Vehicle 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 rate we currently using in our business:

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-Cargo Van	$19.95
15	Pick Up-Mid Size	$69.95
16	Pick Up-Full Size	$105.99
17	Motorcycle-Touring	$19.95
18	Motorcycle-Cruiser	$199.99
19	Motorcycle-Scooter	$79.95
20	Passenger Van (12 passengers)	$161.00
21	Passenger Shuttle (16 passengers)	$180.00
22	Passenger Shuttle (20 passengers)	$220.00
23	Passenger Mini-Bus (30 passengers)	$250.00
24	Passenger Mini-Bus (40 passengers)	$280.00
25	Passenger Large-Bus (80 passengers)	$300.00

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

1. ***A vehicle is a member of a vehicle rental category.***
2. ***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 business rules apply to a reservation:

1. ***A vehicle can be reserved to be picked up at the INDICATED rental agency and dropped off at the SAME rental agency.***
2. ***A vehicle can be reserved to be picked up at the INDICATED rental agency and dropped off at a DIFFERENT rental agency.***
3. ***A reservation is made only for one pick-up rental agency, but a rental agency can have many reservations for pick-ups taking place.***
4. ***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 rental category for a specific rental agency, we wish to capture the following:

- A unique ***reservation ID*** which is used by the business to manage and track reservations, the ***rental agency ID*** 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*** and ***status description*** we currently use in our business.

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

For a reservation we must adhere to the following business rules:

1. ***A customer can make none, one or many reservations for a vehicle rental category at a rental agency.***
2. ***A rental category can be reserved by none, one or many customers at a rental agency.***
3. ***A rental agency can get many or no reservations for a vehicle rental category by a customer.***
4. ***A reservation can only have one pick-up rental agency location, but a rental agency can have many reservation pick-ups happening.***
5. ***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 business rules apply:

1. ***A customer rents a vehicle Rental Category at a rental agency. This means the rental process requires the customer, vehicle rental category, and & rental agency for a rental to be complete.***
2. ***A Rental includes a specific Vehicle of the vehicle rental category. A vehicle can be rented many times, but a rental is only for one vehicle only. You cannot rent multiple vehicles in one rental contract.***
3. ***During the rental process we may have any of the following business rules/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) ***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.

For the rental process, the following business rules also apply:

1. ***A rental can only be for one pick-up rental agency, but a rental agency can have many rental pick-ups taking place.***
2. ***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*** and ***rental drop off odometer value***.

Business Requirements (Cont.)

The Rental Process (Cont.):

- In addition, a customers receive a vehicle with a full tank of gas and customers are expected to return the car on a full tank of gas otherwise they must pay a penalty upon return. Since we understand our customers are busy and may forget to return the car with a full tank of gas, we offer our customers with the option to pay in advance for a full tank of gas at our rates and don't have to worry about returning the vehicle with a full tank of gas. Therefore, we need to capture the unique ***rental fuel option ID*** or option chosen by the customer, ***rental fuel option description*** and ***rental fuel option additional cost***. We currently use the following fuel option IDs, descriptions, and example of each of the additional cost for the fuel option:

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.	$13.97 (Important, this Decimal value of $13.97 is just an example, since the value is calculated during car return process and is based on the current price for a gallon of gas etc. therefore price will vary.)
2	Pay for full tank in advanced at time of rental, return car empty. No refund for unused gas.	$45.99 (Important, this Decimal value of $45.99 is just an example, since the value is calculated during car return process and is based on the current price for a gallon of gas etc. therefore 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.):

- Other attribute 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** and for any damage or other charges that were incurred during the rental period. This deposit is refunded to the customer's credit card when the vehicle is returned in the condition in which it was rented.
- Finally another attribute we need to capture is the ***rental total cost*** or total cost that needs to be paid by the customer. This value is calculated based on selected *fuel option*, *insurance option*, *vehicle rental category* price and other factor such as such as duration of the rental etc.

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:

1. ***A reservation is made for a rental and the opposite holds true; a rental is based on a reservation.***
2. ***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.***
3. ***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 currently has 5,500 employees across the world. We do expect to grow as we move into new markets such as Asia, Africa, and the Mediterranean. But our business does not require a large workforce, therefore, we don't expect to grow more than 12,000 in the next 10+ years. Our employees consist of customer service agents in the Rental Agencies & online support who interact with our customer to reserve and rent vehicles. In addition, back-office inventory personnel, auto specialists who work in our services centers servicing our vehicles, 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***. In addition, we need to capture the employee ***social security number***. Below are some business rules and usage for the ***EmployeeID*** and the ***social security number***.

1. The employee ***social security number*** needs to be protected and secured as per federal regulations. All security measures such as encryption, etc., need to be taken to protect the ***social security number***; therefore, the full ***social security number*** ***cannot*** be seen by employees, reports, and other business processes.
2. In special cases where the ***social security number*** needs to be displayed, only the last 4 digits will be shown using the following format ******_**_1234**. Nevertheless, the goal is NOT to display the ***social security number*** as much as possible, and it should only be used internally within the application for processing but not displaying.
3. The ***EmployeeID*** number is what is used throughout the business to identify an employee for searching, reporting. business processing, etc.
4. Therefore, the ***EmployeeID*** is the unique ID for an employee to be identified and managed from a business perspective.

Security & Application 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, the ***employee password*** alpha-numeric that is known only to the user, and finally the employee ***email*** to map the user-account to an Employee. Note the following business rule:

1. 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.

Business Requirements (Cont.)

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, the ***customer password*** that is an alpha-numeric known only to the customer, and finally, the customer ***email*** to map the customer user-account to a customer. Note the following business rule:

1. A customer can own one customer user account only, and a customer user account can only be owned by one customer.
2. For a period of time, we will need to register customers into our **EZRental.com** business, nevertheless the web portal may NOT be implemented or completed when new customers are registering at this time, therefore, for period of time, creating a customer user account when registering a new customer is optional until the Web Portal Application is created. But is important in the future, that we force the creation of customer user accounts when a new customer is registered once the Web Portal Application is ready. It is the responsibility of the database architect(s) and full-stack developers to update this feature when the appropriate time comes.

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. Therefore, we need to capture the ***Transport Status ID*** or unique number that identifies a status and the ***Transport Status Description***, or description of each status ID. 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 located at any time and their status.

Business Requirements (Cont.)

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 in the future. 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 are that the design is modular and scalable for future growth.

Application Development & Technical Requirements

- Interviews were held with **EZRental Inc.,** project's Business Decision Makers (BDMs), stakeholders and Information System Technology Decision Makers(TDMs) to compile a list of ***Application Development & Technical Requirements*** to design & develop the application
- The Application Development & Technical Requirements are highlighted in the following 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 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*** & **ASP.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/Print 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/Print, 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.

Application Development & Technical Requirements

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.

Application Development & Technical Requirements

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.

Application Development & Technical Requirements

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.	$13.97 (Important, this Decimal value of $13.97 is just an example, since the value is calculated during car return process and is based on the current price for a gallon of gas etc. therefore price will vary.)
2	Pay for full tank in advanced at time of rental, return car empty. No refund for unused gas.	$45.99 (Important, this Decimal value of $45.99 is just an example, since the value is calculated during car return process and is based on the current price for a gallon of gas etc. therefore 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

- **Transportation Reason Option:**

  - UI screens that require the user to populate the TRANSPORTATION OPTIONS field, must be prepopulated with the list of transportation reason options as shown in the table below:

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 transportation reason 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 transportation reason option data. This is not an immediate requirement out of the gate but should be targeted as part of a future upgrade.

- **Transportation Reason Option:**

  - UI screens that require the user to populate the TRANSPORTATION STATUS field, must be prepopulated with the list of transportation status options as shown in the table below:

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 transportation status 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 transportation status 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.

Customer Facing Self-Service Web-Portal Features and Functionalities Requirements:

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.

Application Development & Technical Requirements

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 Physical Technical Architecture

The following architecture was designed after reviewing the business requirements and technical requirements:

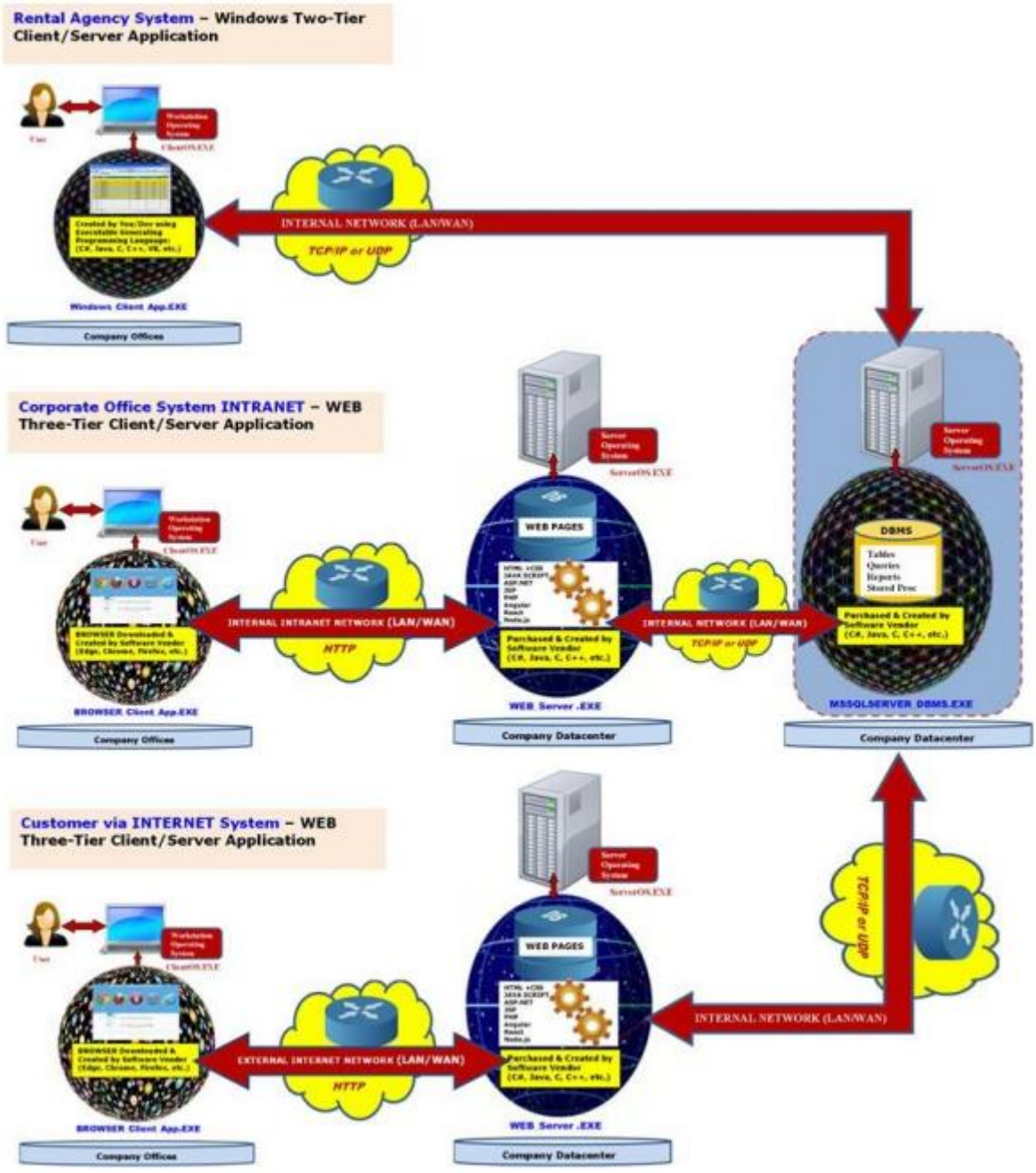
- Rental Agency Employees:
    - The system in our agencies used by the customer service representatives or front-line workers, must be able to quickly respond and execute the necessary requests such as:
        - **POS Customer Management (Retail Customer & Corporate Customer) features** such as Customer Search & Print, New Customer Registration, Customer Update, Customer Deletion, & Customer Listing functionalities
        - **POS Vehicle Reservation, Rental & Return Management Feature** such as Vehicle Reservations, Vehicle Rental & Vehicle Return functionalities
        - **POS Vehicle Inventory Management Feature** allows inventory personnel and employees to bulk-manage vehicles such as Cars, SUVs, Mini-Vans, Cargo Vans, and other vehicles to be searched, added, updated, deleted, printed, listed etc.
        - **POS Credit Card Management Feature** such as Credit Card Search & Print, New Credit Card Registration, Credit Card Update, Credit Card Deletion, & Credit Card Listing functionalities
    - Customer service agents is provided with a rich user-interface experience
    - The system is designed to provide well for other back-end personnel such as vehicle inventory managers and administrators, service personnel, vehicle transport drivers, etc.

- Corporate Offices:
    - The corporate offices are where our business operations are managed by business employees and employees at the rental agencies via the INTRANET Web Portal and the features include:

        - **Intranet Web Enterprise Resource Planning Systems (ERP) Portal Feature** such as providing access to Enterprise Resource Planning Systems (ERP) Applications such as: Customer Credit Card Management System, Vehicle Inventory Management System, Customer Relationship Management (CRM), Human Resource Management System, & Finance & Operations System, Marketing System, Customer & Field Service System etc.

        - **Web EZRental Point-of-Sales Corporate Management Feature** which allows employees to manage & execute Point-of-Sales (POS) transaction via the Intranet Web Poral such as: Search Customer Profile Information, Customer Account Management, Customer Registration, Customer Update, Customer Delete, & Customer Listing functionalities, Manage & Make Reservations of a Vehicle, Manage an existing Rental, etc.

- Customer self-service:
  - The customer has the ability to make reservations, manage reservations and rentals online via the internet and should be able to do so from anywhere in the world via our web portal



Application Development Features and Functionalities (Agile Backlog)

During analysis and meetings, it was decided that the application is to deliver the following **features** & **functionality**. These **features** make up the **AGILE BACKLOG:**

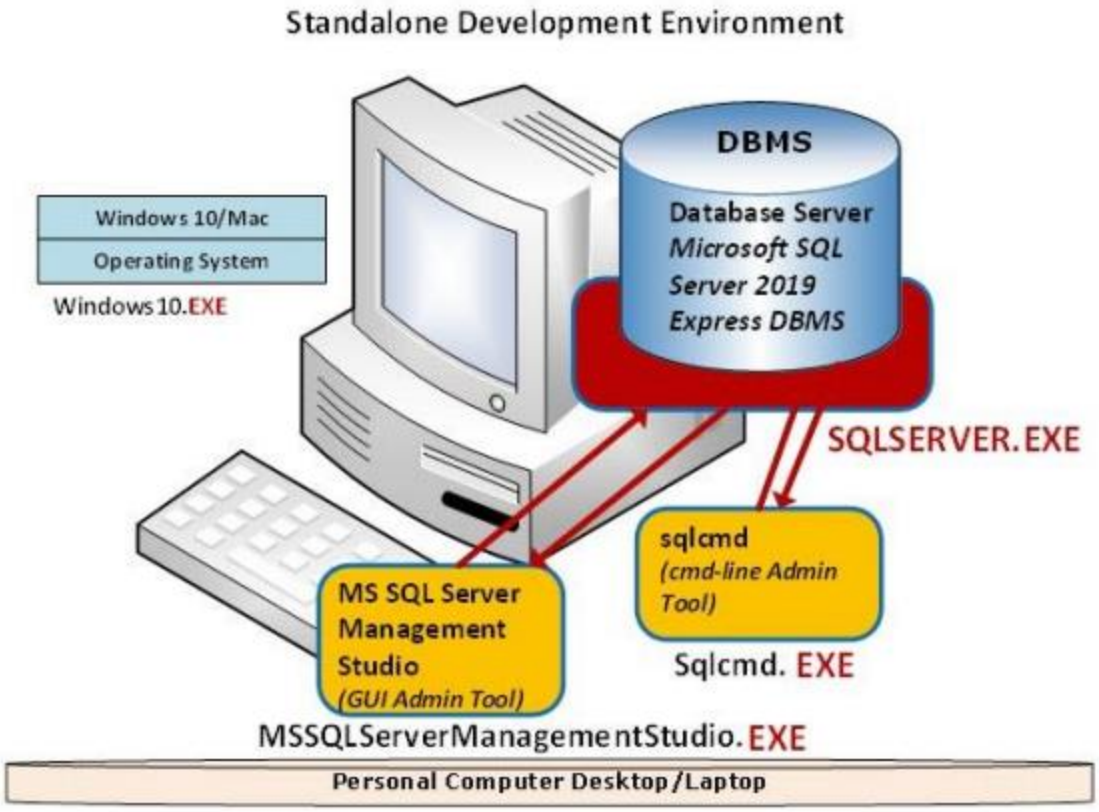
FEATURE #1A	**FEATURE #1A – EZRental Rental Agency Point-of-Sales (POS) BACK-OFFICE CREDIT CARD MANAGEMENT SYSTEM:**
	
	• **WINDOWS CLIENT POINT-OF SALES SYSTEM** – ***WINDOWS UI FORM(S) FRONT-END APPLICATION & OOP PROGRAMMING*** features is **a back-end system** used by **customer service representative & other employees** via the ***Point-of-Sales computer*** machine in ***the Rental Agencies*** to service customer's **CREDIT CARD MANAGEMENT**, or transactions required when servicing customers.
	• The following are features and functionality are required for this application with features such as:
	  ○ **POS Credit Card Management Feature: POS Customer Credit Card Management features** such as **Credit Card Search & Print, New Credit Card Registration, Credit Card Update, Credit Card Deletion, & Credit Card Listing functionalities:**
	  ▪ **Feature UI Form Requirements: Design & programming** of required User-Interface Forms & GUI Controls to support this feature.
	  ▪ **Feature Processing Requirements: Design & programming** of required **Object-Oriented (OOP) Processing & Logic** to support this feature.
	• This feature is designed only to be used by customer service agents and other employees using the **Windows Two-Tiered Client/Server Application** in the **Rental Agencies.**

FEATURE #1B	**FEATURE #1B – EZRental Rental Agency Point-of-Sales (POS) Credit Card Management System Back-end Database Design & Implementation to support this feature:** • **DATABASE SERVER BACK-END SYSTEM – *BACK-END DATABASE DESIGN & FEATURES* (Create the Tables & Relationships, pre-populated tables, stored procedures, views, indexes etc.,) to support this feature.**
FEATURE #2A	**FEATURE #2A** – EZRental Internal Back-Office Credit Card Management System:

	• The EZRental Credit Card Management System is a Back-end 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.
	• **WINDOWS CLIENT-SIDE/FRONT-END APPLICATION** – WINDOWS FORM APPLICATION INTERFACE SCREENS & features required to implement the following features:
	  ○ Credit Card Management System Feature: Manage credit card records by performing the following actions: search & print, add, edit, remove, print & list all credit cards.
	  ▪ Back-Office Inventory Feature Form requirements: Design & programming of required User-Interface Forms & GUI Controls to support the Credit Card Management System feature.

FEATURE #2B	**FEATURE #2B –** EZRental Internal Back-Office Credit Card Management System Back-end Database Design & Implementation to support the feature: • DATABASE SERVER-SIDE/BACK-END – BACK-END DATABASE DESIGN & features (Tables, prepopulated tables, data, stored procedures, views, indices etc.,) to support each of these functionalities.

Database Management System Development Environment & Physical Architecture



MS SQL Server Community Edition is used as the standard DBMS at EZRental Inc. It includes SQL command-line admin tools along with MS SQL Server Management Studio graphical admin tool to create the database development Environment as show above:

Project Roles & Responsibilities

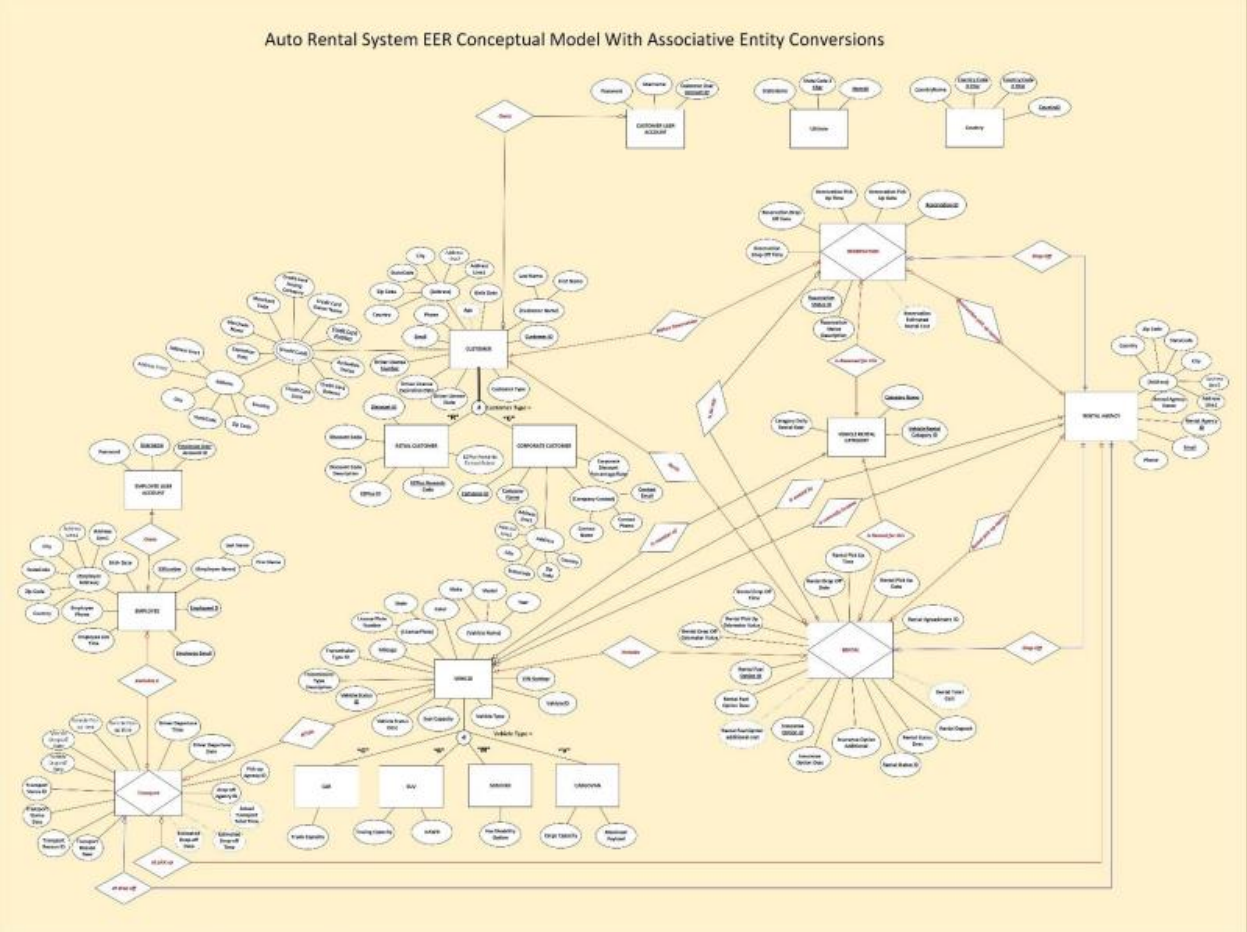
The **Business/Database** Analyst assembled the required database development team and the table below describes each of the roles and the individual(s) that will execute the roles:

Person	Role	Description
Prof. Rodriguez	Program Manager, Agile Scrum Master & Project Manager	Owner of the project and liaison to Manage the EZRental Inc., the customer. ▪ Activities include but not limited to. 1. Owner of project responsible for the success of the project. 2. Project Management 3. Scrum Master ensures the project stays on time and moves in the right direction. Clear any obstacles impeding the team's progress etc.
Consultant #1: Prof. Rodriguez	Business & Database Analyst	A Business/Database Analyst was hired by Prof. Rodriguez to interview the stakeholders at EZRental Inc. And create the Business Requirements that will be the foundation to the database design & implementation. ▪ Activities include but not limited to: 1. Engage in discovery activities & interview the stakeholders at EZRental Inc. 2. From the interview and discovery create 1) ER/EER Conceptual Data Model from the business requirements & 2) Normalized Logical Model.
Consultant #2, 3, 4 & 5 Mohamed Shohatee	Database Developers	This role uses the Normalized Logical Model created by consultant #2 to create the Data Dictionary, Physical Schema Diagram, and Implement the Database Application for the Auto Rental System. ▪ Activities include but not limited to: 1. Use the Normalized Logical Model created by consultant #2 to do the following: 1)Create Data Dictionary tables for each logical table targeting MS SQL Server for CST4708 class and Oracle18c Data Types for CST3604 Class. 2) Create Physical Schema Diagram. 2. From these two deliverables, 1) implement the Database Application using Oracle 18c for the Auto Rental System
Consultant #6 Mohamed Shohatee	Database Administrator	The DB Admin, install the DBMS, maintain, and operate the DBMS throughout its lifetime. ▪ Activities include but not limited to: 1. As DB Admin, you are to 1) Setup & install MS SQL Server for CST4708 class and Oracle18c for CST3604 Class DBMS. 2) Administrative tools for target DBMS. 2. Also, as DB Admin, you are to 3) Operate & Maintain the DBMS.

The **Application Full Stack OOP Architect/Analyst** aligned the required application development team and the table below describes each of the roles and the individual (s) that will execute the roles:

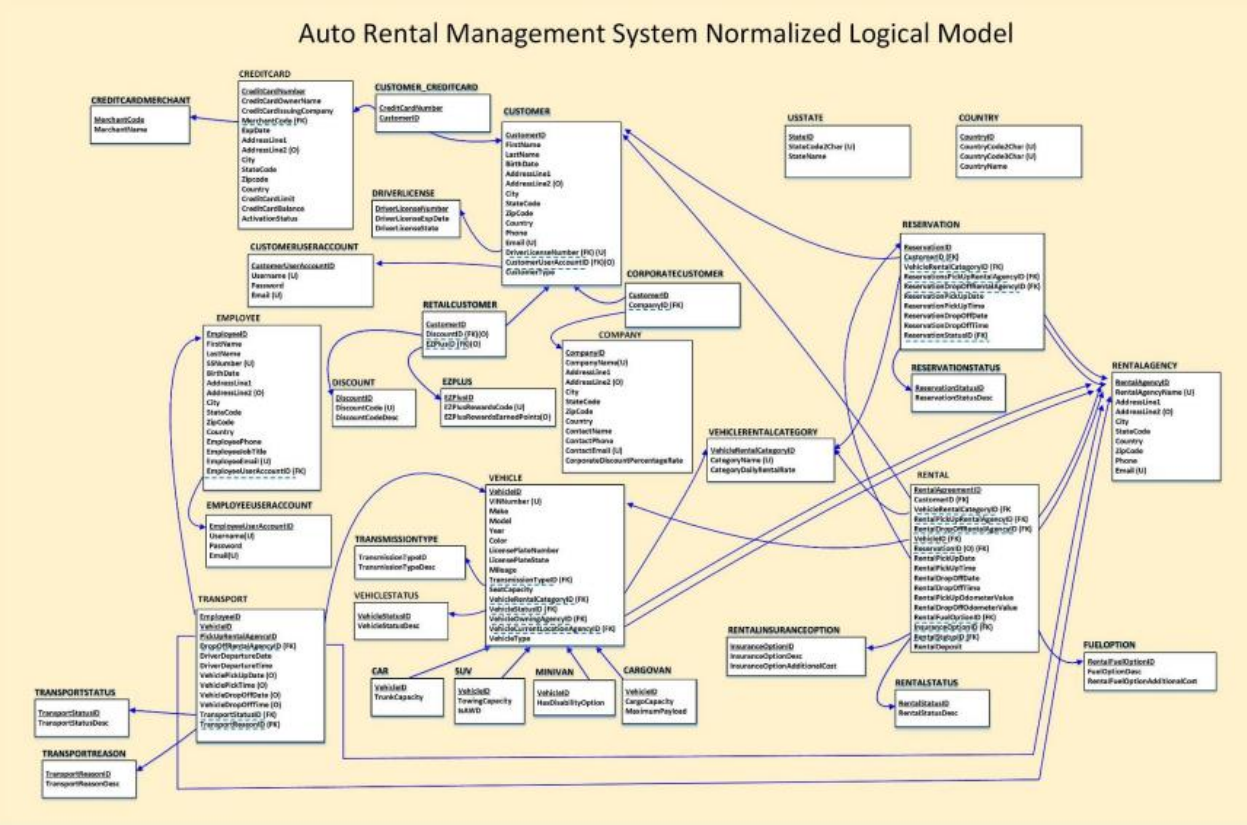
Person	Role	Description
Consultant #7 & 13 Mr. Rodriguez	Full Stack Object Oriented Programming Architect	▪ An Object-Oriented-Programming Architect was hired by Prof. Rodriguez to interview the stakeholders at EZRental Inc. and derive the Application Technical Requirements in addition to designing the Class/Object Model Architecture. This also includes the planning and designing both Windows Client Application and the Web Browser Application. ▪ Activities include but not limited to: 1. Engage in discovery activities & interview the stakeholders at EZRental Inc. 2. From the interview and discovery 1) Design/Architect the Object-Oriented Programming Class/Object Model for the Windows Client Application. 3. Design/Architect the Object-Oriented-Programming Class/Object Model for the Web Browser Application.
Consultants #8, 9, 10, 11 & 12 Mohamed Shohatee	Full Stack Windows Application Developers & UI/UX Client Application Developer	▪ Object-Oriented-Programming developer to implement the Windows Client Application using C# & .NET technologies & on the database side, implement stored procedures and support the databased team as needed. ▪ Activities include but not limited to: 1. As full stack developer, Programming & implementation of the Object Oriented-Programming of Class/Object Model designed by consultant #7 for the Windows Client Application using C# & .NET Technologies. 2. In addition, Development of Database Stored Procedures, and other development requirements in the Back-end DBMS. 3. From the technical requirements, design a high-level Graphical User Interface (GUID) wireframe, & implement the front-end UI Programming, features & functionality
Consultant #14, 15, 16, 17 & 18 Mohamed Shohatee	Full Stack Web Application Developer & UI/UX Web Application Developer	▪ Object-Oriented-Programming developer to implement the Web Browser Application using C# & ASP.NET technologies. ▪ Activities include but not limited to: 1. As full stack developer, Programming & implementation of the Object-Oriented-Programming of Class/Object Model designed by consultant #7 for the Web Browser Client Application using C# & ASP.NET Technologies. 2. From the technical requirements, design a high-level Graphical User Interface (GUID) wireframe, & implement the Webfront-end UI Programming, features & functionality in the Web Server Application

Database Design Deliverable #1 – ER/EER Conceptual Model Diagram

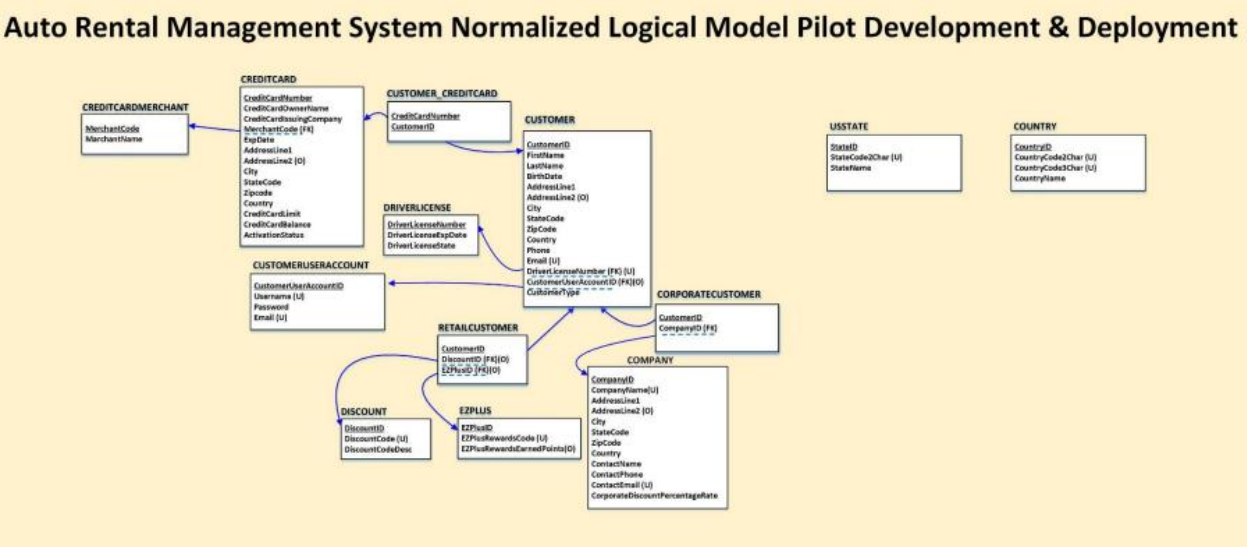


The ER/EER Model is designed to identify the ***entities*** and ***relationships*** in the ***Application Business Requirements*** and is described using the Waterfall Methodology Analysis Phase.

Database Design Deliverable #2 – Normalized Logical Model Diagram



The Normalized Logical Model Diagram is derived from the Logical Model Diagram which converts it into database tables.



The diagram above is a Small PILOT that has been deployed to production for testing for the full implementation after all bugs are fixed and the application is working for the pilot users

Database Design Deliverable #3 – Physical Model Data Dictionary

The tables below represent the Physical Model Data Dictionary which lists the name, data type, constraints and descriptions, the metadata, of the information gathered from and inserted into the database of the EZRENTAL POS Auto Management System:

#1

USSTATE							
Column Num	**Attribute/Column Name**	**Generic Data Type Name**	**MS SQL Server Data Type**	**Is It Required?**	**Length/ Size /Format**	**Constraints**	**Description/ purpose**
1)	**StateID**	Number	TINYINT	Y	1-53	PRIMARY KEY CHECK(StateID between 1 and 53)	State ID. This has business meaning.
2)	StateCode2Char	Character	CHAR(2)	Y	2	UNIQUE NOT NULL	State abbreviation. Specifically U.S,
3)	StateName	String	VARCHAR(50)	Y	50	NOT NULL	State name. Specifically U.S.

#2

CORPORATECUSTOMER							
Column Num	**Attribute/Column Name**	**Generic Data Type Name**	**MS SQL SERVER Data Type**	**Is It Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
1)	**CustomerID**	Number	INT	Y	Default	PRIMARY KEY	CustomerID composite Primary Key
2)	CompanyID	Number	SMALLINT	Y	20000	FOREIGN KEY NOT NULL CHECK(CompanyID between 1 and 20000)	CompanyID composite Foreign Key

#3

CREDITCARD							
Column Num	**Attribute /Column Name**	**Generic Data Type Name**	**MS SQL Server Data Type**	**Is It Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
1)	**CreditCardNumber**	String	VARCHAR(16)	Y	16	**PRIMARY KEY**	Credit card number. This Primary Key has business meaning
2)	CreditCardOwnerName	String	VARCHAR(50)	Y	50	**NOT NULL**	Credit card owner name.
3)	CreditCardIssuingCompany	String	VARCHAR(50)	Y	50	**NOT NULL**	Credit card issuing company.
4)	MerchantCode	Number	TINYINT	Y	1-20	**FOREIGN KEY NOT NULL CHECK(MerchantCode between 1 and 20)**	Merchant Code number. This Foreign Key has business meaning.
5)	ExpDate	Date	DATE	Y	MM/YY	**NOT NULL**	Expiration date of credit card.
6)	AddressLine1	String	VARCHAR(50)	Y	50	**NOT NULL**	House number address and street name #1
7)	AddressLine2	String	VARCHAR(50)	N	50	**NULL**	House number address and street name #2
8)	City	String	VARCHAR(35)	Y	35	**NOT NULL**	Name of City
9)	StateCode	Character	CHAR(2)	Y	2	**NOT NULL**	State Abbreviation. Specifically, U.S.
10)	Zipcode	String	VARCHAR(10)	Y	10	**NOT NULL**	Zip code.
11)	Country	String	VARCHAR(56)	Y	56	**NOT NULL**	Country name.
12)	CreditCardLimit	Number	DECIMAL(8,2)	Y	X = 8 Y = 2	**NOT NULL**	Credit Card spending limit. X being the total amount of digits. Y is number of digits to the right of the decimal.
13)	CreditCardBalance	Number	DECIMAL(8,2)	Y	X = 8 Y = 2	**NOT NULL**	Credit Card balance. X being the total amount of digits. Y is number of digits to the right of the decimal.
14)	ActivationStatus	String	BIT	Y	Default	**NOT NULL**	Credit Card activation status. Active or not?

#4

CREDITCARDMERCHANT							
Column Num	**Attribute/Column Name**	**Generic Data Type Name**	**MS SQL SERVER Data Type**	**Is It Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
1)	**MerchantCode**	Number	TINYINT	Y	1-20	**PRIMARY KEY CHECK(MerchantCode between 1 and 20)**	Merchant Code number. This has business meaning.
2)	MerchantName	String	VARCHAR (40)	Y	40	**NOT NULL**	Name of Merchant.

#5

CUSTOMER_CREDITCARD							
Column Num	**Attribute/Column Name**	**Generic Data Type Name**	**MS SQL SERVER Data Type**	**Is It Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
1)	**CreditCardNumber**	String	VARCHAR(19)	Y	19	**PRIMARY KEY**	CreditCardNumber composite primary key
2)	**CustomerID**	Number	INT	Y	Default	**PRIMARY KEY**	CustomerID composite primary key

#6

COUNTRY							
Column Num	**Attribute/Column Name**	**Generic Data Type Name**	**MS SQL Server Data Type**	**Is It Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
1)	**CountryID**	Number	TINYINT	Y	1-200	**PRIMARY KEY CHECK(CountryID between 1 and 200)**	CountryID. This has business meaning.
2)	CountryCode2Char	Character	CHAR(2)	Y	2	**UNIQUE NOT NULL**	2 Character Country name.
3)	CountryCode3Char	Character	CHAR(3)	Y	3	**UNIQUE NOT NULL**	3 Character Country name.
4)	CountryName	String	VARCHAR(56)	Y	56	**NOT NULL**	Country name.

#7

CUSTOMER							
Column Num	**Attribute/Column Name**	**Generic Data Type Name**	**MS SQL Server Data Type**	**Is It Required ?**	**Length/ Size/Format**	**Constraints**	**Description/ purpose**
1)	**CustomerID**	Number	INT IDENTITY	Y	Default	**PRIMARY KEY**	Uniquely identifies a CUSTOMER RECORD/ROW in the CUSTOMER TABLE
2)	FirstName	String	VARCHAR(50)	Y	50	**NOT NULL**	Customer First Name
3)	LastName	String	VARCHAR(50)	Y	50	**NOT NULL**	Customer Last Name
4)	BirthDate	Date	DATE	Y	MM/DD /YYYY	**NOT NULL**	Date of Birth.
5)	AddressLine1	String	VARCHAR(50)	Y	50	**NOT NULL**	House address and street number #1
6)	AddressLine2	String	VARCHAR(50)	N	50	**NULL**	House address and street number #2
7)	City	String	VARCHAR(35)	Y	35	**NOT NULL**	City name.
8)	StateCode	Character	CHAR(2)	Y	2	**NOT NULL**	State Abbreviation. Specifically U.S.
9)	ZipCode	String	VARCHAR(10)	Y	10	**NOT NULL**	Zip code.
10)	Country	String	VARCHAR(56)	Y	56	**NOT NULL**	Country name.
11)	Phone	String	VARCHAR(15)	Y	15	**NOT NULL**	Phone number.
12)	Email	String	VARCHAR(85)	Y	85	**UNIQUE NOT NULL**	Email Address.
13)	DriverLicenseNumber	String	VARCHAR(16)	Y	16	**FOREIGN KEY UNIQUE NOT NULL**	The unique driver license number.
14)	CustomerUserAccountID	GUID(Globally Unique Identifier())	UNIQUEIDENTIFIER	N	Default	**FOREIGN KEY NULL**	Stores a **GUID VALUE** generated by the **BACK-END of the MS SQL Server**
15)	CustomerType	String	BIT	Y	Default	**NOT NULL**	Customer Type 0 for corporate 1 for retail

#8

DRIVERLICENSE							
Column Num	**Attribute/Column Name**	**Generic Data Type Name**	**MS SQL SERVER Data Type**	**Is It Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
1)	**DriverLicenseNumber**	String	VARCHAR(16)	Y	16	**PRIMARY KEY**	The unique driver license number.
2)	DriverLicenseExpDate	Date	DATE	Y	MM/DD/YYYY	**NOT NULL**	The expiration date for driver license.
3)	DriverLicenseState	Character	CHAR(2)	Y	2	**NOT NULL**	State abbreviation. Specifically U.S.

#9

CUSTOMERUSERACCOUNT							
Column Num	**Attribute/Column Name**	**Generic Data Type Name**	**MS SQL SERVER Data Type**	**Is It Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
1)	**CustomerUserAccountID**	Attribute	UNIQUEIDENTIFIER	Y	Default NewID()	**PRIMARY KEY**	Stores a **GUID VALUE** generated by the **BACK-END of the MS SQL Server**
2)	Username	String	VARCHAR(50)	Y	50	**UNIQUE NOT NULL**	Customer created username.
3)	Password	String	VARCHAR(75)	Y	75	**NOT NULL**	Customer created password.
4)	Email	String	VARCHAR(85)	Y	85	**UNIQUE NOT NULL**	Customers email address.

#10

RETAILCUSTOMER							
Column Num	**Attribute/Column Name**	**Generic Data Type Name**	**MS SQL SERVER Data Type**	**Is It Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
1)	**CustomerID**	Number	INT	Y	Default	**PRIMARY KEY**	CustomerID composite primary key
2)	DiscountID	Number	INT	N	Default	**FOREIGN KEY UNIQUE NULL**	Discount ID.
3)	EZPlusID	Number	INT	N	Default	**FOREIGN KEY UNIQUE NULL**	EZPlusID.

#11

DISCOUNT							
Column Num	**Attribute/Column Name**	**Generic Data Type Name**	**MS SQL SERVER Data Type**	**Is It Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
1)	**DiscountID**	Number	INT IDENTITY	Y	Default	**PRIMARY KEY**	The unique ID used to identify each **Discount Code.** This has no business meaning thus an identity. Autogenerated number starting at 1.
2)	DiscountCode	String	CHAR(10)	Y	10	**UNIQUE NOT NULL**	The discount code.
3)	DiscountCodeDescription	String	VARCHAR(150)	Y	150	**NOT NULL**	The discount code description of what the discount offers.

#12

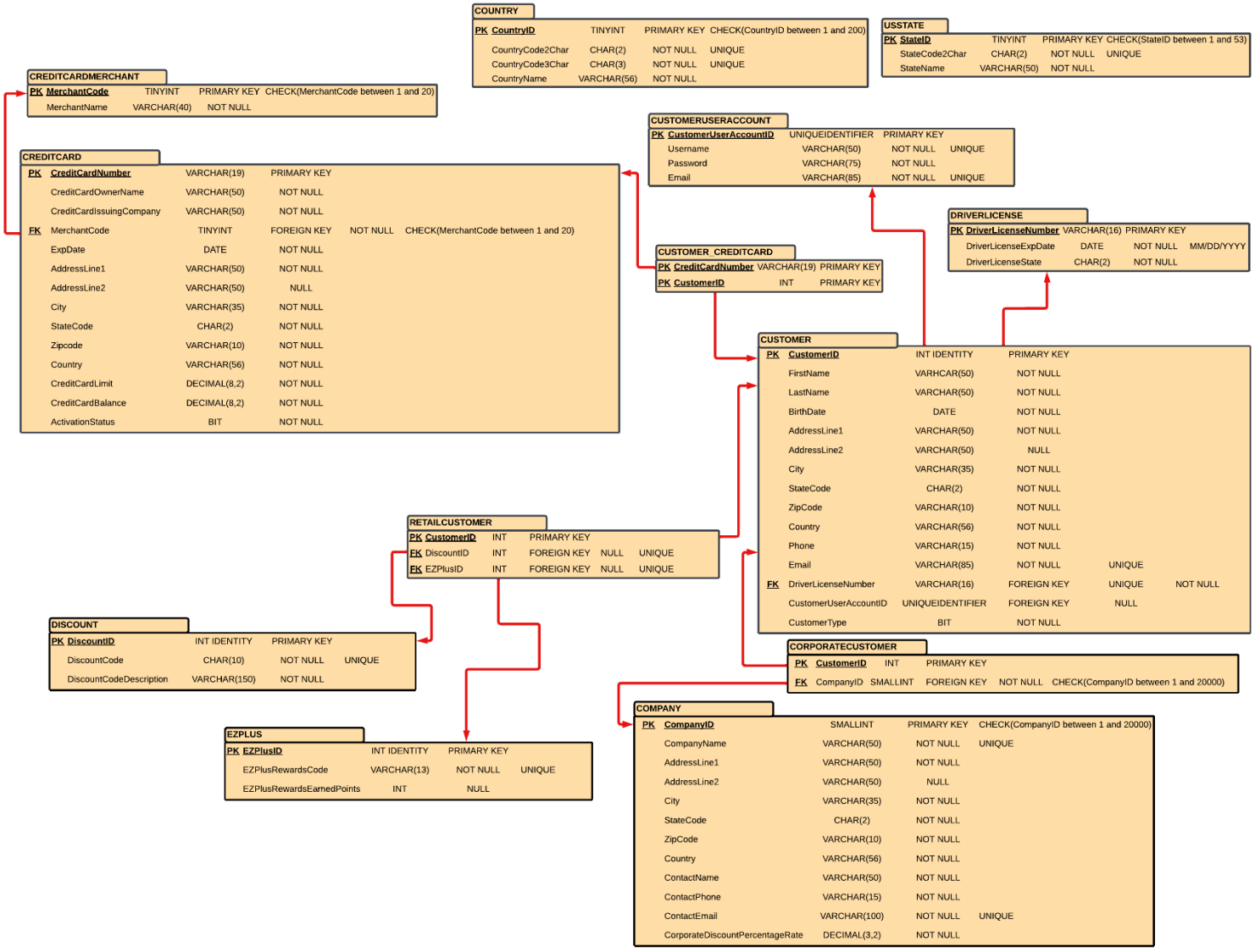
EZPLUS							
Column Num	**Attribute/Column Name**	**Generic Data Type Name**	**MS SQL SERVER Data Type**	**Is It Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
1)	**EZPlusID**	Number	INT IDENTITY	Y	Default	**PRIMARY KEY**	The EZPlusID. This has business meaning.
2)	EZPlusRewardsCode	String	VARCHAR(13)	Y	13	**UNIQUE NOT NULL**	The code used in business when managing the **EZPlus Rewards Program**
3)	EZPlusRewardsEarnedPoints	Number	INT	N	Default	**NULL**	Indicates the number of rewards points earned that accumulated after all the rentals and can be used to save on future rentals.

#13

COMPANY							
Column Num	**Attribute/Column Name**	**Generic Data Type Name**	**MS SQL Server Data Type**	**Is It Required?**	**Length/Size/Format**	**Constraints**	**Description/purpose**
1)	**CompanyID**	Number	SMALLINT	Y	1-20000	**PRIMARY KEY CHECK(CompanyID between 1 and 20000)**	The company identifier. This Primary key has business meaning.
2)	CompanyName	String	VARCHAR(50)	Y	50	**UNIQUE NOT NULL**	Company Name
3)	AddressLine1	String	VARCHAR(50)	Y	50	**NOT NULL**	House address and street number #1
4)	AddressLine2	String	VARCHAR(50)	N	50	**NULL**	House address and street number #2
5)	City	String	VARCHAR(35)	Y	35	**NOT NULL**	City name.
6)	StateCode	Character	CHAR(2)	Y	2	**NOT NULL**	State Abbreviation. Specifically U.S.
7)	ZipCode	String	VARCHAR(10)	Y	10	**NOT NULL**	Zip code.
8)	Country	String	VARCHAR(56)	Y	56	**NOT NULL**	Country name.
9)	ContactName	String	VARCHAR(50)	Y	50	**NOT NULL**	Contacts name.
10)	ContactPhone	String	VARCHAR(15)	Y	15	**NOT NULL**	Phone number.
11)	ContactEmail	String	VARCHAR(100)	Y	100	**UNIQUE NOT NULL**	Email Address. Must be unique.
12)	CorporateDiscountPercentageRate	Number	DECMAIL(3,2)	Y	X = 3 Y = 2	**NOT NULL**	The discount percentage. Example: 0.20 = 20%. 1 = 100%.

Database Design Deliverable #4 – Physical Model Schema Design Diagram

Below is an illustration of The Physical Model Schema Design Diagram which was created by combining the Normalized Logical Model Diagram with the Physical Model Data Dictionary. This is used to implement the database in the DMBS Application.



Database Implementation Deliverable #5 – Development & Implementation

Below is an illustration of the implementation/creation of the database application which was created by using the Physical Model Schema Design Diagram.

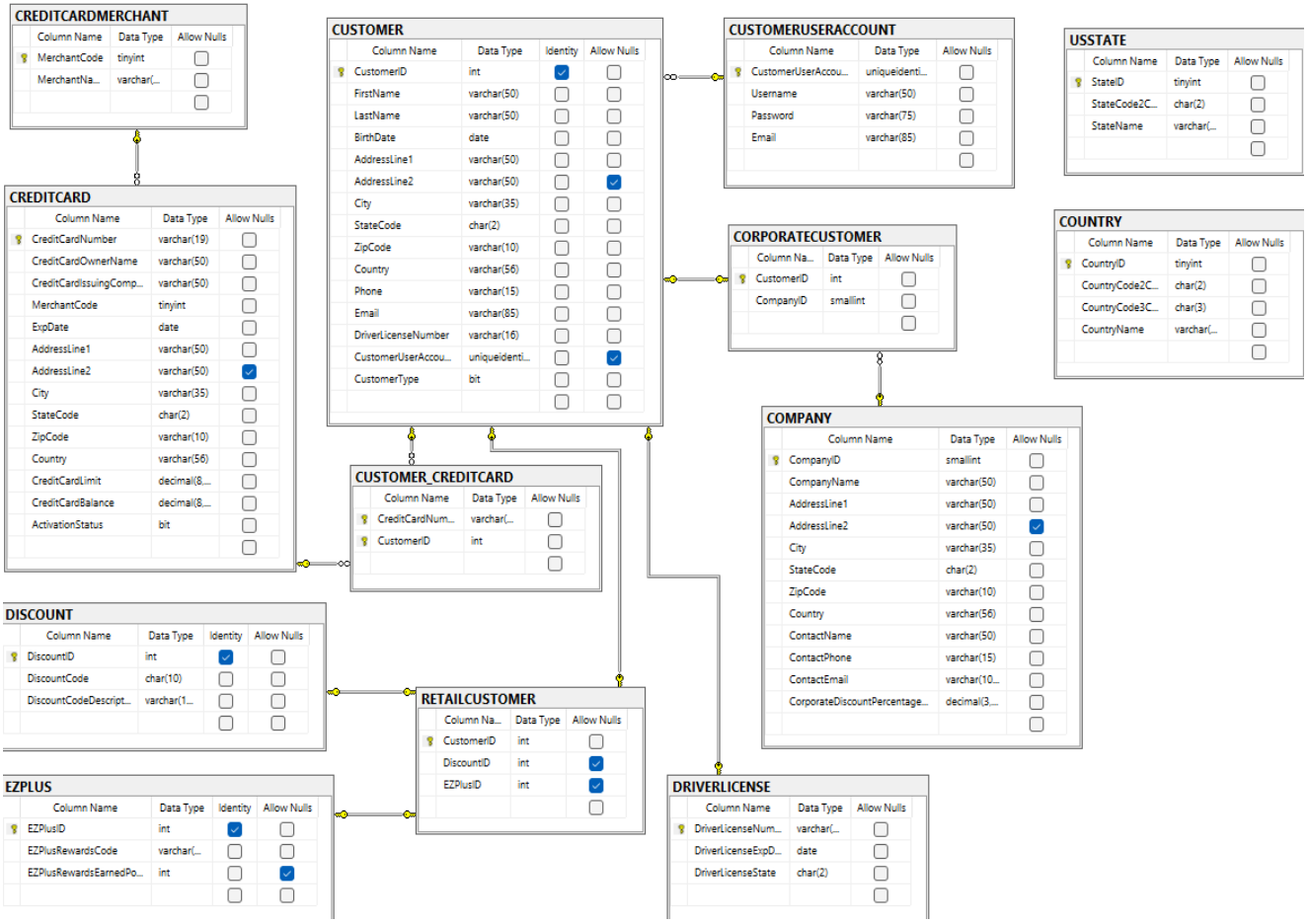
```
USE EZRentalAppDB

CREATE TABLE USSTATE
(
    StateID             TINYINT         CHECK(StateID between 1 and 53)     PRIMARY KEY,
    StateCode2Char      CHAR(2)         UNIQUE                              NOT NULL,
    StateName           VARCHAR(50)     NOT NULL
);

CREATE TABLE COUNTRY
(
    CountryID           TINYINT         CHECK(CountryID between 1 and 200)  PRIMARY KEY,
    CountryCode2Char    CHAR(2)         UNIQUE                              NOT NULL,
    CountryCode3Char    CHAR(3)         UNIQUE                              NOT NULL,
    CountryName         VARCHAR(56)     NOT NULL
);

CREATE TABLE CREDITCARDMERCHANT
(
    MerchantCode        TINYINT     CHECK(MerchantCode between 1 and 20)    PRIMARY KEY,
    MerchantName        VARCHAR(40) NOT NULL,
);

CREATE TABLE CUSTOMERUSERACCOUNT
(
    CustomerUserAccountID       UNIQUEIDENTIFIER    NOT NULL,
    Username    VARCHAR(50)     UNIQUE              NOT NULL,
    Password    VARCHAR(75)     NOT NULL,
    Email       VARCHAR(85)     UNIQUE              NOT NULL,

    CONSTRAINT pk_CustomerUserAccountID
    PRIMARY KEY(CustomerUserAccountID),
);

CREATE TABLE DRIVERLICENSE
(
    DriverLicenseNumber VARCHAR(16) PRIMARY KEY,
    DriverLicenseExpDate    DATE    NOT NULL,
    DriverLicenseState      Char(2) NOT NULL,
);

CREATE TABLE CREDITCARD
(
    CreditCardNumber            VARCHAR(19)     PRIMARY KEY,
    CreditCardOwnerName         VARCHAR(50)     NOT NULL,
    CreditCardIssuingCompany    VARCHAR(50)     NOT NULL,
    MerchantCode                TINYINT         NOT NULL    CHECK(MerchantCode between 1 and 20),
    ExpDate                     DATE            NOT NULL,
    AddressLine1                VARCHAR(50)     NOT NULL,
    AddressLine2                VARCHAR(50)     NULL,
    City                        VARCHAR(35)     NOT NULL,
    StateCode                   CHAR(2)         NOT NULL,
    ZipCode                     VARCHAR(10)     NOT NULL,
    Country                     VARCHAR(56)     NOT NULL,
    CreditCardLimit             DECIMAL(8,2)    NOT NULL,
    CreditCardBalance           DECIMAL(8,2)    NOT NULL,
    ActivationStatus            BIT             NOT NULL,

    CONSTRAINT fk_Merchant_CreditCard
    FOREIGN KEY(MerchantCode)
    REFERENCES CREDITCARDMERCHANT(MerchantCode)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE CUSTOMER
(
    CustomerID              INT IDENTITY    PRIMARY KEY,
    FirstName               VARCHAR(50)     NOT NULL,
    LastName                VARCHAR(50)     NOT NULL,
    BirthDate               DATE            NOT NULL,
    AddressLine1            VARCHAR(50)     NOT NULL,
    AddressLine2            VARCHAR(50)     NULL,
    City                    VARCHAR(35)     NOT NULL,
    StateCode               CHAR(2)         NOT NULL,
    ZipCode                 VARCHAR(10)     NOT NULL,
    Country                 VARCHAR(56)     NOT NULL,
    Phone                   VARCHAR(15)     NOT NULL,
    Email                   VARCHAR(85)     UNIQUE      NOT NULL,
    DriverLicenseNumber     VARCHAR(16)     UNIQUE      NOT NULL,
    CustomerUserAccountID   UNIQUEIDENTIFIER            NULL,
    CustomerType            BIT                         NOT NULL,

    CONSTRAINT fk_CustomerDriverLicenseNumber
    FOREIGN KEY(DriverLicenseNumber)
    REFERENCES DRIVERLICENSE(DriverLicenseNumber)
    ON DELETE CASCADE ON UPDATE CASCADE,

    CONSTRAINT fk_CustomerCustomerUserAccountID
    FOREIGN KEY(CustomerUserAccountID)
    REFERENCES CUSTOMERUSERACCOUNT(CustomerUserAccountID)
    ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE CUSTOMER_CREDITCARD
(
    CreditCardNumber    VARCHAR(19)     NOT NULL,
    CustomerID          INT             NOT NULL,

    CONSTRAINT pk_Customer_CreditCard
    PRIMARY KEY(CreditCardNumber, CustomerID),

    CONSTRAINT fk_Customer_CreditCardNumber
    FOREIGN KEY(CreditCardNumber)
    REFERENCES CREDITCARD(CreditCardNumber)
    ON DELETE CASCADE ON UPDATE CASCADE,

    CONSTRAINT fk_Customer_CustomerID
    FOREIGN KEY(CustomerID)
    REFERENCES CUSTOMER(CustomerID)
    ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE DISCOUNT
(
    DiscountID                  INT IDENTITY    PRIMARY KEY,
    DiscountCode                CHAR(10)        UNIQUE      NOT NULL,
    DiscountCodeDescription     VARCHAR(150)    NOT NULL,
);

CREATE TABLE EZPLUS
(
    EZPlusID                        INT IDENTITY    PRIMARY KEY,
    EZPlusRewardsCode               VARCHAR(13)     UNIQUE      NOT NULL,
    EZPlusRewardsEarnedPoints       INT                         NULL
);
```

```sql
CREATE TABLE RETAILCUSTOMER
(
    CustomerID      INT     NOT NULL,
    DiscountID      INT     UNIQUE   NULL,
    EZPlusID        INT     UNIQUE   NULL,

    CONSTRAINT pk_Retail_CustomerID
    PRIMARY KEY(CustomerID),

    CONSTRAINT fk_Retail_CustomerID
    FOREIGN KEY(CustomerID)
    REFERENCES CUSTOMER(CustomerID)
    ON DELETE CASCADE ON UPDATE CASCADE,

    CONSTRAINT fk_Customer_DicountID
    FOREIGN KEY(DiscountID)
    REFERENCES DISCOUNT(DiscountID)
    ON DELETE CASCADE ON UPDATE CASCADE,

    CONSTRAINT fk_Customer_EZPlusID
    FOREIGN KEY(EZPlusID)
    REFERENCES EZPLUS(EZPlusID)
    ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE COMPANY
(
    CompanyID                       SMALLINT        NOT NULL        CHECK(CompanyID between 1 and 20000)    PRIMARY KEY,
    CompanyName                     VARCHAR(50)     NOT NULL        UNIQUE,
    AddressLine1                    VARCHAR(50)     NOT NULL,
    AddressLine2                    VARCHAR(50)     NULL,
    City                            VARCHAR(35)     NOT NULL,
    StateCode                       CHAR(2)         NOT NULL,
    ZipCode                         VARCHAR(10)     NOT NULL,
    Country                         VARCHAR(56)     NOT NULL,
    ContactName                     VARCHAR(50)     NOT NULL,
    ContactPhone                    VARCHAR(15)     NOT NULL,
    ContactEmail                    VARCHAR(100)    NOT NULL        UNIQUE,
    CorporateDiscountPercentageRate DECIMAL(3,2)    NOT NULL,
);

CREATE TABLE CORPORATECUSTOMER
(
    CustomerID      INT         NOT NULL,
    CompanyID       SMALLINT    NOT NULL    CHECK(CompanyID between 1 and 20000),

    CONSTRAINT pk_Customer_CustomerID
    PRIMARY KEY(CustomerID),

    CONSTRAINT fk_Customer_CompanyID
    FOREIGN KEY(CompanyID)
    REFERENCES COMPANY(CompanyID)
    ON DELETE CASCADE ON UPDATE CASCADE,

    CONSTRAINT fk_Corporate_CustomerID
    FOREIGN KEY(CustomerID)
    REFERENCES CUSTOMER(CustomerID)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```

Database Implementation Deliverable #6 – Implemented Physical Schema Diagram

Below is an illustration of the implemented Physical Schema Diagram that shows every table and relationship created by using the MS SQL DBMS.



Database Implementation Deliverable #7 – Database Validation Testing

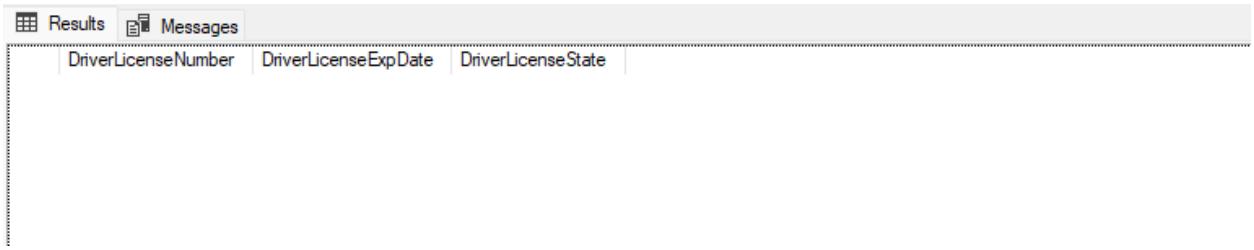
Below is the Database Validation Unit Testing in which the database developer executed SQL Data Manipulation Language (DML) statements. The Testing & Validation is done by identifying a sub-set of the Implemented Physical Schema and executing SELECT, INSERT, UPDATE & DELETE statements to validate and to test the Physical Schema:

DRIVERLICENSE

- To display the state of the **DRIVERLICENSE** Table **BEFORE** inserting records, the following SELECT statement is executed:

```
SELECT *
FROM DRIVERLICENSE;
```

- The results of executing the SELECT statement and the current state of the **DRIVERLICENSE** Table:



- The following **INSERT STATEMENTS** are executed on the **DRIVERLICENSE** Table:

```
INSERT INTO DRIVERLICENSE(DriverLicenseNumber, DriverLicenseExpDate,
DriverLicenseState)
VALUES('123456789','2029-10-31','NY')

INSERT INTO DRIVERLICENSE(DriverLicenseNumber, DriverLicenseExpDate,
DriverLicenseState)
VALUES('567890245','2025-08-31','NY')

INSERT INTO DRIVERLICENSE(DriverLicenseNumber, DriverLicenseExpDate,
DriverLicenseState)
VALUES('E100100100100','2026-05-01','MI')
```

```
INSERT INTO DRIVERLICENSE(DriverLicenseNumber, DriverLicenseExpDate,
DriverLicenseState)
VALUES('W426545307610','2027-08-16','FL')

INSERT INTO DRIVERLICENSE(DriverLicenseNumber, DriverLicenseExpDate,
DriverLicenseState)
VALUES('C65242564712846','2029-05-21','NJ')
```

- To display the state of the **DRIVERLICENSE** Table **AFTER** inserting the records, the following SELECT statement is executed:

```
SELECT *
FROM DRIVERLICENSE;
```

- The results of executing the SELECT statement and the state of the **DRIVERLICENSE** Table **AFTER** execution of the **INSERT STATEMENTS** is shown below:

Results | Messages

	DriverLicenseNumber	DriverLicenseExpDate	DriverLicenseState
1	123456789	2029-10-31	NY
2	567890245	2025-08-31	NY
3	C65242564712846	2029-05-21	NJ
4	E100100100100	2026-05-01	MI
5	W426545307610	2027-08-16	FL

CUSTOMERUSERACCOUNT

- To display the state of the **CUSTOMERUSERACCOUNT** Table **BEFORE** inserting records, the following SELECT statement is executed:

```
SELECT *
FROM CUSTOMERUSERACCOUNT;
```

- The results of executing the SELECT statement and the current state of the **CUSTOMERUSERACCOUNT** Table:

Results | Messages

CustomerUserAccountID	Username	Password	Email

- The following **INSERT STATEMENTS** are executed on the **CUSTOMERUSERACCOUNT** Table:

```
INSERT INTO CUSTOMERUSERACCOUNT(CustomerUserAccountID, Username,
Password, Email)
VALUES(NEWID(), 'iVenE','QA5gli364vsO','heathestclair@gmail.com')

INSERT INTO CUSTOMERUSERACCOUNT(CustomerUserAccountID, Username,
Password, Email)
VALUES(NEWID(), 'EngBI','u#WTXa57j03e','koneshevanv@gmail.com')

INSERT INTO CUSTOMERUSERACCOUNT(CustomerUserAccountID, Username,
Password, Email)
VALUES(NEWID(), 'Vasew','yficy%056WxN','cobbwassup@yahoo.com')

INSERT INTO CUSTOMERUSERACCOUNT(CustomerUserAccountID, Username,
Password, Email)
VALUES(NEWID(), 'ScuRyLAuT','9S5$60ypi','annasaizewa@gmail.com')

INSERT INTO CUSTOMERUSERACCOUNT(CustomerUserAccountID, Username,
Password, Email)
VALUES(NEWID(), 'TAirLiteN','136Uw856#Vx','jrr8888@gmail.com')
```

- To display the state of the **CUSTOMERUSERACCOUNT** Table **AFTER** inserting the records, the following SELECT statement is executed:

```
SELECT *
FROM CUSTOMERUSERACCOUNT;
```

- The results of executing the SELECT statement and the state of the **CUSTOMERUSERACCOUNT** Table **AFTER** execution of the **INSERT STATEMENTS** is shown below:

Results | Messages

	CustomerUserAccountID	Username	Password	Email
1	55E05098-4D64-4B16-8F83-07D4DB12D523	EngBI	u#WTXa57j03e	koneshevanv@gmail.com
2	F4EA3A1F-CF22-42DB-AEF7-08D8C6CD8281	iVenE	QA5gli364vsO	heathestclair@gmail.com
3	F132B995-F0AF-4A4C-91AD-347C41C442B6	TAirLiteN	136Uw856#Vx	jrr8888@gmail.com
4	17AF3D4F-BB61-40C7-B908-39AEDED97D22	Vasew	yficy%056WxN	cobbwassup@yahoo.com
5	426C0FC9-9FD1-4329-BE18-D469467F9CA5	ScuRyLAuT	9S5$60ypi	annasaizewa@gmail.com

CUSTOMER

- To display the state of the **CUSTOMER** Table **BEFORE** inserting records, the following SELECT statement is executed:

```
SELECT *
FROM CUSTOMER;
```

- The results of executing the SELECT statement and the current state of the **CUSTOMER** Table:

- The following **INSERT STATEMENTS** are executed on the **CUSTOMER** Table:

```
INSERT INTO CUSTOMER(FirstName, LastName, BirthDate, AddressLine1,
AddressLine2, City, StateCode, ZipCode, Country, Phone, Email,
DriverLicenseNumber, CustomerUserAccountID, CustomerType)
VALUES('John','Smith', '2000-12-20','2360 Sardis Station', NULL, 'Minneapolis', 'MN',
'55411', 'United States','612-529-9590', 'notarealemail@gmail.com', '123456789',
'55E05098-4D64-4B16-8F83-07D4DB12D523', 1)

INSERT INTO CUSTOMER(FirstName, LastName, BirthDate, AddressLine1,
AddressLine2, City, StateCode, ZipCode, Country, Phone, Email,
DriverLicenseNumber, CustomerUserAccountID, CustomerType)
VALUES('Robert','Johnson', '1999-05-12','4586 Mahlon Street', NULL,'South River',
'NJ', '08882', 'United States', '732-967-5717', 'fakeemail@yahoo.com', '567890245',
'F4EA3A1F-CF22-42DB-AEF7-08D8C6CD8281', 0)

INSERT INTO CUSTOMER(FirstName, LastName, BirthDate, AddressLine1,
AddressLine2, City, StateCode, ZipCode, Country, Phone, Email,
DriverLicenseNumber, CustomerUserAccountID, CustomerType)
VALUES('Adam','Adamson', '1982-03-16','4874 Burke Street', '4885 Burke Street',
'Boston', 'MA', '02110', 'United States', '781-332-0295', 'nachoemail@gmail.com',
'E100100100100', 'F132B995-F0AF-4A4C-91AD-347C41C442B6', 1)

INSERT INTO CUSTOMER(FirstName, LastName, BirthDate, AddressLine1,
AddressLine2, City, StateCode, ZipCode, Country, Phone, Email,
DriverLicenseNumber, CustomerUserAccountID, CustomerType)
VALUES('Chris','Will', '2002-06-20','2138 Palmer Road', NULL, 'Westerville', 'OH',
'43081', 'United States', '614-601-7282', 'totallyreal@gmail.com', 'W426545307610',
NULL, 0)
```

```
INSERT INTO CUSTOMER(FirstName, LastName, BirthDate, AddressLine1,
AddressLine2, City, StateCode, ZipCode, Country, Phone, Email,
DriverLicenseNumber, CustomerUserAccountID, CustomerType)
VALUES('Will','Power', '2000-11-12','991 Chipmunk Lane', NULL, 'West Ripley', 'ME',
'04930', 'United States', '207-277-5314', 'mygmail@gmail.com', 'C65242564712846',
'426C0FC9-9FD1-4329-BE18-D469467F9CA5', 1)
```

- To display the state of the **CUSTOMER** Table **AFTER** inserting the records, the following SELECT statement is executed:

```
SELECT *
FROM CUSTOMER;
```

- The results of executing the SELECT statement and the state of the **CUSTOMER** Table **AFTER** execution of the **INSERT STATEMENTS** is shown below:

Results | Messages

	CustomerID	FirstName	LastName	BirthDate	AddressLine1	AddressLine2
1	1	John	Smith	2000-12-20	2360 Sardis Station	NULL
2	2	Robert	Johnson	1999-05-12	4586 Mahlon Street	NULL
3	3	Adam	Adamson	1982-03-16	4874 Burke Street	4885 Burke Street
4	4	Chris	Will	2002-06-20	2138 Palmer Road	NULL
5	5	Will	Power	2000-11-12	991 Chipmunk Lane	NULL

City	StateCode	ZipCode	Country	Phone	Email
Minneapolis	MN	55411	United States	612-529-9590	notarealemail@gmail.com
South River	NJ	08882	United States	732-967-5717	fakeemail@yahoo.com
Boston	MA	02110	United States	781-332-0295	nachoemail@gmail.com
Westerville	OH	43081	United States	614-601-7282	totallyreal@gmail.com
West Ripley	ME	04930	United States	207-277-5314	mygmail@gmail.com

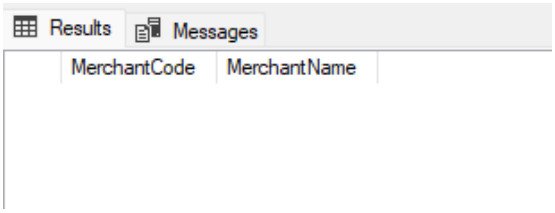
DriverLicenseNumber	CustomerUserAccountID	CustomerType
123456789	55E05098-4D64-4B16-8F83-07D4DB12D523	1
567890245	F4EA3A1F-CF22-42DB-AEF7-08D8C6CD8281	0
E100100100100	F132B995-F0AF-4A4C-91AD-347C41C442B6	1
W426545307610	NULL	0
C65242564712846	426C0FC9-9FD1-4329-BE18-D469467F9CA5	1

CREDITCARDMERCHANT

- To display the state of the **CREDITCARDMERCHANT** Table **BEFORE** inserting records, the following SELECT statement is executed:

```
SELECT *
FROM CREDITCARDMERCHANT;
```

- The results of executing the SELECT statement and the current state of the **CREDITCARDMERCHANT** Table:



- The following **INSERT STATEMENTS** are executed on the **CREDITCARDMERCHANT** Table:

```
INSERT INTO CREDITCARDMERCHANT(MerchantCode, MerchantName)
VALUES(1 ,'Stax by Fattmerchant')

INSERT INTO CREDITCARDMERCHANT(MerchantCode, MerchantName)
VALUES(2 ,'Helcim')

INSERT INTO CREDITCARDMERCHANT(MerchantCode, MerchantName)
VALUES(3 ,'Dharma Merchant Services')

INSERT INTO CREDITCARDMERCHANT(MerchantCode, MerchantName)
VALUES(4 ,'Payment Depot')

INSERT INTO CREDITCARDMERCHANT(MerchantCode, MerchantName)
VALUES(5 ,'National Processing')
```

- To display the state of the **CREDITCARDMERCHANT** Table **AFTER** inserting the records, the following SELECT statement is executed:

```
SELECT *
FROM CREDITCARDMERCHANT;
```

- The results of executing the SELECT statement and the state of the **CREDITCARDMERCHANT** Table **AFTER** execution of the **INSERT STATEMENTS** is shown below:

Results | Messages

	MerchantCode	MerchantName
1	1	Stax by Fattmerchant
2	2	Helcim
3	3	Dharma Merchant Services
4	4	Payment Depot
5	5	National Processing

CREDITCARD

- To display the state of the **CREDITCARD** Table **BEFORE** inserting records, the following SELECT statement is executed:

```
SELECT *
FROM CREDITCARD;
```

- The results of executing the SELECT statement and the current state of the **CREDITCARD** Table:

Results | Messages

CreditCardNumber	CreditCardOwnerName	CreditCardIssuingCompany	MerchantCode	ExpDate	AddressLine1	AddressLine2	City	StateCode	ZipCode	Country	CreditCardLimit	CreditCardBalance	ActivationStatus

- The following **INSERT STATEMENTS** are executed on the **CREDITCARD** Table:

```
INSERT INTO CREDITCARD(CreditCardNumber, CreditCardOwnerName,
CreditCardIssuingCompany, MerchantCode, ExpDate, AddressLine1,
AddressLine2, City, StateCode, ZipCode, Country, CreditCardLimit,
CreditCardBalance, ActivationStatus)
VALUES('4539645188469971','Eula Salazar','Visa', 5, '2024-05-01', '1006 Forest Drive',
NULL, 'Mclean', 'VA', '22101', 'United States', 2500.00, 150.00, 1)

INSERT INTO CREDITCARD(CreditCardNumber, CreditCardOwnerName,
CreditCardIssuingCompany, MerchantCode, ExpDate,  AddressLine1,
AddressLine2, City, StateCode, ZipCode, Country, CreditCardLimit,
CreditCardBalance, ActivationStatus)
VALUES('5280385063802875','Andrew Wise','Mastercard', 4, '2023-10-01', '278
Mulberry Street', NULL, 'Nacogdoches', 'TX', '75964', 'United States', 5000.00, 0.00,
1)
```

```
INSERT INTO CREDITCARD(CreditCardNumber, CreditCardOwnerName,
CreditCardIssuingCompany, MerchantCode, ExpDate,  AddressLine1,
AddressLine2, City, StateCode, ZipCode, Country, CreditCardLimit,
CreditCardBalance, ActivationStatus)
VALUES('6011428854645618','Ana Cortez','Discover', 3, '2025-08-01', '4033 Morris
Street', NULL, 'San Antonio', 'TX', '78205', 'United States', 1000.00, 25.00, 1)

INSERT INTO CREDITCARD(CreditCardNumber, CreditCardOwnerName,
CreditCardIssuingCompany, MerchantCode, ExpDate, AddressLine1,
AddressLine2, City, StateCode, ZipCode, Country, CreditCardLimit,
CreditCardBalance, ActivationStatus)
VALUES('6011579977590659','Alma Knight','Discover', 2, '2027-04-01', '2508 Park
Street', NULL, 'Martinez', 'CA', '94553', 'United States', 2000.00, 100.00, 1)

INSERT INTO CREDITCARD(CreditCardNumber, CreditCardOwnerName,
CreditCardIssuingCompany, MerchantCode, ExpDate, AddressLine1,
AddressLine2, City, StateCode, ZipCode, Country, CreditCardLimit,
CreditCardBalance, ActivationStatus)
VALUES('4489735556383027','Ervin Wood','Visa', 1, '2027-02-01', '4309 Beechwood
Drive', NULL, 'Pittsburgh', 'PA', '15222', 'United States', 1500.00, 500.00, 0)
```

- To display the state of the **CREDITCARD** Table **AFTER** inserting the records, the following SELECT statement is executed:

```
SELECT *
FROM CREDITCARD;
```

- The results of executing the SELECT statement and the state of the **CREDITCARD** Table **AFTER** execution of the **INSERT STATEMENTS** is shown below:

Results | Messages

	CreditCardNumber	CreditCardOwnerName	CreditCardIssuingCompany	MerchantCode	ExpDate	AddressLine1	AddressLine2
1	4489735556383027	Ervin Wood	Visa	1	2027-02-01	4309 Beechwood Drive	NULL
2	4539645188469971	Eula Salazar	Visa	5	2024-05-01	1006 Forest Drive	NULL
3	5280385063802875	Andrew Wise	Mastercard	4	2023-10-01	278 Mulberry Street	NULL
4	6011428854645618	Ana Cortez	Discover	3	2025-08-01	4033 Morris Street	NULL
5	6011579977590659	Alma Knight	Discover	2	2027-04-01	2508 Park Street	NULL

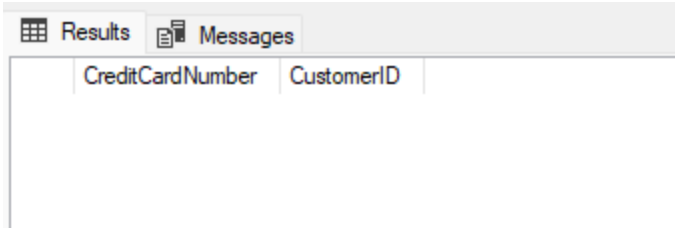
City	StateCode	ZipCode	Country	CreditCardLimit	CreditCardBalance	ActivationStatus
Pittsburgh	PA	15222	United States	1500.00	500.00	0
Mclean	VA	22101	United States	2500.00	150.00	1
Nacogdoches	TX	75964	United States	5000.00	0.00	1
San Antonio	TX	78205	United States	1000.00	25.00	1
Martinez	CA	94553	United States	2000.00	100.00	1

CUSTOMER_CREDITCARD

- To display the state of the **CUSTOMER_CREDITCARD** Table **BEFORE** inserting records, the following SELECT statement is executed:

```
SELECT *
FROM CUSTOMER_CREDITCARD;
```

- The results of executing the SELECT statement and the current state of the **CUSTOMER_CREDITCARD** Table:



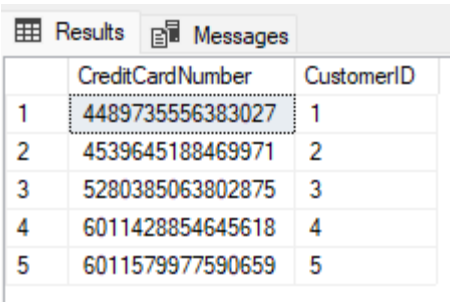
- The following **INSERT STATEMENTS** are executed on the **CUSTOMER_CREDITCARD** Table:

```
INSERT INTO CUSTOMER_CREDITCARD(CreditCardNumber, CustomerID)
VALUES('4489735556383027', 1)

INSERT INTO CUSTOMER_CREDITCARD(CreditCardNumber, CustomerID)
VALUES('4539645188469971', 2)

INSERT INTO CUSTOMER_CREDITCARD(CreditCardNumber, CustomerID)
VALUES('5280385063802875', 3)

INSERT INTO CUSTOMER_CREDITCARD(CreditCardNumber, CustomerID)
VALUES('6011428854645618', 4)

INSERT INTO CUSTOMER_CREDITCARD(CreditCardNumber, CustomerID)
VALUES('6011579977590659', 5)
```

- To display the state of the **CUSTOMER_CREDITCARD** Table **AFTER** inserting the records, the following SELECT statement is executed:

```
SELECT *
FROM CUSTOMER_CREDITCARD;
```

- The results of executing the SELECT statement and the state of the **CUSTOMER_CREDITCARD** Table **AFTER** execution of the **INSERT STATEMENTS** is shown below:



Results | Messages

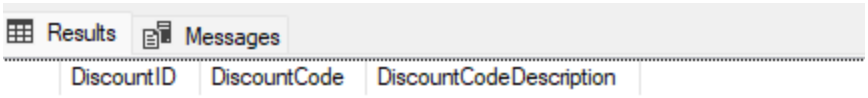
	CreditCardNumber	CustomerID
1	4489735556383027	1
2	4539645188469971	2
3	5280385063802875	3
4	6011428854645618	4
5	6011579977590659	5

DISCOUNT

- To display the state of the **DISCOUNT** Table **BEFORE** inserting records, the following SELECT statement is executed:

```
SELECT *
FROM DISCOUNT;
```

- The results of executing the SELECT statement and the current state of the **DISCOUNT** Table:



Results | Messages

DiscountID	DiscountCode	DiscountCodeDescription

- The following **INSERT STATEMENTS** are executed on the **DISCOUNT** Table:

```
INSERT INTO DISCOUNT(DiscountCode, DiscountCodeDescription)
VALUES('AAA9970054','AAA Membership Discount - 25% off base rate plus 10% donated for breast cancer research.')

INSERT INTO DISCOUNT(DiscountCode, DiscountCodeDescription)
VALUES('GOV8756921', 'Government Employee Discount - 30% off base rate.')

INSERT INTO DISCOUNT(DiscountCode, DiscountCodeDescription)
VALUES('STA3415632', 'State Employee Discount for 25% off base rate.')

INSERT INTO DISCOUNT(DiscountCode, DiscountCodeDescription)
VALUES('VET2055179', 'Veteran Discount 35% off base rate Plus 10% donation to veteran's family fund.')

INSERT INTO DISCOUNT(DiscountCode, DiscountCodeDescription)
VALUES('PET7055289', 'Pet Discount 25% off base rate one pet item.')
```

- To display the state of the **DISCOUNT** Table **AFTER** inserting the records, the following SELECT statement is executed:

```
SELECT *
FROM DISCOUNT;
```

- The results of executing the SELECT statement and the state of the **DISCOUNT** Table **AFTER** execution of the **INSERT STATEMENTS** is shown below:

Results | Messages

	DiscountID	DiscountCode	DiscountCodeDescription
1	1	AAA9970054	AAA Membership Discount - 25% off base rate plus...
2	2	GOV8756921	Government Employee Discount - 30% off base rate.
3	3	STA3415632	State Employee Discount for 25% off base rate.
4	4	VET2055179	Veteran Discount 35% off base rate Plus 10% dona...
5	5	PET7055289	Pet Discount 25% off base rate one pet item.

EZPLUS

- To display the state of the **EZPLUS** Table **BEFORE** inserting records, the following SELECT statement is executed:

```
SELECT *
FROM EZPLUS;
```

- The results of executing the SELECT statement and the current state of the **EZPLUS** Table:

Results | Messages

EZPlusID	EZPlusRewardsCode	EZPlusRewardsEarnedPoints

- The following **INSERT STATEMENTS** are executed on the **EZPLUS** Table:

```
INSERT INTO EZPLUS(EZPlusRewardsCode, EZPlusRewardsEarnedPoints)
VALUES('EZP9009854637', 10000)

INSERT INTO EZPLUS(EZPlusRewardsCode, EZPlusRewardsEarnedPoints)
VALUES('EZP1000192461', 500)

INSERT INTO EZPLUS(EZPlusRewardsCode, EZPlusRewardsEarnedPoints)
VALUES('EZP6493238865', 159000)
```

```
INSERT INTO EZPLUS(EZPlusRewardsCode, EZPlusRewardsEarnedPoints)
VALUES('EZP2005135627', 23000)

INSERT INTO EZPLUS(EZPlusRewardsCode, EZPlusRewardsEarnedPoints)
VALUES('EZP2005135634', 24000)
```

- To display the state of the **EZPLUS** Table **AFTER** inserting the records, the following SELECT statement is executed:

```
SELECT *
FROM EZPLUS;
```

- The results of executing the SELECT statement and the state of the **EZPLUS** Table **AFTER** execution of the **INSERT STATEMENTS** is shown below:

Results | Messages

	EZPlusID	EZPlusRewardsCode	EZPlusRewardsEarnedPoints
1	1	EZP9009854637	10000
2	2	EZP1000192461	500
3	3	EZP6493238865	159000
4	4	EZP2005135627	23000
5	5	EZP2005135634	24000

RETAILCUSTOMER

- To display the state of the **RETAILCUSTOMER** Table **BEFORE** inserting records, the following SELECT statement is executed:

```
SELECT *
FROM RETAILCUSTOMER;
```

- The results of executing the SELECT statement and the current state of the **RETAILCUSTOMER** Table:

Results | Messages

CustomerID	DiscountID	EZPlusID

- The following **INSERT STATEMENTS** are executed on the **RETAILCUSTOMER** Table:

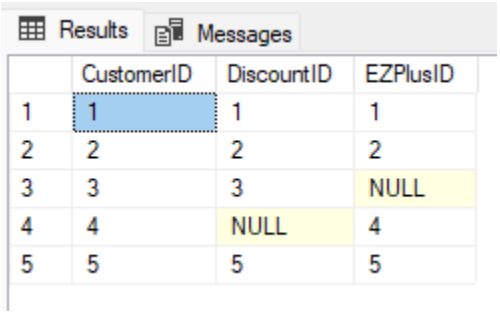
```
INSERT INTO RETAILCUSTOMER(CustomerID, DiscountID, EZPlusID)
VALUES(1, 1, 1)
```

```
INSERT INTO RETAILCUSTOMER(CustomerID, DiscountID, EZPlusID)
VALUES(2, 2, 2)

INSERT INTO RETAILCUSTOMER(CustomerID, DiscountID, EZPlusID)
VALUES(3, 3, NULL)

INSERT INTO RETAILCUSTOMER(CustomerID, DiscountID, EZPlusID)
VALUES(4, NULL, 4)

INSERT INTO RETAILCUSTOMER(CustomerID, DiscountID, EZPlusID)
VALUES(5, 5, 5)
```

- To display the state of the **RETAILCUSTOMER** Table **AFTER** inserting the records, the following SELECT statement is executed:

```
SELECT *
FROM RETAILCUSTOMER;
```

- The results of executing the SELECT statement and the state of the **RETAILCUSTOMER** Table **AFTER** execution of the **INSERT STATEMENTS** is shown below:


Results Messages

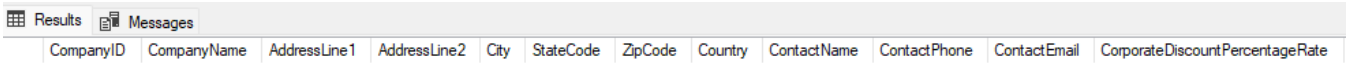
	CustomerID	DiscountID	EZPlusID
1	1	1	1
2	2	2	2
3	3	3	NULL
4	4	NULL	4
5	5	5	5

COMPANY

- To display the state of the **COMPANY** Table **BEFORE** inserting records, the following SELECT statement is executed:

```
SELECT *
FROM COMPANY;
```

- The results of executing the SELECT statement and the current state of the **COMPANY** Table:



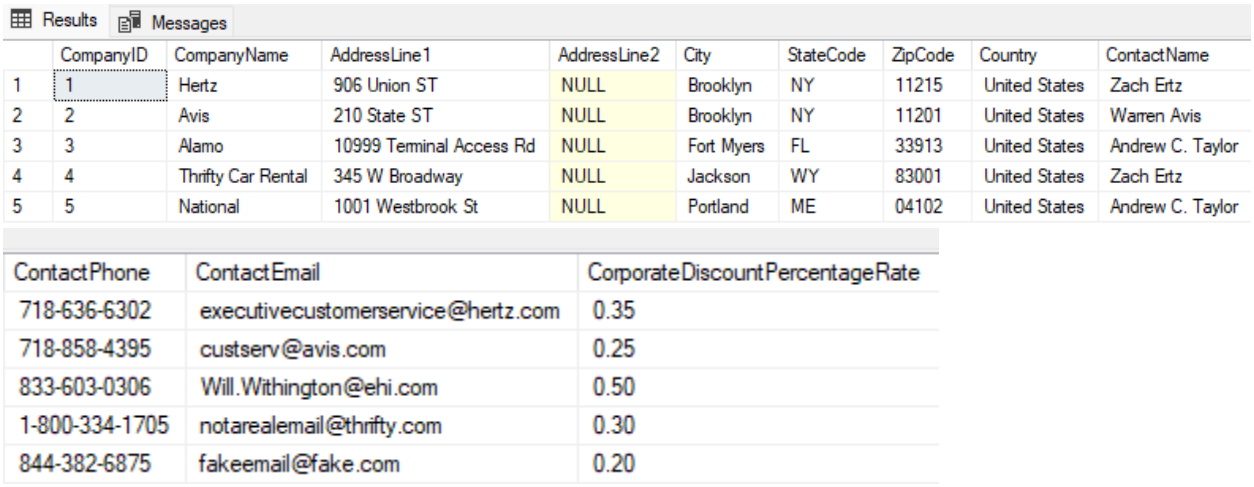
- The following **INSERT STATEMENTS** are executed on the **COMPANY** Table:

```
INSERT INTO COMPANY(CompanyID, CompanyName, AddressLine1,
AddressLine2, City, StateCode, ZipCode, Country, ContactName,
ContactPhone, ContactEmail, CorporateDiscountPercentageRate)
VALUES(1, 'Hertz', '906 Union ST', NULL, 'Brooklyn', 'NY', '11215', 'United States', 'Zach
Ertz', '718-636-6302', 'executivecustomerservice@hertz.com', 0.35)

INSERT INTO COMPANY(CompanyID, CompanyName, AddressLine1,
AddressLine2, City, StateCode, ZipCode, Country, ContactName,
ContactPhone, ContactEmail, CorporateDiscountPercentageRate)
VALUES(2, 'Avis', '210 State ST', NULL, 'Brooklyn', 'NY', '11201', 'United States',
'Warren Avis', '718-858-4395', 'custserv@avis.com', 0.25)

INSERT INTO COMPANY(CompanyID, CompanyName, AddressLine1,
AddressLine2, City, StateCode, ZipCode, Country, ContactName,
ContactPhone, ContactEmail, CorporateDiscountPercentageRate)
VALUES(3, 'Alamo', '10999 Terminal Access Rd', NULL, 'Fort Myers', 'FL', '33913',
'United States', 'Andrew C. Taylor', '833-603-0306', 'Will.Withington@ehi.com', 0.50)

INSERT INTO COMPANY(CompanyID, CompanyName, AddressLine1,
AddressLine2, City, StateCode, ZipCode, Country, ContactName,
ContactPhone, ContactEmail, CorporateDiscountPercentageRate)
VALUES(4, 'Thrifty Car Rental', '345 W Broadway', NULL, 'Jackson', 'WY', '83001',
'United States', 'Zach Ertz', '1-800-334-1705', 'notarealemail@thrifty.com', 0.30)

INSERT INTO COMPANY(CompanyID, CompanyName, AddressLine1,
AddressLine2, City, StateCode, ZipCode, Country, ContactName,
ContactPhone, ContactEmail, CorporateDiscountPercentageRate)
VALUES(5, 'National', '1001 Westbrook St', NULL, 'Portland', 'ME', '04102', 'United
States', 'Andrew C. Taylor', '844-382-6875', 'fakeemail@fake.com', 0.20)
```

- To display the state of the **COMPANY** Table **AFTER** inserting the records, the following SELECT statement is executed:

```
SELECT *
FROM COMPANY;
```

- The results of executing the SELECT statement and the state of the **COMPANY** Table **AFTER** execution of the **INSERT STATEMENTS** is shown below:



Results | Messages

	CompanyID	CompanyName	AddressLine1	AddressLine2	City	StateCode	ZipCode	Country	ContactName
1	1	Hertz	906 Union ST	NULL	Brooklyn	NY	11215	United States	Zach Ertz
2	2	Avis	210 State ST	NULL	Brooklyn	NY	11201	United States	Warren Avis
3	3	Alamo	10999 Terminal Access Rd	NULL	Fort Myers	FL	33913	United States	Andrew C. Taylor
4	4	Thrifty Car Rental	345 W Broadway	NULL	Jackson	WY	83001	United States	Zach Ertz
5	5	National	1001 Westbrook St	NULL	Portland	ME	04102	United States	Andrew C. Taylor

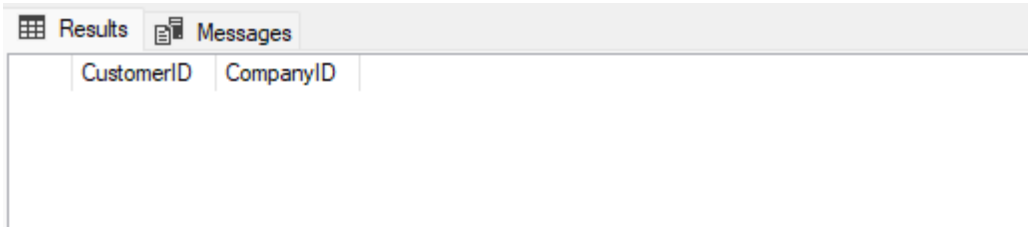
ContactPhone	ContactEmail	CorporateDiscountPercentageRate
718-636-6302	executivecustomerservice@hertz.com	0.35
718-858-4395	custserv@avis.com	0.25
833-603-0306	Will.Withington@ehi.com	0.50
1-800-334-1705	notarealemail@thrifty.com	0.30
844-382-6875	fakeemail@fake.com	0.20

CORPORATECUSTOMER

- To display the state of the **CORPORATECUSTOMER** Table **BEFORE** inserting records, the following SELECT statement is executed:

```
SELECT *
FROM CORPORATECUSTOMER;
```

- The results of executing the SELECT statement and the current state of the **CORPORATECUSTOMER** Table:



Results | Messages

CustomerID	CompanyID

- The following **INSERT STATEMENTS** are executed on the **CORPORATECUSTOMER** Table:

```
INSERT INTO CORPORATECUSTOMER(CustomerID, CompanyID)
VALUES( 1, 1)

INSERT INTO CORPORATECUSTOMER(CustomerID, CompanyID)
VALUES( 2, 3)

INSERT INTO CORPORATECUSTOMER(CustomerID, CompanyID)
VALUES( 3, 4)
```

```
INSERT INTO CORPORATECUSTOMER(CustomerID, CompanyID)
VALUES( 4, 5)

INSERT INTO CORPORATECUSTOMER(CustomerID, CompanyID)
VALUES( 5, 1)
```

- To display the state of the **CORPORATECUSTOMER** Table **AFTER** inserting the records, the following SELECT statement is executed:

```
SELECT *
FROM CORPORATECUSTOMER;
```

- The results of executing the SELECT statement and the state of the **CORPORATECUSTOMER** Table **AFTER** execution of the **INSERT STATEMENTS** is shown below:

Results | Messages

	CustomerID	CompanyID
1	1	1
2	2	3
3	3	4
4	4	5
5	5	1

SELECT STATEMENT#1

- To display one record that includes all the columns based on the primary key, the following SELECT statement is executed:

```
SELECT *
FROM CUSTOMER
WHERE CustomerID=1;
```

- The results of executing the SELECT statement is shown below:

Results | Messages

	CustomerID	FirstName	LastName	BirthDate	AddressLine1	AddressLine2	City	StateCode	ZipCode	Country	Phone	Email
1	1	John	Smith	2000-12-20	2360 Sardis Station	NULL	Minneapolis	MN	55411	United States	612-529-9590	notarealemail@gmail.com

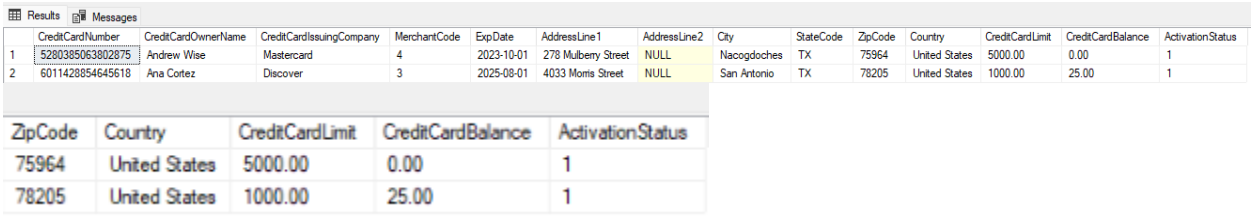
DriverLicenseNumber	CustomerUserAccountID	CustomerType
123456789	55E05098-4D64-4B16-8F83-07D4DB12D523	1

SELECT STATEMENT#2

- To display multiple records that includes all the columns based on a criteria other than the primary key that shows records with the same data, the following SELECT statement is executed:

```
SELECT *
FROM CREDITCARD
WHERE StateCode='TX';
```

- The results of executing the SELECT statement is shown below:



Results Messages

	CreditCardNumber	CreditCardOwnerName	CreditCardIssuingCompany	MerchantCode	ExpDate	AddressLine1	AddressLine2	City	StateCode	ZipCode	Country	CreditCardLimit	CreditCardBalance	ActivationStatus
1	5280385063802875	Andrew Wise	Mastercard	4	2023-10-01	278 Mulberry Street	NULL	Nacogdoches	TX	75964	United States	5000.00	0.00	1
2	6011428854645618	Ana Cortez	Discover	3	2025-08-01	4033 Morris Street	NULL	San Antonio	TX	78205	United States	1000.00	25.00	1

ZipCode	Country	CreditCardLimit	CreditCardBalance	ActivationStatus
75964	United States	5000.00	0.00	1
78205	United States	1000.00	25.00	1

SELECT STATEMENT#3

- To display records based on three queries of an associative entity set that returns records and specific columns based on a criteria, the following SELECT statement is executed:

```
SELECT CREDITCARD.CreditCardNumber, CREDITCARD.CreditCardOwnerName,
CUSTOMER.FirstName, CUSTOMER.LastName, CUSTOMER.AddressLine1,
CUSTOMER.City, CUSTOMER.StateCode,CUSTOMER.ZipCode
FROM CUSTOMER, CREDITCARD, CUSTOMER_CREDITCARD
WHERE CUSTOMER.CustomerID = CUSTOMER_CREDITCARD.CustomerID AND
CUSTOMER_CREDITCARD.CreditCardNumber = CREDITCARD.CreditCardNumber
AND CUSTOMER.AddressLine1 = CREDITCARD.AddressLine1;
```

- The results of executing the SELECT statement is shown below which shows the record of people who had the same address:

Results Messages

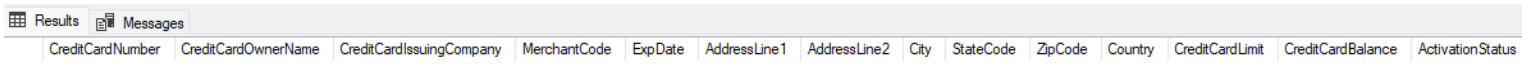
	CreditCardNumber	CreditCardOwnerName	FirstName	LastName	AddressLine1	City	StateCode	ZipCode
1	4489735556383027	Ervin Wood	John	Smith	4309 Beechwood Drive	Pittsburgh	PA	15222

UPDATE STATEMENT#1

- To display the state of the **CREDITCARD** Table **BEFORE** updating records, the following SELECT statement is executed:

```
SELECT *
FROM CREDITCARD
WHERE StateCode='MN';
```

- The results of executing the SELECT statement and the current state of the **CREDITCARD** Table:



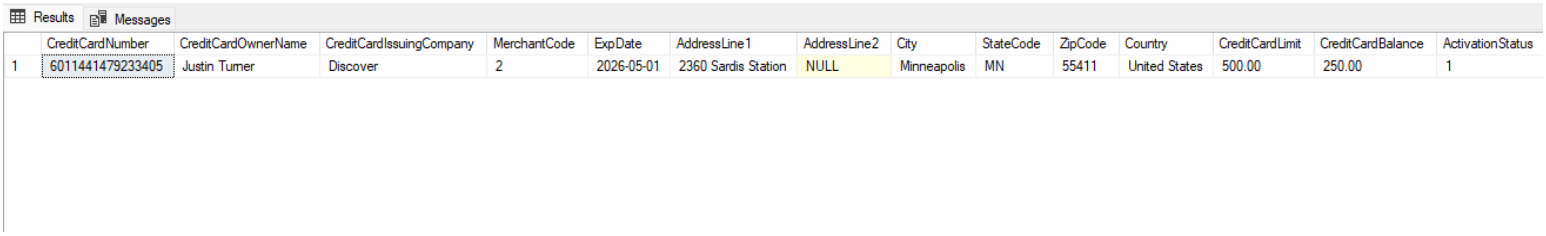
- The following **UPDATE STATEMENT** is executed on the **CREDITCARD** Table:

```
UPDATE CREDITCARD
SET CreditCardNumber='6011441479233405',
CreditCardOwnerName='Justin Turner',
CreditCardIssuingCompany='Discover',
MerchantCode= 2,
ExpDate='2026-05-01',
AddressLine1 = '2360 Sardis Station',
AddressLine2=NULL,
City = 'Minneapolis',
StateCode = 'MN',
ZipCode = '55411',
Country='United States',
CreditCardLimit= 500.00,
CreditCardBalance=250.00,
ActivationStatus=1
WHERE  CreditCardOwnerName='Ervin Wood' AND Addressline1='4309
Beechwood Drive';
```

- To display the state of the **CREDITCARD** Table **AFTER** updating the records, the following SELECT statement is executed:

```
SELECT *
FROM CREDITCARD
WHERE StateCode='MN';
```

- The results of executing the SELECT statement and the state of the **CREDITCARD** Table **AFTER** execution of the **UPDATE STATEMENT** is shown below:



Results Messages

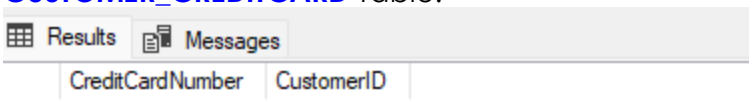
	CreditCardNumber	CreditCardOwnerName	CreditCardIssuingCompany	MerchantCode	ExpDate	AddressLine1	AddressLine2	City	StateCode	ZipCode	Country	CreditCardLimit	CreditCardBalance	ActivationStatus
1	6011441479233405	Justin Turner	Discover	2	2026-05-01	2360 Sardis Station	NULL	Minneapolis	MN	55411	United States	500.00	250.00	1

UPDATE STATEMENT#2

- To display the state of the **CUSTOMER_CREDITCARD** Table **BEFORE** updating records, the following SELECT statement is executed:

```
SELECT *
FROM CUSTOMER_CREDITCARD
WHERE CustomerID=1;
```

- The results of executing the SELECT statement and the current state of the **CUSTOMER_CREDITCARD** Table:



Results Messages

CreditCardNumber	CustomerID

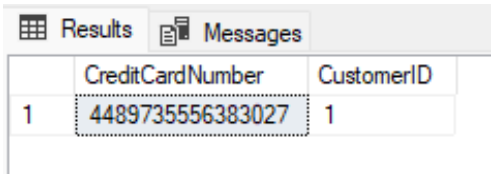
- The following **UPDATE STATEMENT** is executed on the **CUSTOMER_CREDITCARD** Table:

```
UPDATE CUSTOMER_CREDITCARD
SET CustomerID='1'
WHERE CreditCardNumber='4489735556383027';
```

- To display the state of the **CUSTOMER_CREDITCARD** Table **AFTER** updating the records, the following SELECT statement is executed:

```
SELECT *
FROM CUSTOMER_CREDITCARD
WHERE CustomerID=1;
```

- The results of executing the SELECT statement and the state of the **CUSTOMER_CREDITCARD** Table **AFTER** execution of the **UPDATE STATEMENT** is shown below:


Results Messages

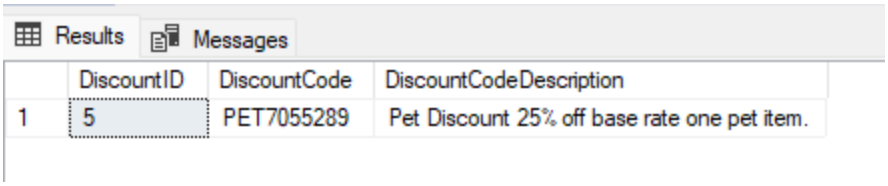
	CreditCardNumber	CustomerID
1	4489735556383027	1

DELETE_STATEMENT#1

- To display the state of the **DISCOUNT** Table **BEFORE** deleting records, the following SELECT statement is executed:

```
SELECT *
FROM DISCOUNT
WHERE DiscountID=5;
```

- The results of executing the SELECT statement and the current state of the **DISCOUNT** Table:


Results Messages

	DiscountID	DiscountCode	DiscountCodeDescription
1	5	PET7055289	Pet Discount 25% off base rate one pet item.

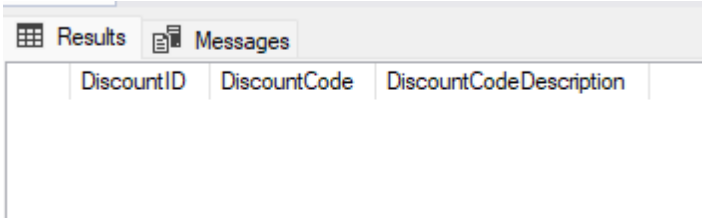
- The following **DELETE STATEMENT** is executed on the **DISCOUNT** Table:

```
DELETE
FROM DISCOUNT
WHERE DiscountID=5;
```

- To display the state of the **DISCOUNT** Table **AFTER** deleting the records, the following SELECT statement is executed:

```
SELECT *
FROM DISCOUNT
WHERE DiscountID=5;
```

- The results of executing the SELECT statement and the state of the **DISCOUNT** Table **AFTER** execution of the **DELETE STATEMENT** is shown below:


Results Messages

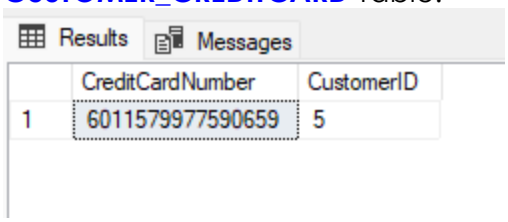
DiscountID	DiscountCode	DiscountCodeDescription

DELETE_STATEMENT#2

- To display the state of the **CUSTOMER_CREDITCARD** Table **BEFORE** deleting records, the following SELECT statement is executed:

```
SELECT *
FROM CUSTOMER_CREDITCARD
WHERE CustomerID=5;
```

- The results of executing the SELECT statement and the current state of the **CUSTOMER_CREDITCARD** Table:



- The following **DELETE STATEMENT** is executed on the **CUSTOMER_CREDITCARD** Table:

```
DELETE
FROM CUSTOMER_CREDITCARD
WHERE CustomerID=5;
```

- To display the state of the **CUSTOMER_CREDITCARD** Table **AFTER** deleting the records, the following SELECT statement is executed:

```
SELECT *
FROM CUSTOMER_CREDITCARD
WHERE CustomerID=5;
```

- The results of executing the SELECT statement and the state of the **CUSTOMER_CREDITCARD** Table **AFTER** execution of the **DELETE STATEMENT** is shown below:

Conclusion

The purpose of the project was to design and implement a suite of **Auto Rental Point-of-Sales Management System Application** that includes business modules such as **EZRental Point-of-Sales(POS)**, A **Corporate INTRANET Website** named **EZRentalCorp.com** for the ***Customer Service Representative*** and other ***employees*** in the ***rental agencies***, and an **e-commerce INTERNET Website** named **EZRental.com** for customers to make and manage reservations via the public internet. A couple things that were completed in this project is the creation of the **DATABASE APPLICATION** using the **Physical Model Schema Design Diagram** as well as the Database Validation Unit Testing done by executing **SELECT, INSERT, UPDATE & DELETE** statements to validate the database. Some key features in the **EZRental Auto Rental Management System** include allowing customers, both retail and corporate, to reserve vehicles for renting. Another feature in the system is the ability for customers to make & manage vehicle reservations, profile, account etc., via the public internet. What was also completed in the project was the creation of the back-end of the application that we can use to store and retrieve data from and that is the outcome of this project.