EXECUTIVE SUMMARY

The project contains all the procedures followed by project planning, analysis, design, development, and implementation of the Client/Server Application that allows users to use the **EZRental POS** System and e-commerce site which has the functionality such as reserving and renting the vehicle provided by the company **EZRental Inc**.

PROBLEM OBJECTIVES

The objective of the project was to design and implement a Client/Server Application Auto Rental Point-of-Sales Management System named **EZRental POS**, which includes an e-commerce site name **EZRental.com** for the vehicle renting company called **EZRental Inc**.

The **EZRental POS System** was designed which allows customers, both retail and corporate, to reserve vehicles for renting similar to existing in-person or online car rental systems such as Avis, Hertz, Budget, etc. The application was designed that supports dozens of major cities around the world. In addition, the application provides a great user experience both in the physical rental agencies as well as online system with the best competitive pricing available in the market.

TECHNICAL APPLICATION ARCHITECTURE

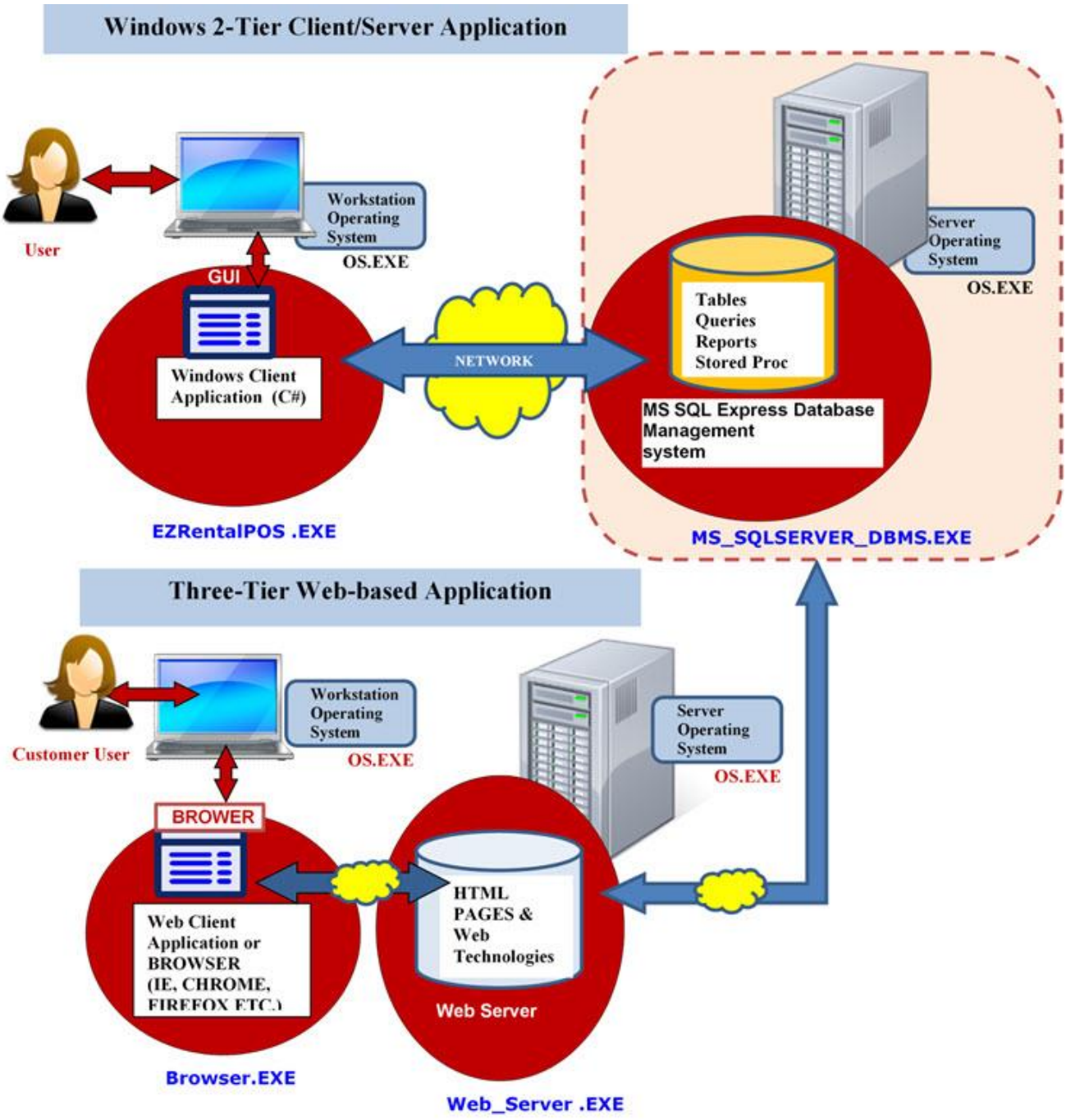
The application contains following architecture and components shown below:

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

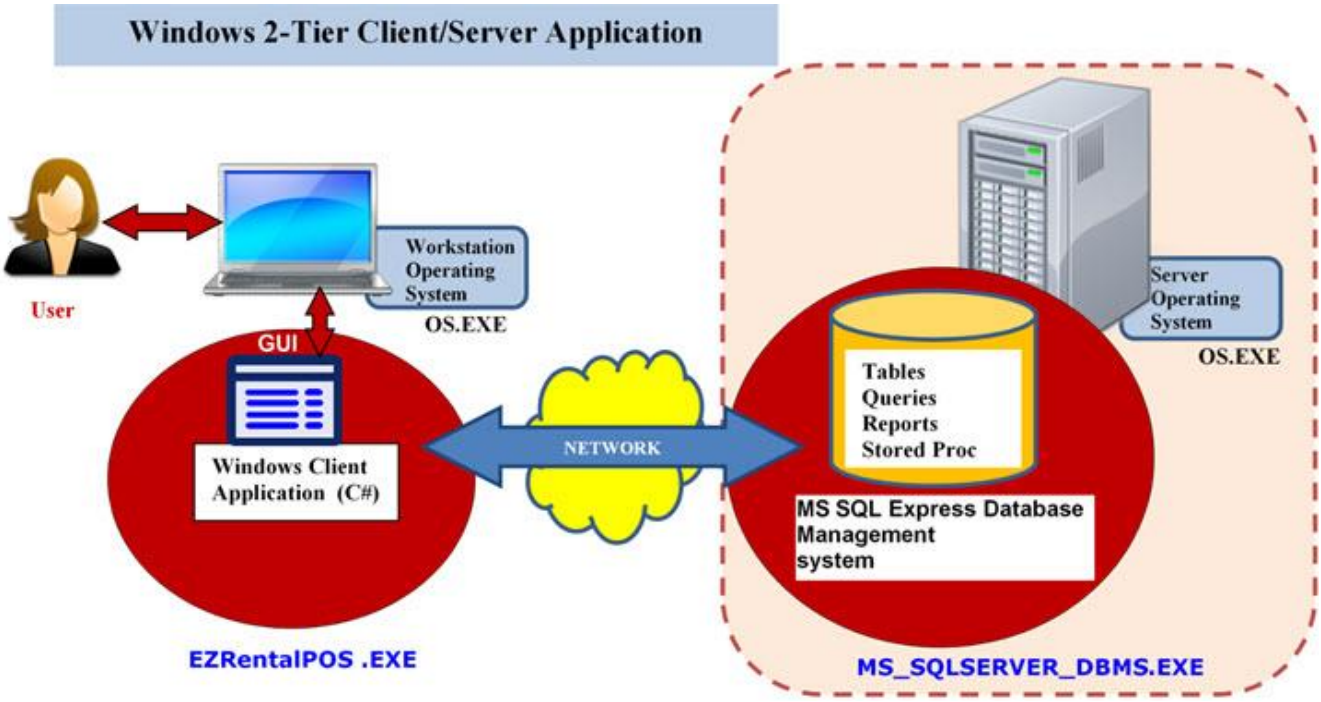
▪ **Three-Tiered Web-based Client/Server –** This Web Application named EZRentalCar.com, targeted for customers who will rent a car online, in addition to the day-to-day activities of the business and office workers personnel via a Browser Application.

▪ **Database Tier –** both the Three-tiered Web and Two-tiered Windows client/server can share the same **DATABASE TIER**.

Th technical application architecture is shown below:



Two and Three Tier Client/Server Application



Two Tier Client/Server Application

PROJECT MANAGEMENT METHODOLOGY

Project Management & Implementation Objectives:

❑ **Waterfall** and a limited **Agile Methodology** was used for the Auto Rental Management System.

❑ Below is a listing of all the **5 PHASES** & their main deliverables:

▪ **Planning –** The entire project was planned from a high-level after gathering requirements from business.

▪ **Analysis –** The business and organization were analyzed to determine requirements for developing database.

▪ **Design –** All the requirements were organized to produce a detailed specification of all data, forms, reports, displays, and processing or business rules.

▪ **Implementation –** Created the database, tables, view, stored procedures etc. Created Front-End Client programs. Tested and installed system, trained users, and documented the installation and **Standard Operation Procedures (SOP)** for the implementation process.

▪ **Maintenance\Administration –** Included all the functionalities such as: Monitor, backup & repair to the system. Contains a disaster/recovery system in place for ease of maintenance.

Project Management Methodology

BUSINESS REQUIREMENTS

From the business requirements captured, the Business Analyst has followed database design principles and created the **Conceptual Model (ER/EER)**, **Logical Model**, & **Normalized Logical Model**.

1) The Normalized Logical Model was used to create the Data Dictionary for each table in the normalized logical model,

2) Physical Model Schema Diagram was created that

3) implements the database for the application using Microsoft SQL Server DBMS.

❑ Below are 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. In addition, specialized vehicles such as trucks, motorcycles, etc. We operate in several countries with rental agency locations in the US, Canada, Mexico, UK, Japan & Australia. In each country we operate, multiple rental agencies can exist in a city. For example, New York City has 2 rental agencies in Manhattan, one in Brooklyn and two in Queens, one in each airport. With multiple rental agencies in cities, a customer can pick up a vehicle in one location and drop it off at another.

Current Challenges:

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

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

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

Our Agencies:

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

Our Customers:

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

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

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

For our corporate customers only, we must store the following properties: unique *company ID* (we have an ID number for each company), *company name*, *company contact* which is composed of *contact name*, *contact phone number & contact email*. Finally, we need to store the *company's daily rental rate*.

Retail customers are automatically enrolled in the EZPlus rewards program information, where they earn points every time they rent and can redeem these points for future rentals. In addition, retail customers are eligible for special discounts based on other businesses and organizations. Therefore, data unique to a retail customer that we need to capture the discount information: a unique random number *discount ID*, the unique *discount code* itself, and *discount code description*. For the EZPlus rewards program we need: unique random *EZPlus ID*, the unique *Ezplus rewards code*, *EZPlus rewards earned points*. Examples of common *discount ID*, *discount code*, *discount code description*, *EZPlus ID*, *EZPlus rewards Code* and *EZPlus earned points* are:

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

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

Business Requirements (Cont.)

In this business we have the following rules for our customers.

- We only have two types of customers retail customer or corporate customers. No other type of customer exists.
- A customer cannot be a retail & corporate customer at the same time. A customer can only rent as a retail customer or as a corporate. These transactions must be separate. We don't want our customers to be able to benefits of both type of customer discounts at that same time.

Reservation Process:

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

We have the following rules for reserving a vehicle:

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

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

Vehicle Rental Category ID	*Vehicle Rental Category Name*	*Category Daily Rental Rate*
1	Car-Economic	$113.99
2	Car-Compact	$115.99
3	Car-Intermediate	$116.67
4	Car-Standard	$119.99
5	Car-Full Size	$121.99
6	Car-Premium	$127.79
7	Car-Luxury	$139.99
8	SUV-Intermediate	$127.99
9	SUV-Standard	$128.99
10	SUV-Standard Elite	$135.00
11	SUV-Full Size	$148.99
12	SUV-Premium	$157.99
13	Minivan-Standard	$152.99
14	Van-Passenger Van (12 passengers)	$161.00
15	Van-Cargo Van	$19.95
16	Pick Up-Mid Size	$69.95
17	Pick Up-Full Size	$105.99
18	Motorcycle-Touring	$19.95
19	Motorcycle-Cruiser	$199.99
20	Motorcycle-Scooter	$79.95

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

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

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

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

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

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

Business Requirements (Cont.)

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

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

For a reservation we must adhere to the following rules:

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

Our Vehicles:

EZ-Car Rental needs a system to manage their vehicles for renting, maintenance, selling, etc. Vehicles are classified into 4 main types: cars, SUVs, minivans, and cargo vans. These are the vehicles most rented and available at every rental agency. Nevertheless, there are other categories of vehicles available only certain locations such as recreational vehicle, motorcycles etc. No matter what type of vehicle, all vehicle types of vehicles share the following common characteristics:

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

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

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

- A Car is a vehicle whose *trunk capacity* measured in cubic feet volume is advertised to our customers. Customers can decide which vehicles better fits their needs based on the number of luggage they are carrying. For example, a luxury Mercedes E class car has a trunk capacity of 18.5 cubic ft.
- An SUV has a *towing capacity* in pounds and the option of being *All-Wheel-Drive* or not.
- A Minivan has the option of having a *disability option package* or not.
- Finally, a Cargo Van, has a *cargo capacity* in cubic feet volume and *maximum payload* it can hold in pounds.

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

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

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

Business Requirements (Cont.)

The Rental Process:

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

For the rental process, the following rules apply:

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

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

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

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

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

- The *rental agreement ID* that uniquely identifies the rental transaction, *rental pick up date*, *rental pick up time*, *rental drop off date* and *rental drop off time*, *rental pick up odometer value*, *rental drop off odometer value* & *rental total cost* which can be calculated based on selected fuel option, insurance option, vehicle rental category price and other factors.
- In addition to the above, customers receive a vehicle with a full tank of gas and customers have the option to return the car on a full tank of gas they purchase or pay a charge to return the car as is, therefore, we need to capture the unique *rental fuel option ID*, *rental fuel option description* and *rental fuel option additional cost*. We currently use the following fuel option IDs, descriptions, and cost:

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

- In addition, 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

Business Requirements (Cont.)

- The final set of 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

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

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

Our Employees:

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

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

Security & Access:

To access our systems proper security and authentication should must be implemented to make sure only authorized users access our service agencies Point-0f-Sales & Back-End Management systems in addition to our **EZRental.com** by our customers. Therefore, for security purpose we want to separate the employee access from the customer access into 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 that identifies the employee user account, *employee username* another unique alpha-numeric that identifies each individual user, and finally the *employee password* alpha-numeric that is known only to the user, An employee can own one employee user account only, and an employee user account can only be owned by one employee only since the user account represents the identify of that one employee.

Security Access for our Customers who register for our EZ-CarRental.com web site (Customer User Accounts):
Customer who access 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.

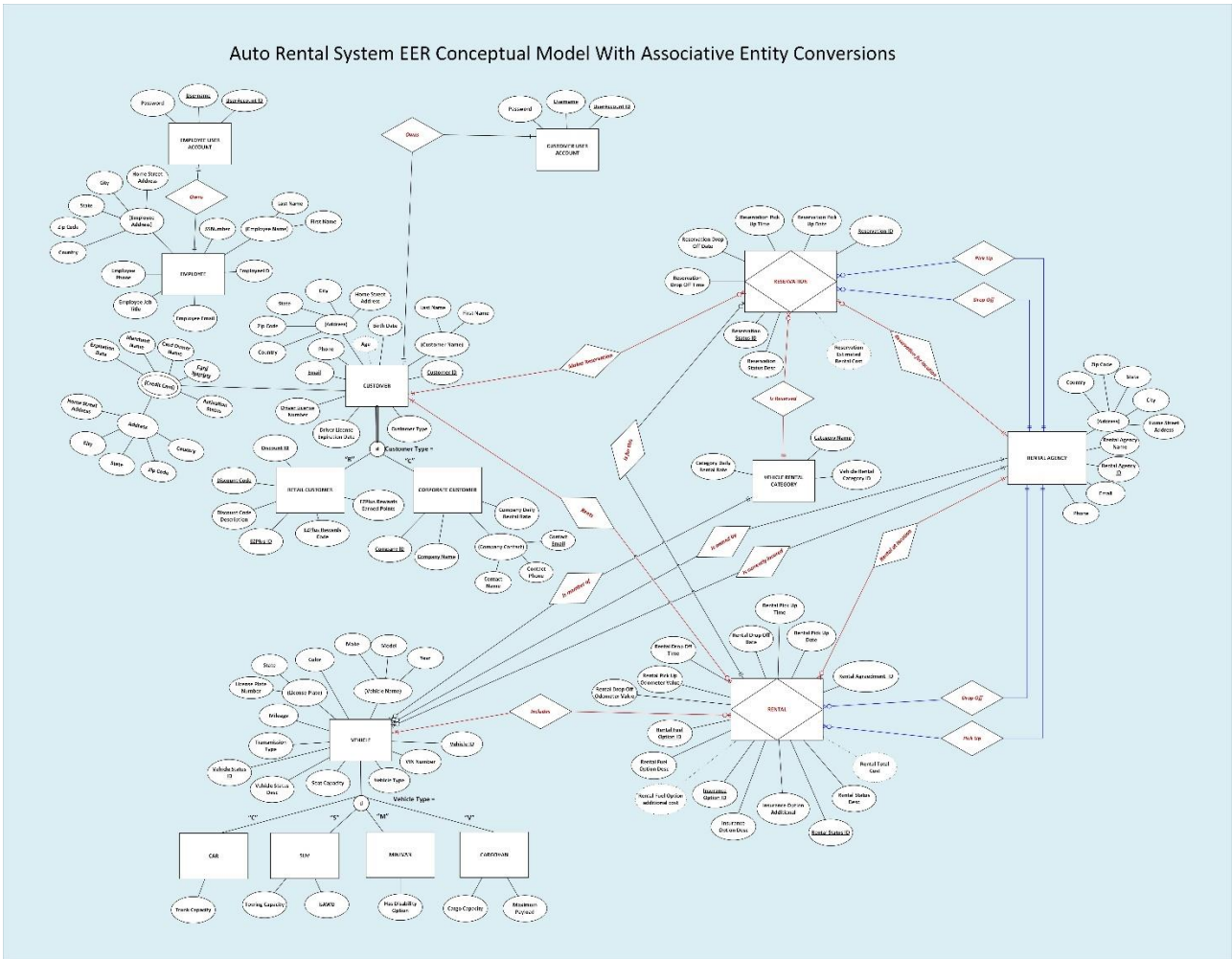
A customer user account should store the user *customer user account ID* a unique alpha-numeric identifier that identifies the customer user account, *customer username* another unique alpha-numeric value that identifies each customer, and finally, the *customer password* that is an alpha-numeric known only to the customer. A customer can own one customer user account only, and a customer user account can only be owned by one customer.

Conclusion.

For the moment, the business data listed in these requirements is what we need to capture for our business. As our business evolve, additional data will be required for example for invoice processing & employee management at our rental agencies. Our expectations of the database design is modular and scalable for growth.

ER/EER CONCEPTUAL MODEL

The **EER model** defines the design of the database which is a conceptual model incorporating extensions to the original **entity-relationship model**. The diagram shown below displays the detailed overview of the **Auto Rental System EER Conceptual Model** with Associative Entity Conversions. Furthermore, it clarifies how the data and entities should be organized for the rental system to implement the database.



Auto Rental System EER Conceptual Model with Associative Entity Conversions

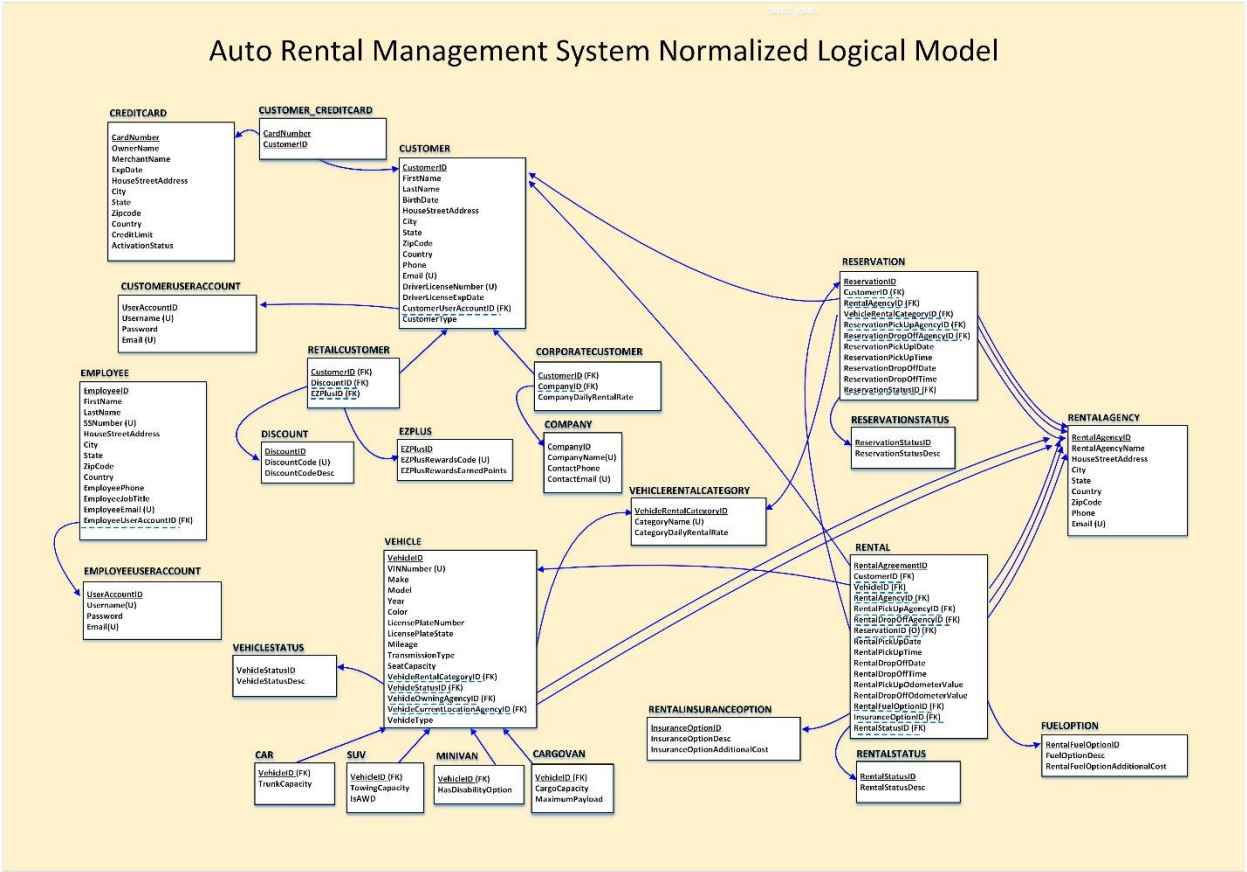
NORMALIZED LOGICAL MODEL

Normalized logical model is the model created after the data normalization in which data attributes within a data model are organized to increase the cohesion of entity types.

The data normalization is generally completed in three forms:

- *First Normal Form (1NF)*
- *Second Normal Form (2NF)*
- *Third Normal Form (3NF)*

These processes were followed for the normalization of the data in the database and the normalized logical model was created which simplifies the data model.



Auto Rental Management System Normalized Logical Model

PHYSICAL MODEL DATA DICTIONARY

The physical model contains the data dictionary with all the metadata. All the tables are listed with their attributes, properties and data types and highlighted for all the 25 tables which is taken from the Normalized Logical Model diagram.

CUSTOMER						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
CustomerID	Number	IDENTITY	Y	Default INT data type size	PRIMARY KEY	Auto-generated INTEGER IDENTITY primary key. This primary key has no business meaning.
FirstName	String	VARCHAR (50)	Y	50	NOT NULL	First name of customer
LastName	String	VARCHAR (50)	Y	50	NOT NULL	Last name of customer
BirthDate	Date	DATE	Y	MM/DD/YY	NOT NULL	Date of Birth
HouseStreetAddress	String	VARCHAR (50)	Y	50	NOT NULL	House number & street
City	String	VARCHAR (30)	Y	30	NOT NULL	City name
State	Character	CHAR (2)	Y	2	NOT NULL	State name
ZipCode	String	VARCHAR (10)	Y	10	NOT NULL	Zip code
Country	String	VARCHAR (50)	Y	50	NOT NULL	Country
Phone	String	VARCHAR (20)	Y	20	NOT NULL	Phone
Email (U)	String	VARCHAR (50)	Y	50	NOT NULL UNIQUE	Email Address
DriverLicenseNumber(U)	String	VARCHAR (30)	Y	30	NOT NULL UNIQUE	Driver License Number
DriverLicenseExpDate	Date	DATE	Y	MM/DD/YY	NOT NULL	Driver License Expiration Date
CustomerUserAccountID (FK)	Number	INT	Y	Default Integer size	FOREIGN KEY NOT NULL	ID of customer user account (Foreign Key of CustomerUserAccount)
CustomerType	String	VARCHAR (50)	Y	50	NOT NULL	Customer Type

RETAILCUSTOMER						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
CustomerID (FK)	Number	INT	Y	Default Maximum size	FOREIGN KEY NOT NULL	ID of customer who purchased (Foreign Key of Customer Table)
DiscountID (FK)	Number	INT	Y	Default Maximum size	FOREIGN KEY NOT NULL	ID of the discount made for the purchase (Foreign Key of Discount Table)
EZPlusID (FK)	Number	INT	Y	Default Maximum size	FOREIGN KEY NOT NULL	ID of the EZPLUS (Foreign Key of EZPLUS Table)

DISCOUNT						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
DiscountID	Number	IDENTITY	Y	Default INTEGER data type size	Primary Key	Auto-generated INTEGER IDENTITY primary key. This primary key has no business meaning.
DiscountCode (U)	String	VARCHAR (20)	Y	20	NOT NULL UNIQUE	Discount Code
DiscountCodeDesc	String	VARCHAR (50)	Y	50	NOT NULL	Discount Code Description

EZPLUS						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
EZPlusID	Number	IDENTITY	Y	Default INTEGER data type size	Primary Key	EZPlusRewardsCode SO PRIMARY KEY NAS NO BUSINESS MEANING SO SHOULD BE AN IDENTITY.
EZPlusRewardsCode (U)	String	VARCHAR (20)	Y	20	NOT NULL UNIQUE	EZPLUS Reward Code
EZPlusRewardsEarnedPoints	Number	INT	Y	20	NOT NULL	EZPLUS Rewards Earned Points

CORPORATECUSTOMER						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
CustomerID (FK)	Number	INT	Y	Default INTEGER data type size	FOREIGN KEY NOT NULL	ID of customer who purchased (Foreign Key of Customer Table)
CompanyID (FK)	Number	INT	Y	Default INTEGER data type size	FOREIGN KEY NOT NULL	ID of company (Foreign Key of Company Table)
CompanyDailyRentalRate	Decimal Number	DEC (9,2)	Y	X = 9 Y = 2	NOT NULL	Price has maximum of 9,999,999.99

COMPANY						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
CompanyID	Number	INT	Y	Default INTEGER data type size	Primary Key	Company ID has a business meaning.
CompanyName	String	VARCHAR (50)	Y	50	NOT NULL	Name of the Company
ContactPhone	String	VARCHAR (20)	Y	20	NOT NULL	Contact Phone Number
ContactEmail (U)	String	VARCHAR (50)	Y	50	NOT NULL UNIQUE	Email Address

CREDITCARD						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
CardNumber	Number	INT WITH LIMITS	Y	16	Primary Key	Credit Card Number
OwnerName	String	VARCHAR (50)	Y	50	NOT NULL	Owner name in Credit Card
MerchantName	String	VARCHAR (50)	Y	50	NOT NULL	Merchant name
ExpDate	Date	DATE	Y	MM/DD/YY	NOT NULL	Expiration Date of Credit Card
HouseStreetAddress	String	VARCHAR (50)	Y	50	NOT NULL	House number and street
City	String	VARCHAR (30)	Y	30	NOT NULL	City Name
State	Character	CHAR (2)	Y	2	NOT NULL	State Name
Zipcode	String	VARCHAR (10)	Y	10	NOT NULL	Zip Code
Country	String	VARCHAR (50)	Y	50	NOT NULL	Country Name
CreditLimit	Decimal Number	DEC (9,2)	Y	X = 9 Y = 2	NOT NULL	Price has maximum of 9,999,999.99
ActivationStatus	String	VARCHAR (15)	Y	15	NOT NULL	Card activation status

CUSTOMER_CREDITCARD						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
CardNumber	Number	INT WITH LIMITS	Y	16	FOREIGN KEY NOT NULL	Credit Card Number (Foreign key of Credit Card Table)
CustomerID	Number	INT	Y	Default INTEGER DATA TYPE size	FOREIGN KEY NOT NULL	ID of customer who purchased (Foreign Key of Customer Table)

CUSTOMERUSERACCOUNT						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
User Account ID	OTHER	UNIQUEIDENTIFIER	Y	Default DATA TYPE	PRIMARY KEY NOT NULL	UserAccountID UNIQUEIDENTIFIER PRIMARY KEY USING DEFAULT NEWID () FUNCTION.
Username (U)	String	VARCHAR (20)	Y	20	NOT NULL UNIQUE	Username of Customer User Account
Password	String	VARCHAR (20)	Y	20	NOT NULL	Password of Customer User Account
Email (U)	String	VARCHAR (50)	Y	50	NOT NULL UNIQUE	Email Address

VEHICLE						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
VehicleID	Number	IDENTITY	Y	Default INTEGER DATA TYPE	PRIMARY KEY	Vehicle ID of the Vehicle.
VINNumber (U)	Number	INT	Y	15	NOT NULL UNIQUE	VIN Number of Vehicle
Make	String	VARCHAR (25)	Y	25	NOT NULL	Make of vehicle
Model	String	VARCHAR (30)	Y	30	NOT NULL	Model of vehicle
Year	Number	INT	Y	10	NOT NULL	Year of vehicle
Color	String	VARCHAR (30)	Y	30	NOT NULL	Color of vehicle
LicensePlateNumber	String	VARCHAR (30)	Y	30	NOT NULL	License Plate Number of Vehicle
LicensePlateState	Character	CHAR (2)	Y	2	NOT NULL	License Plate of Vehicle
Mileage	Number	INT	Y	10	NOT NULL	Mileage of vehicle
TransmissionType	String	VARCHAR (50)	Y	50	NOT NULL	Transmission type of vehicle
SeatCapacity	Number	INT	Y	6	NOT NULL	Seat Capacity of vehicle
VehicleRentalCategoryID (FK)	Number	INT WITH LIMITS	Y	1 TO 20	FOREIGN KEY NOT NULL	(Foreign Key of Vehicle Rental Category)
VehicleStatusID (FK)	Number	INT WITH LIMITS	Y	1 TO 9	FOREIGN KEY NOT NULL	Vehicle Status ID (Foreign Key of Vehicle Status)
VehicleOwningAgencyID (FK)	Number	INT	Y	10	FOREIGN KEY NOT NULL	Vehicle Owning Agency ID
VehicleCurrentLocationAgencyID (FK)	Number	INT	Y	10	FOREIGN KEY NOT NULL	Vehicle Current Location Agency ID
VehicleType	String	VARCHAR (30)	Y	30	NOT NULL	Type of Vehicle

VEHICLESTATUS						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
VehicleStatusID	Number	INT WITH LIMITS	Y	1 TO 9	PRIMARY KEY	Status ID of Vehicle.
VehicleStatusDesc	String	VARCHAR (50)	Y	50	NOT NULL	Description of Vehicle Status

VEHICLERENTALCATEGORY						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
VehicleRentalCategoryID	Number	INT WITH LIMITS	Y	1 TO 20	PRIMARY KEY	Vehicle Rental Category ID of vehicle.
CategoryName	String	VARHCAR (50)	Y	50	NOT NULL	Category name of Vehicle
CategoryDailyRentalRate	Decimal Number	DEC (9,2)	Y	X = 9 Y = 2	NOT NULL	Price has maximum of 9,999,999.99

CAR						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
VehicleID (FK)	Number	INT	Y	Default INTEGER type	FOREIGN KEY NOT NULL	Vehicle ID (Foreign Key of Vehicle Table)
TrunkCapacity	Decimal Number	DEC (4,1)	Y	X = 4 Y = 1	NOT NULL	Trunk capacity has maximum of 999.9

SUV						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
VehicleID (FK)	Number	INT	Y	Default INTEGER type	FOREIGN KEY NOT NULL	Vehicle ID (Foreign Key of Vehicle Table)
TowingCapacity	Number	INT	Y	10	NOT NULL	Towing Capacity of SUV
IsAWD	String	VARCHAR (15)	Y	15	NOT NULL	AWD of SUV

MINIVAN						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
VehicleID (FK)	Number	INT	Y	Default INTEGER type	FOREIGN KEY NOT NULL	Vehicle ID (Foreign Key of Vehicle Table)
HasDisabilityOption	String	VARCHAR (10)	Y	10	NOT NULL	Disability option of MiniVan

CARGOVAN						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
VehicleID (FK)	Number	INT	Y	Default INTEGER type	FOREIGN KEY NOT NULL	Vehicle ID (Foreign Key of Vehicle Table)
CargoCapacity	Number	INT	Y	10	NOT NULL	Cargo Capacity of Cargo Van
MaximumPayload	Number	INT	Y	15	NOT NULL	Maximum Payload of Cargo Van

RESERVATION						
Attribute/Column Name	**General Data Type Name**	**MS SQL Oracle Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
ReservationID	Number	INT	Y	Default maximum size	PRIMARY KEY	Reservation ID for reserving vehicle
CustomerID (FK)	Number	INT	Y	Default INTEGER DATA TYPE size	FOREIGN KEY NOT NULL	ID of customer who purchased (Foreign Key of Customer Table)
RentalAgencyID (FK)	Number	INT	Y	Default INTEGER DATA TYPE size	FOREIGN KEY NOT NULL	Foreign Key of Rental Agency
VehicleRentalCategoryID (FK)	Number	INT WITH LIMITS	Y	1 TO 20	FOREIGN KEY NOT NULL	Vehicle Rental Category ID (Foreign Key of Vehicle Rental Category)
ReservationPickUpAgencyID (FK)	Number	INT	Y	15	FOREIGN KEY NOT NULL	Foreign Key of Pickup Agency
ReservationDropOffAgencyID (FK)	Number	INT	Y	15	FOREIGN KEY NOT NULL	Foreign Key of DropOff Agency
ReservationPickUpIDate	Date	DATE	Y	MM/DD/YY	NOT NULL	Reservation Pick Up Date
ReservationPickUpTime	Number	INT WITH LIMITS	Y	0000 TO 2400	NOT NULL	Reservation Pick Up Time
ReservationDropOffDate	Date	DATE	Y	MM/DD/YY	NOT NULL	Reservation Drop Off Date
ReservationDropOffTime	Number	INT WITH LIMITS	Y	0000 TO 2400	NOT NULL	Reservation Drop Off Time
ReservationStatusID (FK)	Number	INT WITH LIMITS	Y	1 TO 9	FOREIGN KEY NOT NULL	Reservation Status ID

RESERVATIONSTATUS						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
ReservationStatusID	Number	INT WITH LIMITS	Y	1 TO 9	PRIMARY KEY	Reservation Status ID
ReservationStatusDesc	String	VARCHAR (50)	Y	50	NOT NULL	Reservation Status Description

EMPLOYEE						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
EmployeeID	Number	IDENTITY	Y	Default INT Maximum Size	PRIMARY KEY	Employee ID of an Employee
FirstName	String	VARCHAR (50)	Y	50	NOT NULL	Employee FIRST NAME
LastName	String	VARCHAR (50)	Y	50	NOT NULL	Employee LAST NAME
SSNumber (U)	Char	CHAR (14)	Y	14	NOT NULL UNIQUE	Social Security Number of Employee
BirthDate	Date	DATE	Y	MM/DD/YY	NOT NULL	Employee's Date of Birth
HouseStreetAddress	String	VARCHAR (50)	Y	50	NOT NULL	House number & street
City	String	VARCHAR (30)	Y	30	NOT NULL	City name
State	Character	CHAR (2)	Y	2	NOT NULL	State name
ZipCode	String	VARCHAR (10)	Y	10	NOT NULL	Zip code
Country	String	VARCHAR (50)	Y	50	NOT NULL	Country
EmployeePhone	String	VARCHAR (20)	Y	20	NOT NULL	Phone
EmployeeJobTitle	String	VARCHAR (60)	Y	60	NOT NULL	Job title of employee
EmployeeEmail (U)	String	VARCHAR (50)	Y	50	NOT NULL UNIQUE	Email Address
EmployeeUserAccountID (FK)	Character	CHAR (36)	Y	36	FOREIGN KEY NOT NULL	Foreign Key of Employee User Account Table

RENTAL						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
RentalAgreementID	Number	INT	Y	Default Maximum Size	PRIMARY KEY	Rental Agreement ID
CustomerID (FK)	Number	INT	Y	Default INTEGER data type size	FOREIGN KEY NOT NULL	ID of customer who purchased (Foreign Key of Customer Table)
VehicleID (FK)	Number	INT	Y	Default INTEGER DATA TYPE	FOREIGN KEY NOT NULL	Vehicle ID (Foreign Key of Vehicle Table)
RentalAgencyID (FK)	Number	INT	Y	Default maximum size	FOREIGN KEY NOT NULL	Foreign Key of Rental Agency
RentalPickUpAgencyID (FK)	Number	INT	Y	15	FOREIGN KEY NOT NULL	Foreign Key of Pickup Agency
RentalDropOffAgencyID (FK)	Number	INT	Y	15	FOREIGN KEY NOT NULL	Foreign Key of Dropoff Agency
ReservationID (O) (FK)	Number	INT	Y	Default maximum size	FOREIGN KEY NULL	Optional Foreign Key of Reservation ID
RentalPickUpDate	Date	DATE	Y	MM/DD/YY	NOT NULL	Rental Pick Up Date
RentalPickUpTime	Number	INT WITH LIMITS	Y	0000 TO 2400	NOT NULL	Rental Pick Up Time
RentalDropOffDate	Date	DATE	Y	MM/DD/YY	NOT NULL	Rental Drop Off Date
RentalDropOffTime	Number	INT WITH LIMITS	Y	0000 TO 2400	NOT NULL	Rental Drop Off Time
RentalPickUpOdometerValue	Number	INT	Y	6	NOT NULL	Rental Pickup Odometer Value
RentalDropOffOdometerValue	Number	INT	Y	6	NOT NULL	Rental Dropoff Odometer Value
RentalFuelOptionID (FK)	Number	INT WITH LIMITS	Y	1 TO 9	FOREIGN KEY NOT NULL	Foreign Key of Fuel Option
InsuranceOptionID (FK)	Number	INT WITH LIMITS	Y	1 TO 9	FOREIGN KEY NOT NULL	Foreign Key of Rental Insurance Option
RentalStatusID (FK)	Number	INT WITH LIMITS	Y	1 TO 9	FOREIGN KEY NOT NULL	Foreign Key of Rental Status

RENTALINSURANCEOPTION						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
InsuranceOptionID	Number	INTEGER WITH LIMITS	Y	1 TO 9	PRIMARY KEY	Rental Insurance Option ID
InsuranceOptionDesc	String	VARCHAR (50)	Y	50	NOT NULL	Rental Insurance Option Description
InsuranceOptionAdditionalCost	Decimal Number	DEC (8,2)	Y	X = 8 Y = 2	NOT NULL	Trunk capacity has maximum of 999999.99

RENTALSTATUS						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
RentalStatusID	Number	INTEGER WITH LIMITS	Y	1 TO 9	PRIMARY KEY	Rental Status ID
RentalStatusDesc	String	VARCHAR (50)	Y	50	NOT NULL	Rental Status Description

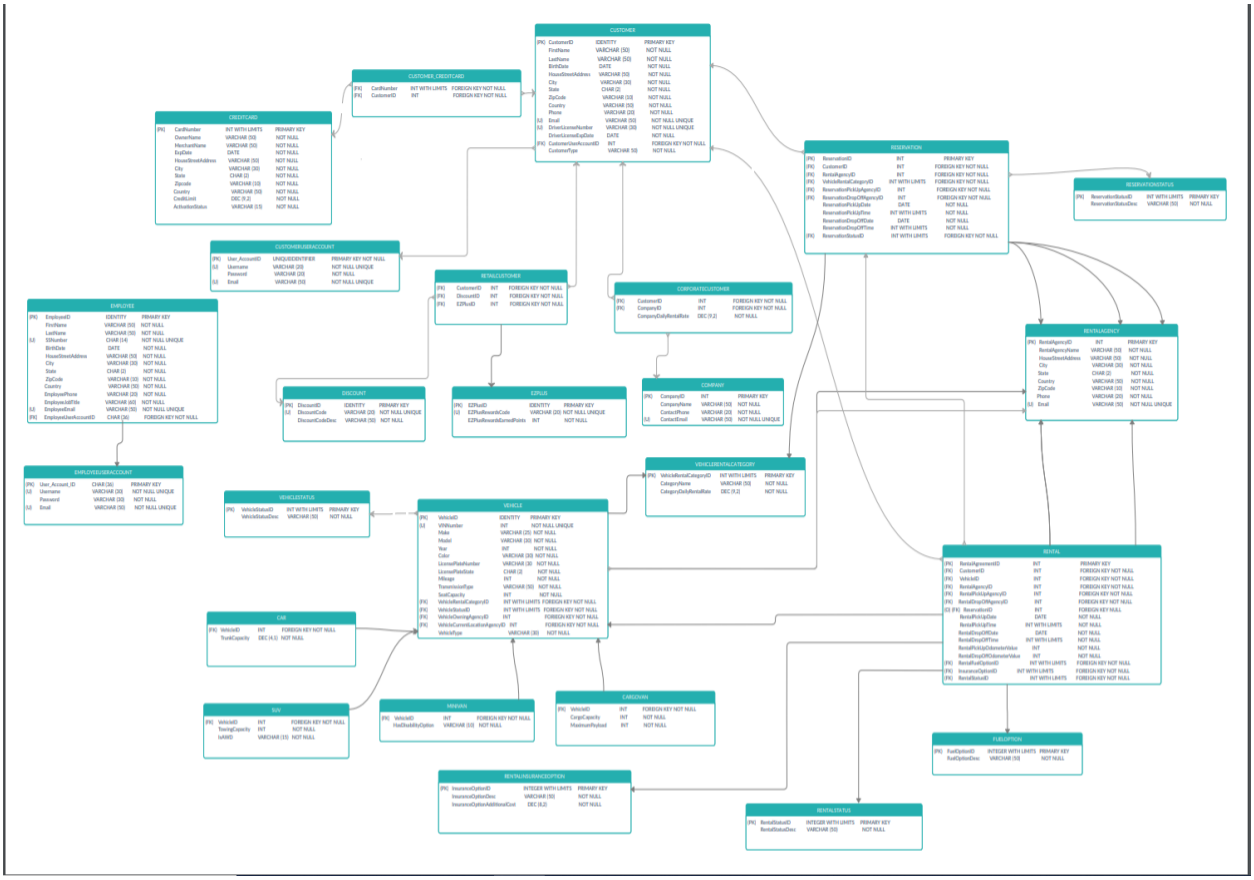
FUELOPTION						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
FuelOptionID	Number	INTEGER WITH LIMITS	Y	1 TO 9	PRIMARY KEY	Fuel Option ID
FuelOptionDesc	String	VARCHAR (50)	Y	50	NOT NULL	Fuel Option Description

EMPLOYEEUSERACCOUNT						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
User Account ID	Character	CHAR (36)	Y	36	PRIMARY KEY	Employee User Account ID
Username (U)	String	VARCHAR (30)	Y	30	NOT NULL UNIQUE	Employee Username
Password	String	VARCHAR (30)	Y	30	NOT NULL	Employee User Account Password
Email (U)	String	VARCHAR (50)	Y	50	NOT NULL UNIQUE	EMAIL ADDRESS

RENTALAGENCY						
Attribute/Column Name	**General Data Type Name**	**MS SQL Data Type Name**	**Is it Required?**	**Length/Size /Format**	**Constraints**	**Description/ purpose**
RentalAgencyID	Number	INT	Y	Default maximum size	PRIMARY KEY	Rental Agency ID of Rental Agency
RentalAgencyName	String	VARCHAR (50)	Y	50	NOT NULL	Rental Agency Name
HouseStreetAddress	String	VARCHAR (50)	Y	50	NOT NULL	House number & street
City	String	VARCHAR (30)	Y	30	NOT NULL	City name
State	Character	CHAR (2)	Y	2	NOT NULL	State name
Country	String	VARCHAR (50)	Y	50	NOT NULL	Country
ZipCode	String	VARCHAR (10)	Y	10	NOT NULL	Zip code
Phone	String	VARCHAR (20)	Y	20	NOT NULL	Phone
Email (U)	String	VARCHAR (50)	Y	50	NOT NULL UNIQUE	Email Address

PHYSICAL MODEL SCHEMA DIAGRAM

The **Physical Model Schema Diagram** is the conversion of **Data Dictionary Table** & the **Normalized Logical Model**. A drawing tool was used to create the diagram which defines all the entities and its properties in the diagram shown below:



PHYSICAL MODEL SCHEMA DIAGRAM FROM THE DATA DICTIONARY AND LOGICAL DIAGRAM

DEVELOPMENT & IMPLEMENTATION

The Development & Implementation is done using the SQL Server. The Physical Schema was used to create the Data Definition Language (DDL) Statements as shown below. Each table are created using the relationships and the file is saved separately as a Script file.

```
--Create DATABASE EZPLUSPROJECT
CREATE DATABASE EZPLUSPROJECT
--SELECT DATABASE EZPLUSPROJECT
Use EZPLUSPROJECT
--1  Create table CUSTOMERUSERACCOUNT
CREATE TABLE CustomerUserAccount
(
CustomerUserAccountID UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
Username VarChar(20) NOT NULL UNIQUE,
UserPassword VarChar(20) NOT NULL,
Email VarChar(50) NOT NULL UNIQUE
);
--2  Create table CREDITCARD
CREATE TABLE CreditCard
(
CardNumber Int PRIMARY KEY,
OwnerName VarChar(50) NOT NULL,
MerchantName VarChar(50) NOT NULL,
ExpDate Date NOT NULL,
HouseStreetAddress VarChar(50) NOT NULL,
City VarChar(30) NOT NULL,
StateCode Char(2) NOT NULL,
```

```
ZipCode VarChar(10) NOT NULL,
Country VarChar(50) NOT NULL,
CreditLimit Decimal(9,2) NOT NULL,
ActivationStatus VarChar(15) NOT NULL
);
--3 Create table CUSTOMER
CREATE TABLE Customer
(
CustomerID INT Identity PRIMARY KEY NOT NULL,
FirstName VarChar(50) NOT NULL,
LastName VarChar(50) NOT NULL,
BirthDate Date NOT NULL,
HouseStreetAddress VarChar(50) NOT NULL,
City VarChar(30) NOT NULL,
StateCode Char(2) NOT NULL,
ZipCode VarChar(10) NOT NULL,
Country VarChar(50) NOT NULL,
Phone VarChar(20) NOT NULL,
Email VarChar(50) NOT NULL UNIQUE,
DriverLicenseNumber VarChar(30) NOT NULL UNIQUE,
DriverLicenseExpDate Date NOT NULL,
CustomerUserAccountID UNIQUEIDENTIFIER NOT NULL,
CustomerType VarChar(50) NOT NULL,

CONSTRAINT fk_Customer_AccountID
FOREIGN KEY (CustomerUserAccountID)
REFERENCES CustomerUserAccount(CustomerUserAccountID)
ON DELETE CASCADE ON UPDATE CASCADE,
```

```
);
--4  Create table CUSTOMERCREDITCARD
CREATE TABLE CustomerCreditCard
(
CardNumber INT NOT NULL,
CustomerID INT NOT NULL,

CONSTRAINT fk_Creditcard_Number
FOREIGN KEY (CardNumber)
REFERENCES CreditCard(CardNumber)
ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk_Customer_CreditCard
FOREIGN KEY (CustomerID)
REFERENCES Customer(CustomerID)
ON DELETE CASCADE ON UPDATE CASCADE,
);
--5 Create table EMPLOYEEUSERACCOUNT
CREATE TABLE EmployeeUserAccount
(
EmployeeUserAccountID Char(36) PRIMARY KEY,
Username VarChar(30) NOT NULL UNIQUE,
UserPassword VarChar(30) NOT NULL,
Email VarChar(50) NOT NULL UNIQUE,
);
--6 CREATE TABLE EMPLOYEE
CREATE TABLE Employee
(
```

```
EmployeeID INT Identity PRIMARY KEY,
FirstName VarChar(50) NOT NULL,
LastName VarChar(50) NOT NULL,
SSNumber Char(14) NOT NULL UNIQUE,
BirthDate Date NOT NULL,
HouseStreetAddress VarChar(50) NOT NULL,
City VarChar(30) NOT NULL,
StateCode Char(2) NOT NULL,
ZipCode VarChar(10) NOT NULL,
Country VarChar(50) NOT NULL,
EmployeePhone VarChar(20) NOT NULL,
EmployeeJobTitle VarChar(60) NOT NULL,
EmployeeEmail VarChar(50) NOT NULL UNIQUE,
EmployeeUserAccountID Char(36) NOT NULL,

CONSTRAINT fk_Employee_UserAccount
FOREIGN KEY (EmployeeUserAccountID)
REFERENCES EmployeeUserAccount(EmployeeUserAccountID)
ON DELETE CASCADE ON UPDATE CASCADE,
);
--7 CREATE TABLE DISCOUNT
CREATE TABLE Discount
(
DiscountID INT Identity PRIMARY KEY,
DiscountCode VARCHAR(20) NOT NULL UNIQUE,
DiscountCodeDesc VARCHAR(50) NOT NULL,
);
```

```
--8 CREATE TABLE VEHICLESTATUS
CREATE TABLE VehicleStatus
(
VehicleStatusID INT PRIMARY KEY,
VehicleStatusDesc VARCHAR(50) NOT NULL,
);
--9 CREATE TABLE EZPLUS
CREATE TABLE EZPlus
(
EZPlusID INT Identity PRIMARY KEY,
EZPlusRewardsCode VARCHAR(20) NOT NULL UNIQUE,
EZPlusRewardsEarnedPoints INT NOT NULL,
);
--10 CREATE TABLE COMPANY
CREATE TABLE Company
(
CompanyID INT PRIMARY KEY,
CompanyName VARCHAR(50) NOT NULL,
ContactPhone VARCHAR(20) NOT NULL,
ContactEmail VARCHAR(50) NOT NULL UNIQUE,
);
--11 CREATE TABLE VEHICLERENTALCATEGORY
CREATE TABLE VehicleRentalCategory
(
VehicleRentalCategoryID INT PRIMARY KEY,
CategoryName VARCHAR(50) NOT NULL,
CategoryDailyRentalRate Decimal(9,2) NOT NULL,
);
```

```
--12 CREATE TABLE RENTALINSURANCEOPTION
CREATE TABLE RentalInsuranceOption
(
InsuranceOptionID INT PRIMARY KEY,
InsuranceOptionDesc VARCHAR(50) NOT NULL,
InsuranceOptionAdditionalCost Decimal(8,2) NOT NULL,
);
--13 CREATE TABLE RENTALSTATUS
CREATE TABLE RentalStatus
(
RentalStatusID  INT PRIMARY KEY,
RentalStatusDesc VARCHAR(50) NOT NULL,
);
--14 CREATE TABLE FUELOPTION
CREATE TABLE FuelOption
(
FuelOptionID INT PRIMARY KEY,
FuelOptionDesc VARCHAR(50) NOT NULL,
);
--15 CREATE TABLE RENTALAGENCY
CREATE TABLE RentalAgency
(
RentalAgencyID INT PRIMARY KEY,
RentalAgencyName VARCHAR(50) NOT NULL,
HouseStreetAddress VARCHAR(50) NOT NULL,
City VARCHAR(30) NOT NULL,
StateCode Char(2) NOT NULL,
Country VARCHAR(50) NOT NULL,
```

```
ZipCode VARCHAR(10) NOT NULL,
Phone VARCHAR(20) NOT NULL,
Email VARCHAR(50) NOT NULL UNIQUE,
);
--16 CREATE TABLE RESERVATIONSTATUS
CREATE TABLE ReservationStatus
(
ReservationStatusID INT PRIMARY KEY,
ReservationStatusDesc VARCHAR(50) NOT NULL,
);
--17 CREATE TABLE CORPORATECUSTOMER
CREATE TABLE CorporateCustomer
(
CustomerID INT NOT NULL,
CompanyID INT NOT NULL,
CompanyDailyRentalRate Decimal(9,2) NOT NULL,

CONSTRAINT fk_Customer_Corporate
FOREIGN KEY (CustomerID)
REFERENCES Customer(CustomerID)
ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk_Company_Corporate
FOREIGN KEY (CompanyID)
REFERENCES Company(CompanyID)
ON DELETE CASCADE ON UPDATE CASCADE,
);
```

```
--18 CREATE TABLE RETAILCUSTOMER
CREATE TABLE RetailCustomer
(
CustomerID INT NOT NULL,
DiscountID INT NOT NULL,
EZPlusID INT NOT NULL,

CONSTRAINT fk_Customer_Retail
FOREIGN KEY (CustomerID)
REFERENCES Customer(CustomerID)
ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk_Discount_Retail
FOREIGN KEY (DiscountID)
REFERENCES Discount(DiscountID)
ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk_EZPlus_Retail
FOREIGN KEY (EZPlusID)
REFERENCES EZPlus(EZPlusID)
ON DELETE CASCADE ON UPDATE CASCADE,
);
--19 CREATE TABLE VEHICLE
CREATE TABLE Vehicle
(
VehicleID INT Identity PRIMARY KEY,
VINNumber INT NOT NULL UNIQUE,
Make VARCHAR(25) NOT NULL,
```

```
Model VARCHAR(30) NOT NULL,
YearN INT NOT NULL,
Color VARCHAR(30) NOT NULL,
LicensePlateNumber VARCHAR(30) NOT NULL,
LicensePlateState CHAR(2) NOT NULL,
Mileage INT NOT NULL,
TransmissionType VARCHAR(50) NOT NULL,
SeatCapacity INT NOT NULL,
VehicleRentalCategoryID INT NOT NULL,
VehicleStatusID INT NOT NULL,
VehicleOwningAgencyID INT NOT NULL,
VehicleCurrentLocationAgencyID INT NOT NULL,
VehicleType VARCHAR(30) NOT NULL,

CONSTRAINT fk_VehicleRental_Vehicle
FOREIGN KEY (VehicleRentalCategoryID)
REFERENCES VehicleRentalCategory(VehicleRentalCategoryID)
ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk_VehicleStatus_Vehicle
FOREIGN KEY (VehicleStatusID )
REFERENCES VehicleStatus(VehicleStatusID )
ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk_VehicleOwning_Vehicle
FOREIGN KEY (VehicleOwningAgencyID)
REFERENCES RentalAgency(RentalAgencyID)
ON DELETE CASCADE ON UPDATE CASCADE,
```

```
CONSTRAINT fk_VehicleLocation_Vehicle
FOREIGN KEY (VehicleCurrentLocationAgencyID)
REFERENCES RentalAgency(RentalAgencyID)
ON DELETE NO ACTION ON UPDATE NO ACTION,
);
--20 CREATE TABLE CAR
CREATE TABLE Car
(
VehicleID INT NOT NULL,
TrunkCapacity Decimal(4,1) NOT NULL,

CONSTRAINT fk_Vehicle_Car
FOREIGN KEY (VehicleID)
REFERENCES Vehicle(VehicleID)
ON DELETE CASCADE ON UPDATE CASCADE,
);
--21 CREATE TABLE SUV
CREATE TABLE Suv
(
VehicleID INT NOT NULL,
TowingCapacity INT NOT NULL,
IsAWD VARCHAR(15) NOT NULL,

CONSTRAINT fk_Vehicle_Suv
FOREIGN KEY (VehicleID)
REFERENCES Vehicle(VehicleID)
ON DELETE CASCADE ON UPDATE CASCADE,
);
```

```
--22 CREATE TABLE MINIVAN
CREATE TABLE MiniVan
(
VehicleID INT NOT NULL,
HasDisabilityOption VARCHAR(10) NOT NULL,

CONSTRAINT fk_Vehicle_MiniVan
FOREIGN KEY (VehicleID)
REFERENCES Vehicle(VehicleID)
ON DELETE CASCADE ON UPDATE CASCADE,
);
--23 CREATE TABLE CARGOVAN
CREATE TABLE CargoVan
(
VehicleID INT NOT NULL,
CargoCapacity INT NOT NULL,
MaximumPayload INT NOT NULL,

CONSTRAINT fk_Vehicle_CargoVan
FOREIGN KEY (VehicleID)
REFERENCES Vehicle(VehicleID)
ON DELETE CASCADE ON UPDATE CASCADE,
);
--24 CREATE TABLE RESERVATION
CREATE TABLE Reservation
(
ReservationID INT PRIMARY KEY,
CustomerID INT NOT NULL,
```

```
RentalAgencyID INT NOT NULL,
VehicleRentalCategoryID INT NOT NULL,
ReservationPickUpAgencyID INT NOT NULL,
ReservationDropOffAgencyID INT NOT NULL,
ReservationPickUpDate Date NOT NULL,
ReservationPickUpTime INT NOT NULL,
ReservationDropOffDate Date NOT NULL,
ReservationDropOffTime INT NOT NULL,
ReservationStatusID INT NOT NULL,

CONSTRAINT fk_Customer_Reservation
FOREIGN KEY (CustomerID)
REFERENCES Customer(CustomerID)
ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk_RentalAgencyID_Reservation
FOREIGN KEY (RentalAgencyID)
REFERENCES RentalAgency(RentalAgencyID)
ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk_VehicleRentalCategoryID_Reservation
FOREIGN KEY (VehicleRentalCategoryID)
REFERENCES VehicleRentalCategory(VehicleRentalCategoryID)
ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk_ReservationPickUpAgencyID_Reservation
FOREIGN KEY (ReservationPickUpAgencyID)
REFERENCES RentalAgency(RentalAgencyID)
```

```
ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT fk_ReservationDropOffAgencyID_Reservation
FOREIGN KEY (ReservationDropOffAgencyID)
REFERENCES RentalAgency(RentalAgencyID)
ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT fk_ReservationStatusID_Reservation
FOREIGN KEY (ReservationStatusID)
REFERENCES ReservationStatus(ReservationStatusID)
ON DELETE CASCADE ON UPDATE CASCADE,
);
--25 CREATE TABLE RENTAL
CREATE TABLE Rental
(
RentalAgreementID INT PRIMARY KEY,
CustomerID INT NOT NULL,
VehicleID INT NOT NULL,
RentalAgencyID INT NOT NULL,
RentalPickUpAgencyID INT NOT NULL,
RentalDropOffAgencyID INT NOT NULL,
ReservationID INT NULL,
RentalPickUpDate Date NOT NULL,
RentalPickUpTime INT NOT NULL,
RentalDropOffDate Date NOT NULL,
RentalDropOffTime INT NOT NULL,
RentalPickUpOdometerValue INT NOT NULL,
RentalDropOffOdometerValue INT NOT NULL,
```

```
FuelOptionID INT NOT NULL,
InsuranceOptionID INT NOT NULL,
RentalStatusID INT NOT NULL,

CONSTRAINT fk_CustomerRental
FOREIGN KEY (CustomerID)
REFERENCES Customer(CustomerID)
ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk_VehicleIDRental
FOREIGN KEY (VehicleID)
REFERENCES Vehicle(VehicleID)
ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk_RentalAgencyIDRental
FOREIGN KEY (RentalAgencyID)
REFERENCES RentalAgency(RentalAgencyID)
ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT fk_RentalPickUpAgencyIDRental
FOREIGN KEY (RentalPickUpAgencyID)
REFERENCES RentalAgency(RentalAgencyID)
ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT fk_RentalDropOffAgencyIDRental
FOREIGN KEY (RentalDropOffAgencyID)
REFERENCES RentalAgency(RentalAgencyID)
ON DELETE NO ACTION ON UPDATE NO ACTION,
```

```
CONSTRAINT fk_ReservationIDRental
FOREIGN KEY (ReservationID)
REFERENCES Reservation(ReservationID)
ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT fk_RentalFuelOptionIDRental
FOREIGN KEY (FuelOptionID)
REFERENCES FuelOption(FuelOptionID)
ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT fk_InsuranceOptionIDRental
FOREIGN KEY (InsuranceOptionID)
REFERENCES RentalInsuranceOption(InsuranceOptionID)
ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT fk_RentalStatusIDRental
FOREIGN KEY (RentalStatusID)
REFERENCES RentalStatus(RentalStatusID)
ON DELETE NO ACTION ON UPDATE NO ACTION,
);
```

DEVELOPMENT & IMPLEMENTATION PHYSICAL SCHEMA DIAGRAM

The **Physical Schema Diagram was created using Microsoft SQL Management Studio. This is the fourth diagram of the Database which is shown below:**

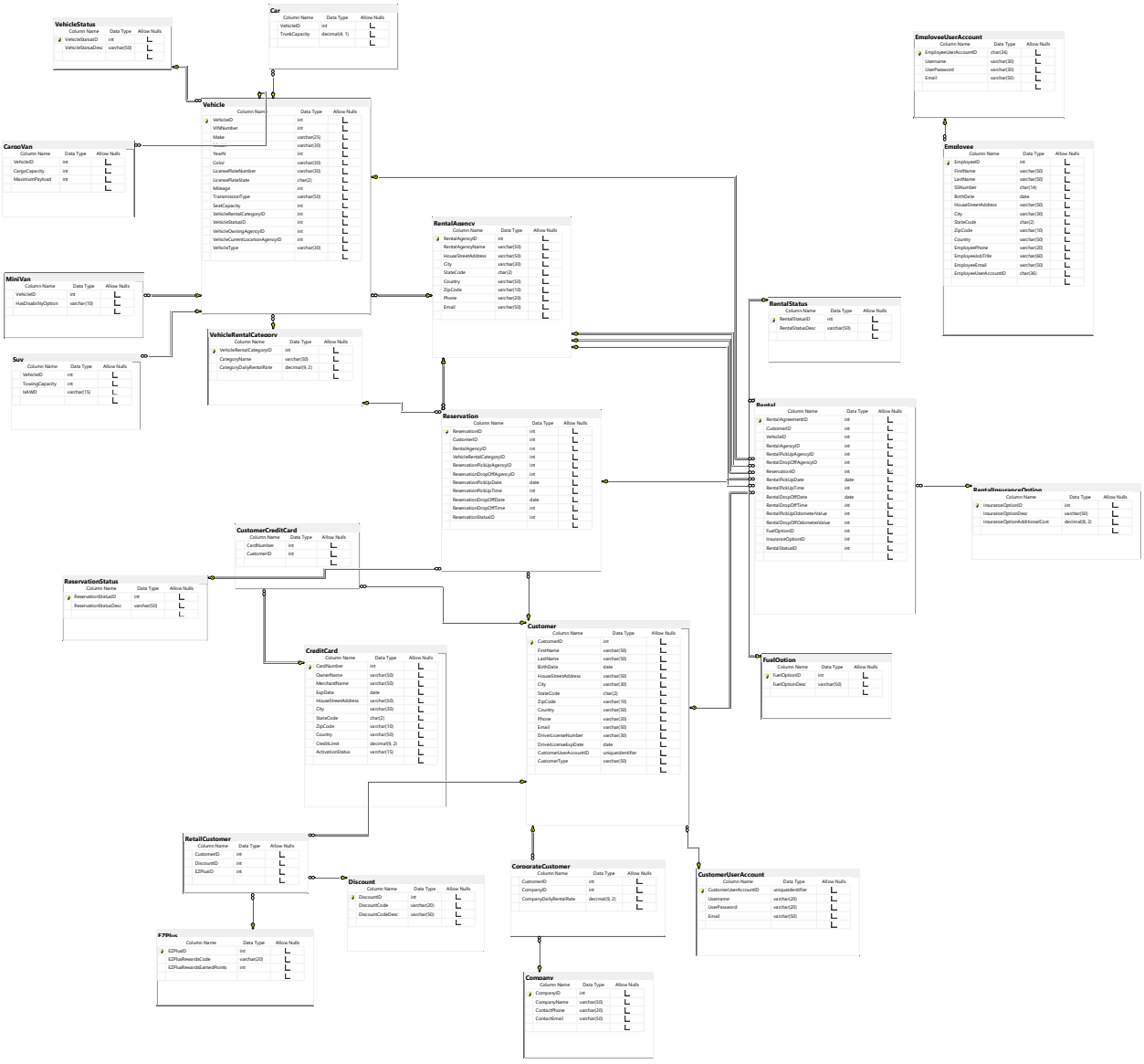


TABLE SCHEMA DIAGRAM

DATABASE DEVELOPMENT & IMPLEMENTATION UNIT TESTING

In the IMPLEMENTATION PHASE, the new script file was generated using Microsoft SQL Management Studio to Create & Execute Data Manipulation Language (DML) SQL Statements (SELECT, INSERT, UPDATE & DELETE). The SQL statements for each query are shown below:

INSERT STATEMENTS FOR CREDIT CARD TABLE

The insert statement was used to insert the data in the Credit Card table. The insert statement is shown below:

```
--INSERT VALUES INTO CREDITCARD TABLE
INSERT INTO
CreditCard(CardNumber,OwnerName,MerchantName,ExpDate,HouseStreetAddress,City,StateCode,Zi
pCode,Country,CreditLimit,ActivationStatus)
VALUES('111111111','Shital','EXPLUS','09/09/1990','11 Caryy
ST','Woodhaven','NY','33453','Nepal','25000','True');

INSERT INTO
CreditCard(CardNumber,OwnerName,MerchantName,ExpDate,HouseStreetAddress,City,StateCode,Zi
pCode,Country,CreditLimit,ActivationStatus)
VALUES('111111112','Garry','MARTIN','04/09/2000','City
ST','Bronx','SF','33422','Japan','2000','True');

INSERT INTO
CreditCard(CardNumber,OwnerName,MerchantName,ExpDate,HouseStreetAddress,City,StateCode,Zi
pCode,Country,CreditLimit,ActivationStatus)
VALUES('111111113','Darry','BONNY','01/02/2002','Molly
ST','Queens','NY','10003','Singapore','5000','False');

INSERT INTO
CreditCard(CardNumber,OwnerName,MerchantName,ExpDate,HouseStreetAddress,City,StateCode,Zi
pCode,Country,CreditLimit,ActivationStatus)
VALUES('111111114','Prince','RENTAL','03/07/2010','Dog ST','Staten
Island','NY','90003','Australia','5000','True');

INSERT INTO
CreditCard(CardNumber,OwnerName,MerchantName,ExpDate,HouseStreetAddress,City,StateCode,Zi
pCode,Country,CreditLimit,ActivationStatus)
VALUES('111111115','Joey','DOGJE','07/07/1999','11 Cat
ST','Woodhaven','NY','33453','Italy','1000','False');
```

After running the following queries, the rows were affected in the Credit Card table and each row were added one by one. To view the added rows in the table, the select statement was run and executed as shown below:

```
select * from CreditCard;
```

	CardNumber	OwnerName	MerchantName	ExpDate	HouseStreetAddress	City	StateCode	ZipCode	Country	CreditLimit	ActivationStatus
1	111111111	Shital	EXPLUS	1002-09-09	11 Caryy ST	Woodhaven	NY	33453	Nepal	25000.00	True
2	111111112	Garry	MARTIN	2000-04-09	City ST	Bronx	SF	33422	Japan	2000.00	True
3	111111113	Darry	BONNY	2002-01-02	Molly ST	Queens	NY	10003	Singapore	5000.00	False
4	111111114	Prince	RENTAL	2010-03-07	Dog ST	Staten Isla...	NY	90003	Australia	5000.00	True
5	111111115	Joey	DOGJE	1999-07-07	11 Cat ST	Woodhaven	NY	33453	Italy	1000.00	False

INSERT STATEMENTS FOR EMPLOYEEUSERACCOUNT TABLE

The insert statement was used to insert the data in the EmployeeUserAccount table. The insert statement is shown below:

```
--INSERT VALUES INTO EMPLOYEEUSERACCOUNT TABLE
INSERT INTO EmployeeUserAccount(EmployeeUserAccountID,Username,UserPassword,Email)
VALUES('33','WENDY','HASHCODE','wendy_hash@gmail.com');

INSERT INTO EmployeeUserAccount(EmployeeUserAccountID,Username,UserPassword,Email)
VALUES('34','WHITNEY','HEELO','whitney_hash@gmail.com');

INSERT INTO EmployeeUserAccount(EmployeeUserAccountID,Username,UserPassword,Email)
VALUES('35','BARRY','WORLD','barry_hash@gmail.com');

INSERT INTO EmployeeUserAccount(EmployeeUserAccountID,Username,UserPassword,Email)
VALUES('36','RONIE','CARROT','ronie_hash@gmail.com');

INSERT INTO EmployeeUserAccount(EmployeeUserAccountID,Username,UserPassword,Email)
VALUES('37','PETTY','FUNKY','petty_hash@gmail.com');
```

After running the following queries, the rows were affected in the EmployeeUserAccount table and each row were added one by one. To view the added rows in the table, the select statement was run and executed as shown below:

```
select * from EmployeeUserAccount;
```

	EmployeeUserAccountID	Username	UserPassword	Email
1	33	WENDY	HASHCODE	wendy_hash@gmail.com
2	34	WHITNEY	HEELO	whitney_hash@gmail.com
3	35	BARRY	WORLD	barry_hash@gmail.com
4	36	RONIE	CARROT	ronie_hash@gmail.com
5	37	PETTY	FUNKY	petty_hash@gmail.com

INSERT STATEMENTS FOR EMPLOYEE TABLE

The insert statement was used to insert the data in the Employee table. The insert statement is shown below:

```
--Insert Query to insert new record to Employee Table
INSERT INTO
Employee(FirstName,LastName,SSNumber,BirthDate,HouseStreetAddress,City,StateCode,ZipCode,
Country,EmployeePhone,EmployeeJobTitle,EmployeeEmail,EmployeeUserAccountID)
```

```
VALUES('Harry','Smith','224567890','01/01/1971','111 Carry
ST','WoodHaven','NY','11234','United
States','987234567','Painter','harry_smith@gmail.com','33');

INSERT INTO
Employee(FirstName,LastName,SSNumber,BirthDate,HouseStreetAddress,City,StateCode,ZipCode,
Country,EmployeePhone,EmployeeJobTitle,EmployeeEmail,EmployeeUserAccountID)
VALUES('Jack','Flown','224234567','07/02/1971','Mulberry
ST','Queens','SF','11000','United
States','9851079283','Tester','jack_flown@gmail.com','34');

INSERT INTO
Employee(FirstName,LastName,SSNumber,BirthDate,HouseStreetAddress,City,StateCode,ZipCode,
Country,EmployeePhone,EmployeeJobTitle,EmployeeEmail,EmployeeUserAccountID)
VALUES('Laxmi','Dangol','289768978','10/10/1972','Grand ST','Bronx','BA','11190','United
States','9849363170','Accountant','dangol_laxmi@gmail.com','35');

INSERT INTO
Employee(FirstName,LastName,SSNumber,BirthDate,HouseStreetAddress,City,StateCode,ZipCode,
Country,EmployeePhone,EmployeeJobTitle,EmployeeEmail,EmployeeUserAccountID)
VALUES('Pretty','Zaphar','9807645653','07/11/1871','Grand Central
ST','Manhattan','NY','12890','United
States','9843279283','Singer','pretey_zack@gmail.com','36');

INSERT INTO
Employee(FirstName,LastName,SSNumber,BirthDate,HouseStreetAddress,City,StateCode,ZipCode,
Country,EmployeePhone,EmployeeJobTitle,EmployeeEmail,EmployeeUserAccountID)
VALUES('Astha','Maskey','9908234135','12/02/1978','Boulevard
ST','Queens','NY','11223','United
States','9821567890','Auditer','astha_maskey@gmail.com','37');
```

After running the following queries, the rows were affected in the Employee table and each row were added one by one. To view the added rows in the table, the select statement was run and executed as shown below:

```
select * from Employee;
```

	EmployeeID	FirstName	LastName	SSNumber	BirthDate	HouseStreetAddress	City	StateCode	ZipCode	Country	EmployeePhone	EmployeeJobTitle	EmployeeEmail	EmployeeUserAccountID
1	19	Harry	Smith	224567890	1971-01-01	111 Carry ST	WoodHaven	NY	11234	United States	987234567	Painter	harry_smith@gmail.com	33
2	26	Jack	Flown	224234567	1971-07-02	Mulberry ST	Queens	SF	11000	United States	9851079283	Tester	jack_flown@gmail.com	34
3	29	Laxmi	Dangol	289768978	1972-10-10	Grand ST	Bronx	BA	11190	United States	9849363170	Accountant	dangol_laxmi@gmail.com	35
4	30	Pretty	Zaphar	9807645...	1871-07-11	Grand Central ST	Manhattan	NY	12890	United States	9843279283	Singer	pretey_zack@gmail.com	36
5	31	Astha	Maskey	9908234...	1978-12-02	Boulevard ST	Queens	NY	11223	United States	9821567890	Auditer	astha_maskey@gmail.c...	37

INSERT STATEMENTS FOR CUSTOMERUSERACCOUNT TABLE

The insert statement was used to insert the data in the CustomerUserAccount table. The insert statement is shown below:

```
--INSERT QUERY INTO CUSTOMERUSERACCOUNT TABLE

INSERT INTO CustomerUserAccount(Username,UserPassword,Email)
VALUES('Bikky','TukTuk','biktuk_tuk@yahoo.com');

INSERT INTO CustomerUserAccount(Username,UserPassword,Email)
VALUES('Aunty','Ganjo','aunty_ganjo@yahoo.com');

INSERT INTO CustomerUserAccount(Username,UserPassword,Email)
```

```
VALUES('Simmy','Dragon','dragon_simmy@yahoo.com');

INSERT INTO CustomerUserAccount(Username,UserPassword,Email)
VALUES('Multi','Tasking','tasking_multi@yahoo.com');

INSERT INTO CustomerUserAccount(Username,UserPassword,Email)
VALUES('Red','Berry','berry_red@aol.com');
```

After running the following queries, the rows were affected in the CustomerUserAccount table and each row were added one by one. To view the added rows in the table, the select statement was run and executed as shown below:

```
select * from CustomerUserAccount;
```

	CustomerUserAccountID	Username	UserPassword	Email
1	B107FEDE-CD5A-4736-B78B-0BD86EB353F7	Bikky	TukTuk	biktuk_tuk@yahoo.com
2	FF5757DA-CE3B-443C-99D3-340BD2716E12	Red	Berry	berry_red@aol.com
3	70BF1710-5B5E-4901-96C5-64114982EEF9	Multi	Tasking	tasking_multi@yahoo...
4	D1862F47-AB98-4B4D-985F-BAA26668EEBB	Aunty	Ganjo	aunty_ganjo@yahoo....
5	D5826BE8-130E-4D98-B47C-FE7FC775A586	Simmy	Dragon	dragon_simmy@yah...

INSERT STATEMENTS FOR CUSTOMER TABLE

The insert statement was used to insert the data in the Customer table. The insert statement is shown below:

```
--INSERT QUERY INTO CUSTOMER TABLE

INSERT INTO
Customer(FirstName,LastName,BirthDate,HouseStreetAddress,City,StateCode,ZipCode,Country,P
hone,Email,DriverLicenseNumber,DriverLicenseExpDate,
CustomerUserAccountID,CustomerType)
VALUES('Humain','Chapper','10/10/2000','Henry ST','Brooklyn','NY','11900','United
States','9349087980','humain_chapper@hotmail.com','980009','02/01/2034',
'B107FEDE-CD5A-4736-B78B-0BD86EB353F7','Regular');

INSERT INTO
Customer(FirstName,LastName,BirthDate,HouseStreetAddress,City,StateCode,ZipCode,Country,P
hone,Email,DriverLicenseNumber,DriverLicenseExpDate,
CustomerUserAccountID,CustomerType)
VALUES('Jacky','Decosta','04/02/2003','123 Whitney ST','Manhattan','NY','11001','United
States','2123458765','jacky_decosta@hotmail.com','120987','04/09/2024',
'FF5757DA-CE3B-443C-99D3-340BD2716E12','Buyers');

INSERT INTO
Customer(FirstName,LastName,BirthDate,HouseStreetAddress,City,StateCode,ZipCode,Country,P
hone,Email,DriverLicenseNumber,DriverLicenseExpDate,
CustomerUserAccountID,CustomerType)
VALUES('Maiden','Denver','1/1/2001','34 Washington ST','Canada','OH','12345','United
States','9870090233','maiden_denver@hotmail.com','980011','02/01/2022',
'70BF1710-5B5E-4901-96C5-64114982EEF9','Regular');
```

```
INSERT INTO
Customer(FirstName,LastName,BirthDate,HouseStreetAddress,City,StateCode,ZipCode,Country,P
hone,Email,DriverLicenseNumber,DriverLicenseExpDate,
CustomerUserAccountID,CustomerType)
VALUES('Dainton','Hons','10/10/2000','Henry ST','Brooklyn','NY','11900','United
States','9349087980','dainton_hons@yahoo.com','912009','07/11/2024',
'D1862F47-AB98-4B4D-985F-BAA26668EEBB','Buyers');

INSERT INTO
Customer(FirstName,LastName,BirthDate,HouseStreetAddress,City,StateCode,ZipCode,Country,P
hone,Email,DriverLicenseNumber,DriverLicenseExpDate,
CustomerUserAccountID,CustomerType)
VALUES('Pinto','Joneer','01/12/2003','Penssy ST','Queens','NY','11901','United
States','9349087777','pinto_jonner@hotmail.com','123578','02/01/2022',
'D5826BE8-130E-4D98-B47C-FE7FC775A586','Regular');
```

After running the following queries, the rows were affected in the Customer table and each row were added one by one. To view the added rows in the table, the select statement was run and executed as shown below:

```
select * from Customer;
```

	CustomerID	FirstName	LastName	BirthDate	HouseStreetAddress	City	StateCode	ZipCode	Country	Phone	Email	DriverLicenseNumber	DriverLicenseExpDate	CustomerUserAccountID	CustomerType
1	1	Humain	Chapper	2000-10-10	Henry ST	Brooklyn	NY	11900	United States	9349087980	humain_chapper@hotmail.com	980009	2034-02-01	B107FEDE-CD5A-4736-B78B-0BD86EB353F7	Regular
2	3	Jacky	Decosta	2003-04-02	123 Whitney ST	Manha...	NY	11001	United States	2123458765	jacky_decosta@hotmail.com	120987	2024-04-09	FF5757DA-CE3B-443C-99D3-340BD2716E12	Buyers
3	4	Maiden	Denver	2001-01-01	34 Washington ST	Canada	OH	12345	United States	9870090233	maiden_denver@hotmail.com	980011	2022-02-01	70BF1710-5B5E-4901-96C5-64114982EEF9	Regular
4	5	Dainton	Hons	2000-10-10	Henry ST	Brooklyn	NY	11900	United States	9349087980	dainton_hons@yahoo.com	912009	2024-07-11	D1862F47-AB98-4B4D-985F-BAA26668EEBB	Buyers
5	6	Pinto	Joneer	2003-01-12	Penssy ST	Queens	NY	11901	United States	9349087777	pinto_jonner@hotmail.com	123578	2022-02-01	D5826BE8-130E-4D98-B47C-FE7FC775A586	Regular

INSERT STATEMENTS FOR COMPANY TABLE

The insert statement was used to insert the data in the Company table. The insert statement is shown below:

```
--INSERT QUERY INTO COMPANY TABLE
INSERT INTO Company(CompanyID,CompanyName,ContactPhone,ContactEmail)
VALUES('23','EY','304989909','ernest_young@gmail.com');

INSERT INTO Company(CompanyID,CompanyName,ContactPhone,ContactEmail)
VALUES('24','Accenture','987098768','accenture@gmail.com');

INSERT INTO Company(CompanyID,CompanyName,ContactPhone,ContactEmail)
VALUES('25','Microsoft','987023456','microsoft123@gmail.com');

INSERT INTO Company(CompanyID,CompanyName,ContactPhone,ContactEmail)
VALUES('26','Google','6067892345','google_shot@gmail.com');

INSERT INTO Company(CompanyID,CompanyName,ContactPhone,ContactEmail)
VALUES('27','Amazon','2345678912','amazon_web@gmail.com');
```

After running the following queries, the rows were affected in the Company table and each row were added one by one. To view the added rows in the table, the select statement was run and executed as shown below:

```
select * from company;
```

	CompanyID	CompanyName	ContactPhone	ContactEmail
1	23	EY	304989909	ernest_young@gmail.com
2	24	Accenture	987098768	accenture@gmail.com
3	25	Microsoft	987023456	microsoft123@gmail.com
4	26	Google	6067892345	google_shot@gmail.com
5	27	Amazon	2345678912	amazon_web@gmail.com

INSERT STATEMENTS FOR CORPORATECUSTOMER TABLE

The insert statement was used to insert the data in the CorporateCustomer table. The insert statement is shown below:

```
--INSERT QUERY INTO CORPORATECUSTOMER TABLE
INSERT INTO CorporateCustomer(CustomerID,CompanyID,CompanyDailyRentalRate)
VALUES('1','23','80.00');

INSERT INTO CorporateCustomer(CustomerID,CompanyID,CompanyDailyRentalRate)
VALUES('3','24','800.00');

INSERT INTO CorporateCustomer(CustomerID,CompanyID,CompanyDailyRentalRate)
VALUES('4','25','23.50');

INSERT INTO CorporateCustomer(CustomerID,CompanyID,CompanyDailyRentalRate)
VALUES('5','26','50.30');

INSERT INTO CorporateCustomer(CustomerID,CompanyID,CompanyDailyRentalRate)
VALUES('6','27','33.88');
```

After running the following queries, the rows were affected in the CorporateCustomer table and each row were added one by one. To view the added rows in the table, the select statement was run and executed as shown below:

```
select * from CorporateCustomer;
```

	CustomerID	CompanyID	CompanyDailyRentalRate
1	1	23	80.00
2	1	23	80.00
3	1	23	80.00
4	3	24	800.00
5	4	25	23.50
6	5	26	50.30
7	6	27	33.88

SELECT STATEMENTS FOR CUSTOMER TABLE

❑ Returning all columns from SPECIFIC record via WHERE CLAUSE:

▪ Example of select statement using WHERE clause to return a customer record based on Customer_ID:

```
--SELECT STATEMENT #1 FOR CUSTOMER TABLE
select * from Customer
where CustomerID = '5';
```

• Results:

	CustomerID	FirstName	LastName	BirthDate	HouseStreetAddress	City	StateCode	ZipCode	Country	Phone	Email	DriverLicenseNumber	DriverLicenseExpDate	CustomerUserAccountID	CustomerType
1	5	Dainton	Hons	2000-10-10	Henry ST	Brooklyn	NY	11900	United States	9349087980	dainton_hons@yahoo.com	912009	2024-07-11	D1862F47-AB98-4B4D-985F-BAA26668EEBB	Buyers

SELECT STATEMENTS FOR CUSTOMER TABLE

❑ Returning all records from a table:

▪ Assuming we have the customer table listed in previous section, the select statement to display all records from the customer table is:

```
--SELECT STATEMENT #2 FOR CUSTOMER TABLE
select * from Customer;
```

• Results:

	CustomerID	FirstName	LastName	BirthDate	HouseStreetAddress	City	StateCode	ZipCode	Country	Phone	Email	DriverLicenseNumber	DriverLicenseExpDate	CustomerUserAccountID	CustomerType
1	1	Humain	Chapper	2000-10-10	Henry ST	Brooklyn	NY	11900	United States	9349087980	humain_chapper@hotmail.com	980009	2034-02-01	B107FEDE-CD5A-4736-B78B-0BD86EB353F7	Regular
2	3	Jacky	Decosta	2003-04-02	123 Whitney ST	Manhattan	NY	11001	United States	2123458765	jacky_decosta@hotmail.com	120987	2024-04-09	FF5757DA-CE3B-443C-99D3-340BD2716E12	Buyers
3	4	Maiden	Denver	2001-01-01	34 Washington ST	Canada	OH	12345	United States	9870090233	maiden_denver@hotmail.com	980011	2022-02-01	70BF1710-5B5E-4901-96C5-64114982EEF9	Regular
4	5	Dainton	Hons	2000-10-10	Henry ST	Brooklyn	NY	11900	United States	9349087980	dainton_hons@yahoo.com	912009	2024-07-11	D1862F47-AB98-4B4D-985F-BAA26668EEBB	Buyers
5	6	Pinto	Joneer	2003-01-12	Pensay ST	Queens	NY	11901	United States	9349087777	pinto_jonner@hotmail.com	123578	2022-02-01	D5826BE8-130E-4D98-B47C-FE7FC775A586	Regular

SELECT STATEMENTS FOR MULITPLE TABLES

❑ Returning records from three table:

▪ Assuming we have the Customer, CorporateCustomer and Company tables listed, the select statement to display records is:

```
--SELECT STATEMENT #3 FOR CORPORATECUSTOMER TABLE
select
a.CustomerID,a.FirstName,a.LastName,a.CustomerType,b.CompanyID,b.CompanyDailyRentalRate,c
.CompanyName,c.ContactPhone
from Customer a,CorporateCustomer b, Company c
where a.CustomerID = b.CustomerID
AND b.CompanyID  = c.CompanyID;
```

• Results:

Results | Messages

	CustomerID	FirstName	LastName	CustomerType	CompanyID	CompanyDailyRentalRate	CompanyName	ContactPhone
1	1	Humain	Chapper	Regular	23	80.00	EY	304989909
2	1	Humain	Chapper	Regular	23	80.00	EY	304989909
3	1	Humain	Chapper	Regular	23	80.00	EY	304989909
4	3	Jacky	Decosta	Buyers	24	800.00	Accenture	987098768
5	4	Maiden	Denver	Regular	25	23.50	Microsoft	987023456
6	5	Dainton	Hons	Buyers	26	50.30	Google	6067892345
7	6	Pinto	Joneer	Regular	27	33.88	Amazon	2345678912
8	1	Humain	Chapper	Regular	23	80.00	EY	304989909
		Jacky	Decosta	Buyers	24	800.00	Accenture	987098768
		Maiden	Denver	Regular	25	23.50	Microsoft	987023456
11	5	Dainton	Hons	Buyers	26	50.30	Google	6067892345
12	6	Pinto	Joneer	Regular	27	33.88	Amazon	2345678912
13	1	Humain	Chapper	Regular	23	80.00	EY	304989909
14	3	Jacky	Decosta	Buyers	24	800.00	Accenture	987098768
15	4	Maiden	Denver	Regular	25	23.50	Microsoft	987023456
16	5	Dainton	Hons	Buyers	26	50.30	Google	6067892345
17	6	Pinto	Joneer	Regular	27	33.88	Amazon	2345678912
18	1	Humain	Chapper	Regular	23	80.00	EY	304989909
19	3	Jacky	Decosta	Buyers	24	800.00	Accenture	987098768
20	4	Maiden	Denver	Regular	25	23.50	Microsoft	987023456
21	5	Dainton	Hons	Buyers	26	50.30	Google	6067892345
22	6	Pinto	Joneer	Regular	27	33.88	Amazon	2345678912
23	1	Humain	Chapper	Regular	23	80.00	EY	304989909
24	3	Jacky	Decosta	Buyers	24	800.00	Accenture	987098768
25	4	Maiden	Denver	Regular	25	23.50	Microsoft	987023456
26	5	Dainton	Hons	Buyers	26	50.30	Google	6067892345
27	6	Pinto	Joneer	Regular	27	33.88	Amazon	2345678912
28	1	Humain	Chapper	Regular	23	80.00	EY	304989909
29	3	Jacky	Decosta	Buyers	24	800.00	Accenture	987098768
30	4	Maiden	Denver	Regular	25	23.50	Microsoft	987023456
31	5	Dainton	Hons	Buyers	26	50.30	Google	6067892345
32	6	Pinto	Joneer	Regular	27	33.88	Amazon	2345678912
33	1	Humain	Chapper	Regular	23	80.00	EY	304989909
34	3	Jacky	Decosta	Buyers	24	800.00	Accenture	987098768
35	4	Maiden	Denver	Regular	25	23.50	Microsoft	987023456
36	5	Dainton	Hons	Buyers	26	50.30	Google	6067892345

orporateCustomer

UPDATE STATEMENTS

Update Example #1

❑ Updating all the columns of the record whose CustomerID = 1:

▪ First, we show the state of the customer record before updating by executing a SELECT statement:

```
select * from Customer where CustomerID = '1';
```

Results | Messages

	CustomerID	FirstName	LastName	BirthDate	HouseStreetAddress	City	StateCode	ZipCode	Country	Phone	Email	DriverLicenseNumber	DriverLicenseExpDate	CustomerUserAccountID	CustomerType
1	1	Humain	Chapper	2000-10-10	Henry ST	Brooklyn	NY	11900	United States	9349087980	humain_chapper@hotmail.com	980009	2034-02-01	B107FEDE-CD5A-4736-B78B-0BD86EB353F7	Regular

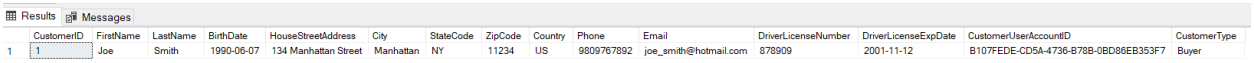
▪ We CREATE the UPDATE QUERY to update all columns of customer record whose CustomerID = 1:

```
--Update query to update all columns of one record
UPDATE Customer
SET FirstName='Joe',
       LastName='Smith',
       BirthDate='06/07/1990',
       HouseStreetAddress='134 Manhattan Street',
       City='Manhattan',
       StateCode='NY',
       ZipCode='11234',
       Country='US',
       Phone='9809767892',
       Email='joe_smith@hotmail.com',
       DriverLicenseNumber='878909',
       DriverLicenseExpDate='11/12/2001',
       CustomerType='Buyer'
```

```
WHERE CustomerID='1';
```

• EXECUTE the PREVIOUS SELECT STATEMENT to verify RESULTS and prove that record was UPDATED:

```
select * from Customer where CustomerID = '1';
```



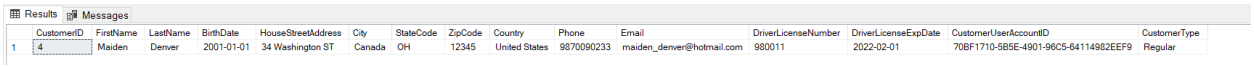
	CustomerID	FirstName	LastName	BirthDate	HouseStreetAddress	City	StateCode	ZipCode	Country	Phone	Email	DriverLicenseNumber	DriverLicenseExpDate	CustomerUserAccountID	CustomerType
1	1	Joe	Smith	1990-06-07	134 Manhattan Street	Manhattan	NY	11234	US	9809767892	joe_smith@hotmail.com	878909	2001-11-12	B107FEDE-CD5A-4736-B78B-0BD86EB353F7	Buyer

Update Example #2

❑ Updating only the LastName and CustomerType columns of the record whose CustomerID = 4:

▪ First, we show the state of the customer record before updating by executing a SELECT statement:

```
select * from Customer where CustomerID = '4';
```



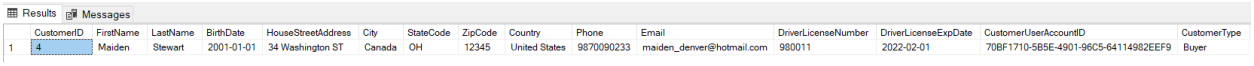
	CustomerID	FirstName	LastName	BirthDate	HouseStreetAddress	City	StateCode	ZipCode	Country	Phone	Email	DriverLicenseNumber	DriverLicenseExpDate	CustomerUserAccountID	CustomerType
1	4	Maiden	Denver	2001-01-01	34 Washington ST	Canada	OH	12345	United States	9870090233	maiden_denver@hotmail.com	980011	2022-02-01	70BF1710-5B5E-4901-96C5-64114982EEF9	Regular

▪ We CREATE the UPDATE QUERY to update columns of customer record whose CustomerID = 4:

```
--Update query to update columns of one record
UPDATE Customer
SET LastName='Stewart',
       CustomerType='Buyer'
WHERE CustomerID='4';
```

• EXECUTE the PREVIOUS SELECT STATEMENT to verify RESULTS and prove that record was UPDATED:

```
select * from Customer where CustomerID = '4';
```



	CustomerID	FirstName	LastName	BirthDate	HouseStreetAddress	City	StateCode	ZipCode	Country	Phone	Email	DriverLicenseNumber	DriverLicenseExpDate	CustomerUserAccountID	CustomerType
1	4	Maiden	Stewart	2001-01-01	34 Washington ST	Canada	OH	12345	United States	9870090233	maiden_denver@hotmail.com	980011	2022-02-01	70BF1710-5B5E-4901-96C5-64114982EEF9	Buyer

DELETE STATEMENTS

DELETE Example #1

❑ We wish to delete the record of the CreditCard with OwnerName = ‘Shital’;

▪ First, we show the state of the CreditCard Table before deleting by executing a SELECT statement:

```
SELECT * FROM CreditCard;
```

	CardNumber	OwnerName	MerchantName	ExpDate	HouseStreetAddress	City	StateCode	ZipCode	Country	CreditLimit	ActivationStatus
1	111111111	Shital	EXPLUS	1002-09-09	11 Caryy ST	Woodhaven	NY	33453	Nepal	25000.00	True
2	111111112	Garry	MARTIN	2000-04-09	City ST	Bronx	SF	33422	Japan	2000.00	True
3	111111113	Darry	BONNY	2002-01-02	Molly ST	Queens	NY	10003	Singapore	5000.00	False
4	111111114	Prince	RENTAL	2010-03-07	Dog ST	Staten Island	NY	90003	Australia	5000.00	True
5	111111115	Joey	DOGJE	1999-07-07	11 Cat ST	Woodhaven	NY	33453	Italy	1000.00	False

▪ We CREATE the DELETE QUERY to delete the credit card record whose OwnerName = 'Shital';

```
--DELETE QUERY
DELETE
FROM CreditCard
WHERE OwnerName = 'Shital';
```

• EXECUTE the PREVIOUS SELECT STATEMENT to verify RESULTS and prove that record was DELETED:

	CardNumber	OwnerName	MerchantName	ExpDate	HouseStreetAddress	City	StateCode	ZipCode	Country	CreditLimit	ActivationStatus
1	111111112	Garry	MARTIN	2000-04-09	City ST	Bronx	SF	33422	Japan	2000.00	True
2	111111113	Darry	BONNY	2002-01-02	Molly ST	Queens	NY	10003	Singapore	5000.00	False
3	111111114	Prince	RENTAL	2010-03-07	Dog ST	Staten Island	NY	90003	Australia	5000.00	True
4	111111115	Joey	DOGJE	1999-07-07	11 Cat ST	Woodhaven	NY	33453	Italy	1000.00	False

▪ As you can see, the records whose OwnerName = 'Shital' was deleted.

DELETE Example #2

❑ We wish to delete the record of the Customer with CustomerID = '1';

▪ First, we show the state of the Customer Table before deleting by executing a SELECT statement:

```
SELECT * FROM Customer;
```

	CustomerID	FirstName	LastName	BirthDate	HouseStreetAddress	City	StateCode	ZipCode	Country	Phone	Email	DriverLicenseNumber	DriverLicenseExpDate	CustomerUserAccountID	CustomerType
1	1	Humain	Chapper	2000-10-10	Henry ST	Brooklyn	NY	11900	United States	9349087980	humain_chapper@hotmail.com	980009	2034-02-01	B107FEDE-CD5A-4736-B78B-0BD86EB353F7	Regular
2	3	Jacky	Decosta	2003-04-02	123 Whitney ST	Manhattan	NY	11001	United States	2123458765	jacky_decosta@hotmail.com	120987	2024-04-09	FF5757DA-CE3B-443C-99D3-340BD2716E12	Buyers
3	4	Maiden	Denver	2001-01-01	34 Washington ST	Canada	OH	12345	United States	9870090233	maiden_denver@hotmail.com	980011	2022-02-01	70BF1710-5B5E-4901-96C5-64114982EEF9	Regular
4	5	Dainton	Hons	2000-10-10	Henry ST	Brooklyn	NY	11900	United States	9349087980	dainton_hons@yahoo.com	912009	2024-07-11	D1862F47-AB98-4B4D-985F-BAA26668EEBB	Buyers
5	6	Pinto	Joneer	2003-01-12	Pensy ST	Queens	NY	11901	United States	9349087777	pinto_jonner@hotmail.com	123578	2022-02-01	D5826BE8-130E-4D98-B47C-FE7FC775A586	Regular

▪ We CREATE the DELETE QUERY to delete the customer record whose CustomerID = '1';

```
--DELETE QUERY 2
DELETE
FROM Customer
WHERE CustomerID = '1';
```

• EXECUTE the PREVIOUS SELECT STATEMENT to verify RESULTS and prove that record was DELETED:

	CustomerID	FirstName	LastName	BirthDate	HouseStreetAddress	City	StateCode	ZipCode	Country	Phone	Email	DriverLicenseNumber	DriverLicenseExpDate	CustomerUserAccountID	CustomerType
1	3	Jacky	Decosta	2003-04-02	123 Whitney ST	Manhattan	NY	11001	United States	2123458765	jacky_decosta@hotmail.com	120987	2024-04-09	FF5757DA-CE3B-443C-99D3-340BD2716E12	Buyers
2	4	Maiden	Stewart	2001-01-01	34 Washington ST	Canada	OH	12345	United States	9870090233	maiden_denver@hotmail.com	980011	2022-02-01	70BF1710-5B5E-4901-96C5-64114982EEF9	Buyer
3	5	Dainton	Hons	2000-10-10	Henry ST	Brooklyn	NY	11900	United States	9349087980	dainton_hons@yahoo.com	912009	2024-07-11	D1862F47-AB98-4B4D-985F-BAA26668EEBB	Buyers
4	6	Pinto	Joneer	2003-01-12	Pensy ST	Queens	NY	11901	United States	9349087777	pinto_jonner@hotmail.com	123578	2022-02-01	D5826BE8-130E-4D98-B47C-FE7FC775A586	Regular
5	182	Humain	Chapper	2000-10-10	Henry ST	Brooklyn	NY	11900	United States	9349087980	humain_chapper@hotmail.c...	980009	2034-02-01	B107FEDE-CD5A-4736-B78B-0BD86EB353...	Regular

▪ As you can see, the records whose CustomerID = '1' was deleted.

CONCLUSION

Finally, the project contains all the requirements including PLANNING, ANALYSIS, DESIGN, DEVELOPMENT & IMPLEMENTATION, DEVELOPMENT & IMPLEMENTATION UNIT TESTING.