**JWheels Dealership project**

**Made by: Jesus Nunez**



**December 2, 2022**

**CS 4318: Database Management Systems**

**Professor: Ling Xu**

**University of Houston - Downtown**

**Table of Contents**

[**Introduction** 4](#_Toc119881724)

[**Abstract:** 4](#_Toc119881725)

[**Mission statement:** 4](#_Toc119881726)

[**Mission Objectives:** 4](#_Toc119881727)

[**Sales Department Use cases:** 6](#_Toc119881728)

[**Service Department Use Cases** 10](#_Toc119881729)

[**Full Use Case Diagram:** 13](#_Toc119881730)

[**E/R diagram:** 13](#_Toc119881731)

[**Schemas** 14](#_Toc119881732)

[**Sales Department** 14](#_Toc119881733)

[**Service Department** 15](#_Toc119881734)

[**Use case Implementation (SQL Statements)** 16](#_Toc119881735)

[**Sales Department: Insert, Update, Delete commands** 17](#_Toc119881736)

[**Service Department: insert, update, and delete commands** 19](#_Toc119881737)

[**Normalization: Sales Department** 21](#_Toc119881738)

[**Customer entity data** 21](#_Toc119881739)

[**Sales Employees entity** 26](#_Toc119881740)

[**Dealership Vehicle entity** 30](#_Toc119881741)

[**Sales entity** 33](#_Toc119881742)

[**Normalization Service Department** 35](#_Toc119881743)

[**Service Customer Entity** 35](#_Toc119881744)

[**Service Employees entity** 39](#_Toc119881745)

[**Service Sales entity** 42](#_Toc119881746)

[**Normalization Dealership Entity** 45](#_Toc119881747)

[**Test plan and records** 47](#_Toc119881748)

[**Dealership Entity: Insert, Delete, Update, and Aggregate commands** 47](#_Toc119881749)

[**Sales Department: Insert, Update, Delete commands** 50](#_Toc119881750)

[**Customer Entity** 50](#_Toc119881751)

[**Sales Employees Entity** 53](#_Toc119881752)

[**Sales Entity** 56](#_Toc119881753)

[**Dealership Vehicle Entity** 59](#_Toc119881754)

[**Join Commands Sales Department** 62](#_Toc119881755)

[**Aggregate Commands Sales Department** 66](#_Toc119881756)

[**Service Department: Insert, Update, Delete commands** 68](#_Toc119881757)

[**Service Customer Entity** 68](#_Toc119881758)

[**Service Employees Entity** 71](#_Toc119881759)

[**Service Invoice Entity** 74](#_Toc119881760)

[**Join Commands Service Department** 77](#_Toc119881761)

[**Aggregate Commands Service Department** 80](#_Toc119881762)

[**Conclusion** 82](#_Toc119881763)

[**Resources** 83](#_Toc119881764)

[**Video Links** 84](#_Toc119881765)

# **Introduction**

## **Abstract:**

The database application for this project will be based on simulating a dealership. The dealership database will contain two departments. The first department will be the sales department. The sales department will contain customer data, employee data, dealership data, vehicle sales data, and customer vehicle data if they want to do a trade in. The second department will be known as the service department. The service department will contain service type data, customer vehicle data, customer data, parts data, and service employee data. The sales department and service department will have different data that will be connected to a dealership.

## **Mission statement:**

Our mission is to make an easy-access database that stores information securely and to ensure proper customer service. We strive to grow as a business and to become one of the most reliable dealerships in the world.

## **Mission Objectives:**

- To maintain (enter, update, and delete) data on sales customer.

- To maintain (enter, update, and delete) data on sales employee.

- To maintain (enter, update, and delete) data on sales vehicles.

- To maintain (enter, update, and delete) data on dealerships.

- To maintain (enter, update, and delete) data on sales invoice.

- To maintain (enter, update, and delete) data on service employee.

- To maintain (enter, update, and delete) data on service customer.

- To maintain (enter, update, and delete) data on service invoice.

- To perform searches on customers.

- To perform searches on sales employees.

- To perform searches on sales vehicles.

- To perform searches on dealerships.

- To perform searches on sales invoice.

- To perform searches on service employee.

- To perform searches on service customer.

- To perform searches on service invoice.

- To track the status of dealership sales vehicles.

- To track the status of customer payments.

- To track the status of invoice.

- To track the status of parts.

- To report on sales customer.

- To report on sales employee.

- To report on dealership.

- To report on service customer.

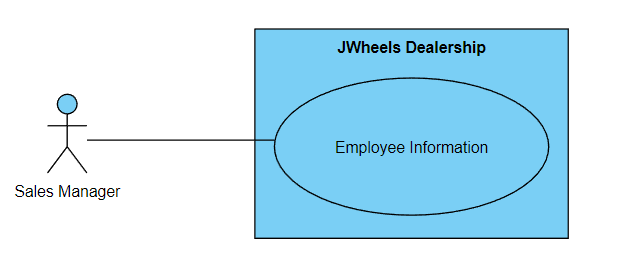
- To report on service employee.

- To report on service parts.

- To report on service invoice.

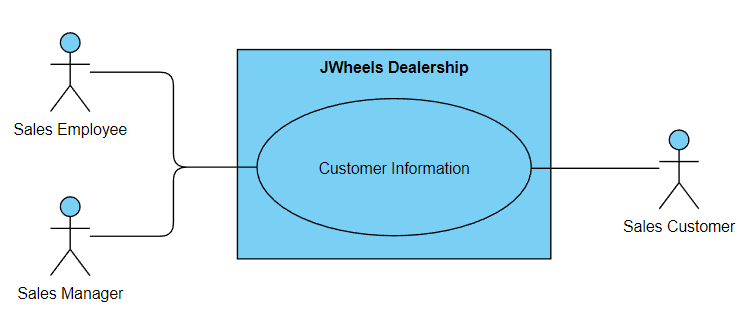
# **Sales Department Use cases:**

Employee Information

****

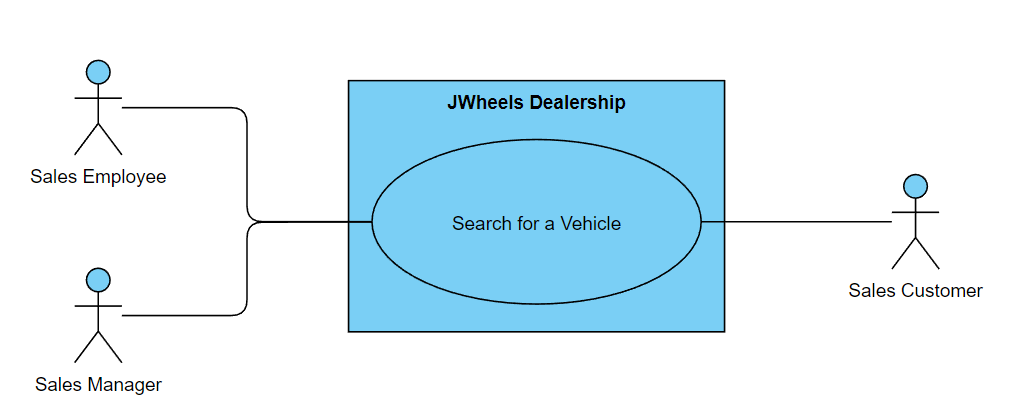
**Brief Description:** Dealership sales managers can add, edit, or delete employee information. Generally, managers will have more privileges when using the database.

Customer Information

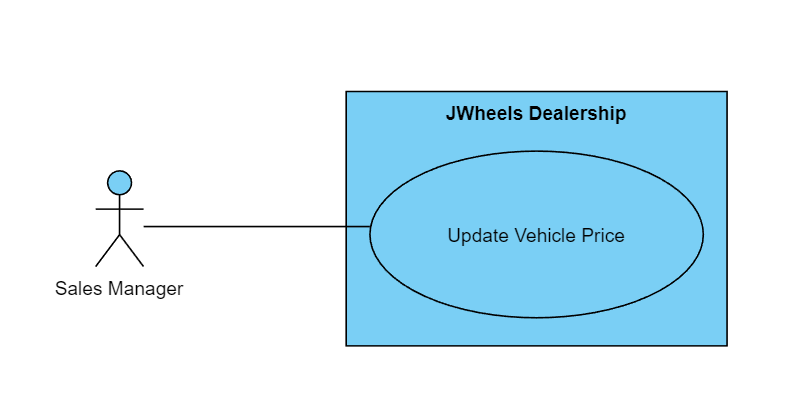


**Brief Description:** Dealerships normally get customers information when making sales. Therefore, sales employees and sales managers will have the privilege to add, edit, or delete customers information.

Search for a Vehicle

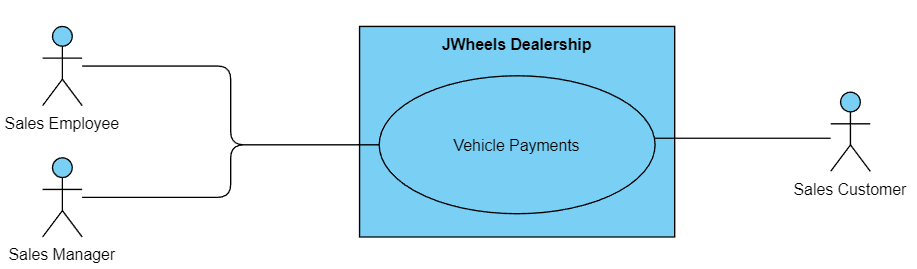
****

**Brief Description:** Managers or employees can assist a customer to find a new or used vehicle. In this case, the database will also contain the number of vehicles in stock.

Update Vehicle Price

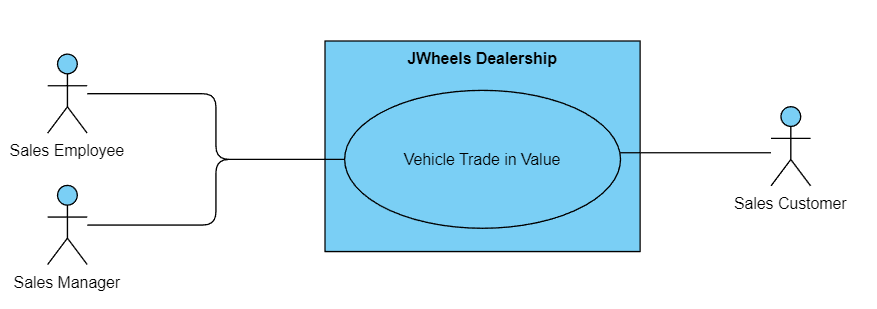
**Brief Description:** A manager only has the privilege to changing car prices. Employees must communicate with managers about any customer price change request such as discounts or other vehicle offers.

Vehicle Payments

****

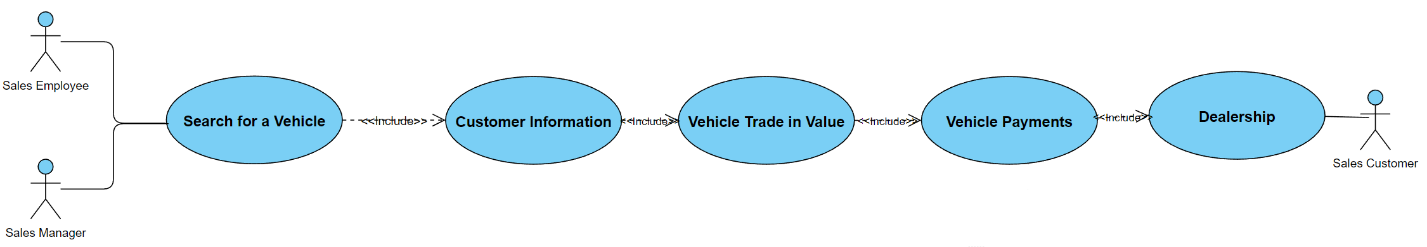
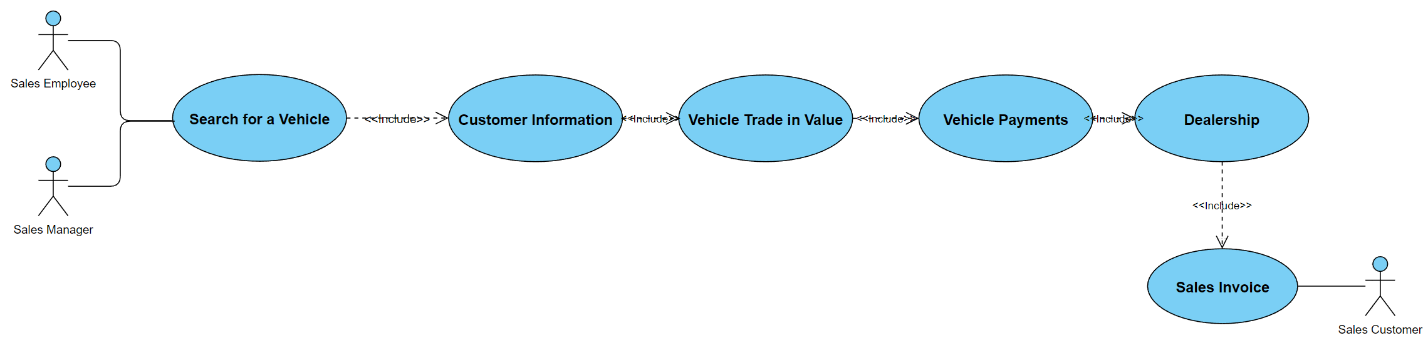
**Brief Description:** Both sales manager and employee can set a customer up for vehicle payments.

Vehicle Trade in Value

****

**Brief Description:** Both sales manager and employee can set a customer up for a vehicle trade-in value for customers that want to trade in their vehicles.

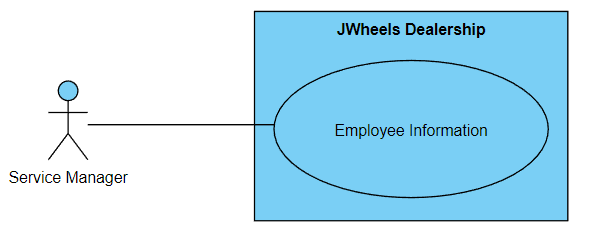
Vehicle Purchase Process

****

**Brief Description:** Either the employee or manager who is assisting a customer in making vehicle purchases must search for the selected customer vehicle. Customer information will also contain credit information to see how much annual percentage rate (APR) a customer will pay. The “Vehicle Trade in Value” is optional, if a customer does not want a vehicle to be traded in, then it could just be placed as zero. Vehicle payments will contain information about customers payments for their chosen purchased vehicle. The dealership branch and date of purchase will be included in the invoice as proof.

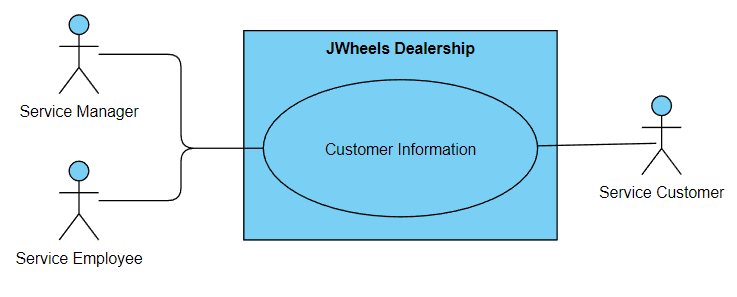
# **Service Department Use Cases**

Employee Information

****

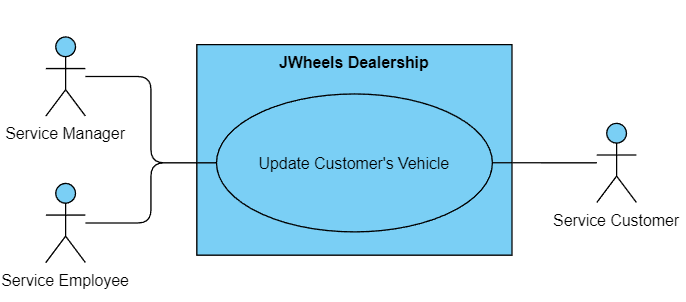
**Brief Description:** Service Managers will have the privilege of adding, editing, or deleting an employee’s information.

Customer Information

****

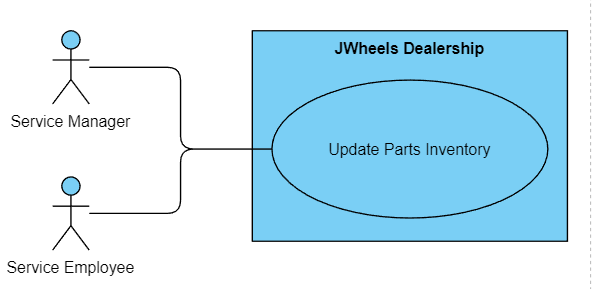
**Brief Description:** Both service manager and employee will have the privileges in adding, editing, or deleting customer data.

Update Customer’s Vehicle

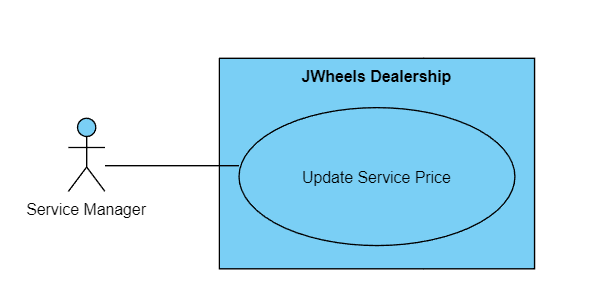


**Brief Description:** Both service manager and employee will have the privileges in adding, editing, or deleting a customer’s vehicle and its information such as date arrived, VIN, make, model, year, and vehicle descriptions for either service or vehicle issue.

Update Parts Inventory

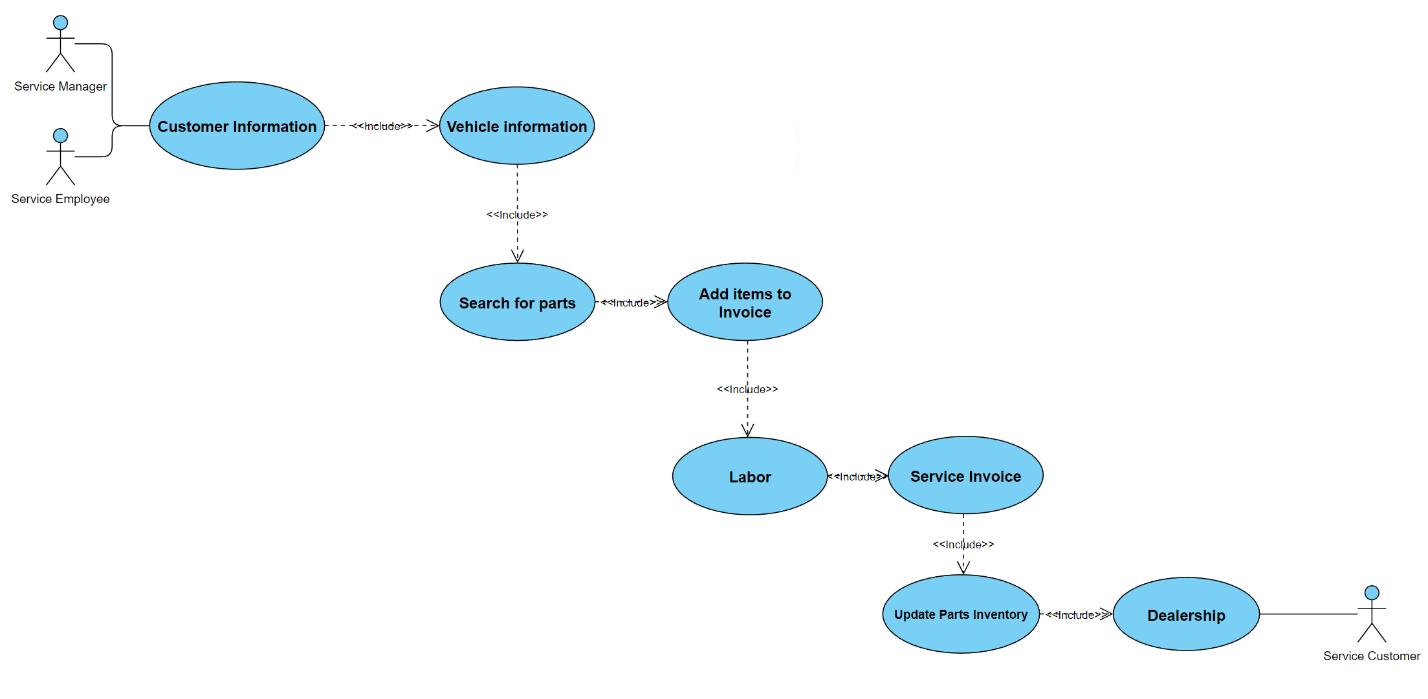


**Brief Description:** Both service employees andmanagers can add, edit, or delete parts inventory. There will be data that also includes the number of parts in stock.

****Update Service Price

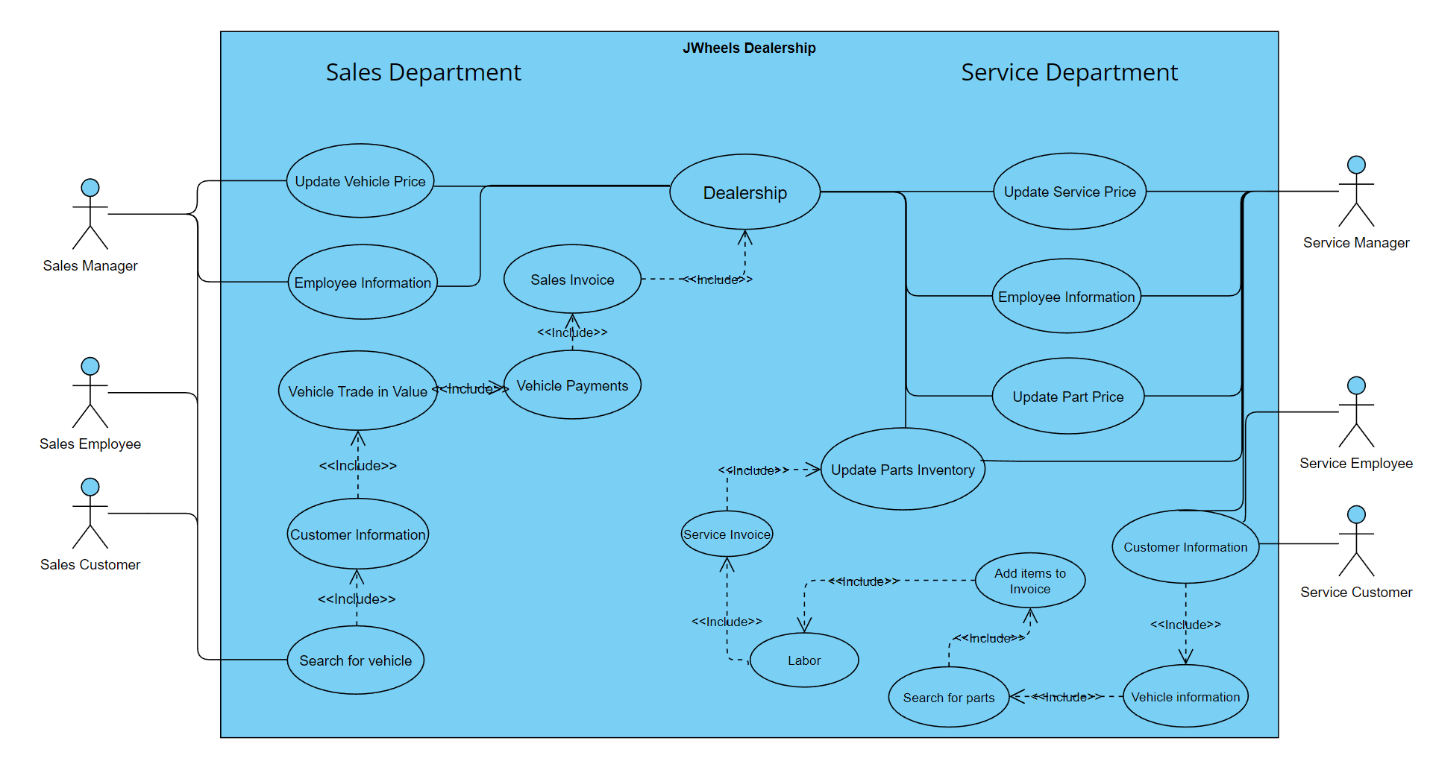
**Brief Description:** Service Manager has the privilege to add, edit, or delete service price.

Vehicle Servicing process

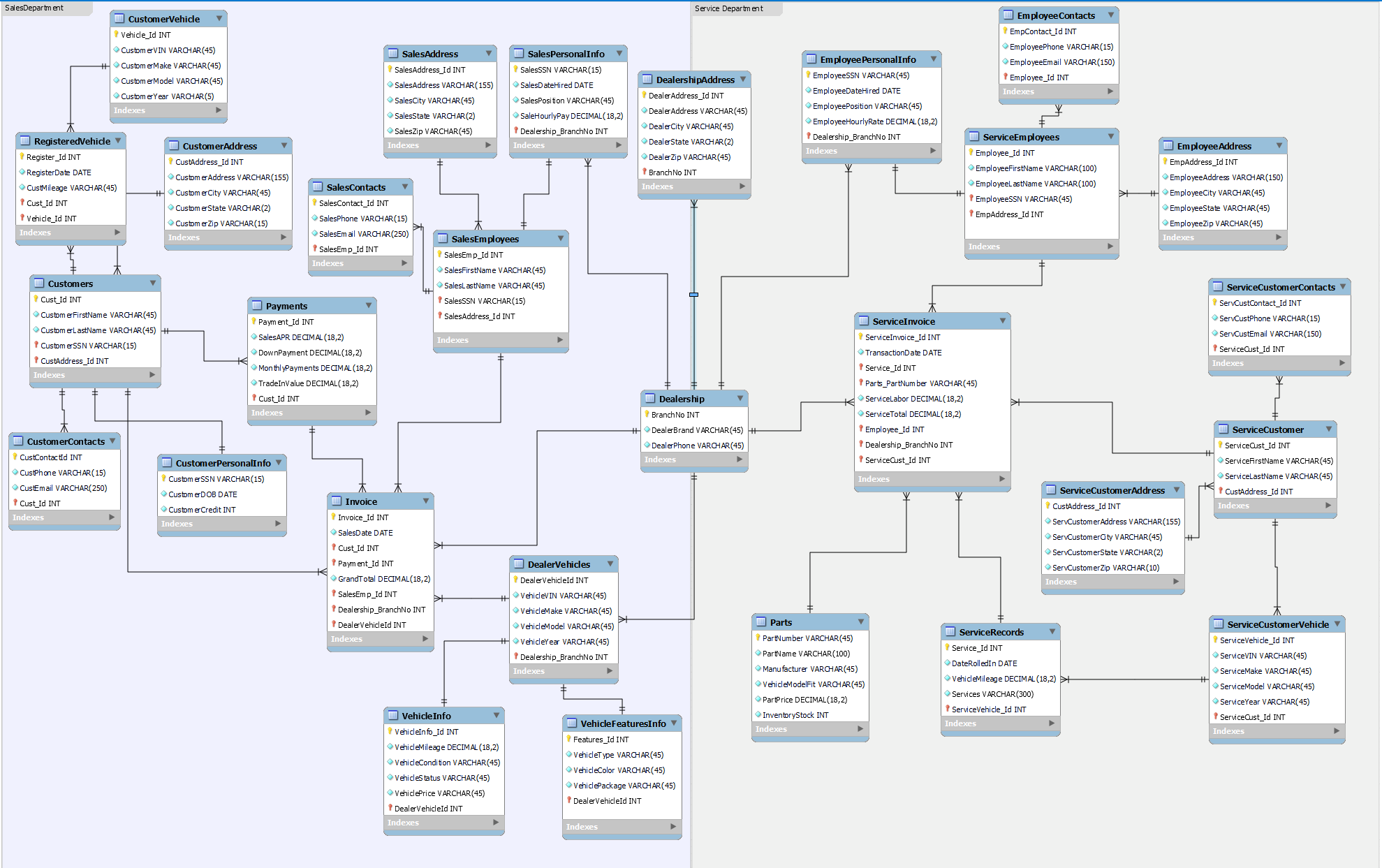


**Brief Description:** Either a service manager oran employee can assist a customer. When a customer rolls their vehicle in the shop, employees will search for customers information. Then they will get customer’s vehicle info. When a technician sees that a part needs to be replaced, an employee or manager will search for parts.Then the parts selected will be added to a bill along with technician labor and print out an invoice along with the dealership branch number for the customer. An employee or manager can update parts inventory for every purchases. The database will add customer’s service history invoice.

# **Full Use Case Diagram:**

****

# **E/R diagram:**

****

# **Schemas**

**Example: RELATION**( Primary key → Attribute, **Foreign key**)

* **Dealership**(BranchNo → DealerBrand, DealerPhone)
* **DealershipAddress** (DealerAddress\_Id → DealerAddress, DealerCity, DealerState**,** DealerZip, **BranchNo**)

## **Sales Department**

* **Customers**(Cust\_Id→ CustomerFirstName, CustomerLastName, **CustomerSSN, CustAddress\_Id**)
* **CustomerPersonalInfo**(CustomerSSN→ CustomerDOB, CustomerCredit)
* **CustomerAddress**(CustAddress\_Id → CustomerAddress, CustomerCity, CustomerState, CustomerZip)
* **CustomerContacts**(CustContactId → CustPhone, CustEmail, **Cust\_Id**)
* **RegisteredVehicle**(Register\_Id → RegisterDate, CustMileage, **Cust\_Id**, **Vehicle\_Id**)
* **CustomerVehicle** (Vehicle\_Id → CustomerVIN, CustomerMake, CustomerModel, CustomerYear)
* **SalesEmployees**(SalesEmp\_Id → SalesFirstName, SalesLastName, **SalesSSN**, **SalesAddress\_Id**)
* **SalesPersonalInfo**(SalesSSN → SalesDateHired, SalesPosition, SaleHourlyPay, **Dealership\_BranchNo**)
* **SalesAddress**(SalesAddress\_Id → SalesAddress, SalesCity, SalesState, SalesZip)
* **SalesContacts**(SalesContact\_Id → SalesPhone, SalesEmail, **SalesEmp\_Id**)
* **DealerVehicles**(DealerVehicleId → VehicleVIN, VehicleMake, VehicleModel, VehicleYear, **Dealership\_BranchNo**)
* **VehicleInfo** (VehicleInfo\_Id→ VehicleMileage, VehicleCondition, VehicleStatus, VehiclePrice, **DealerVehicleId**)
* **VehicleFeaturesInfo**(Features\_Id → VehicleType, VehicleColor, VehiclePackage, **DealerVehicleId**)
* **Payments**(Payment\_Id → SalesAPR, DownPayment, MonthlyPayments, TradeInValue, **Cust\_Id**)
* **Invoice**(Invoice\_Id → SalesDate, **Cust\_Id**, **Payment\_Id**, GrandTotal, **SalesEmp\_Id**, **Dealership\_BranchNo**, **DealerVehicleId**)

## **Service Department**

* **ServiceCustomer**(ServiceCust\_Id → ServiceFirstName, ServiceLastName, **CustAddress\_Id**)
* **ServiceCustomerContacts**(ServCustContact\_Id → ServCustPhone, ServCustEmail, **ServiceCust\_Id**)
* **ServiceCustomerVehicle**(ServiceVehicle\_Id→ ServiceVIN, ServiceMake, ServiceModel, ServiceYear, **ServiceCust\_Id**)
* **ServiceCustomerAddress**(CustAddress\_Id → ServCustomerAddress, ServCustomerCity, ServCustomerState, ServCustomerZip)
* **ServiceEmployees**(Employee\_Id → EmployeeFirstName, EmployeeLastName, **EmployeeSSN**, **EmpAddress\_Id**)
* **EmployeePersonalInfo**(EmployeeSSN → EmployeeDateHired, EmployeePosition, EmployeeHourlyRate, **Dealership\_BranchNo**)
* **EmployeeContacts**(EmpContact\_Id → EmployeePhone, EmployeeEmail, **Employee\_Id**)
* **EmployeeAddress**(EmpAddress\_Id → EmployeeAddress, EmployeeCity, EmployeeState, EmployeeZip)
* **ServiceRecords**(Service\_Id→ DateRolledIn, VehicleMileage, Services, **ServiceVehicle\_Id**)
* **Parts**(PartNumber → PartName, Manufacturer, VehicleModelFit, PartPrice, InventoryStock)
* **ServiceInvoice**(ServiceInvoice\_Id → TransactionDate, **Service\_Id**, **Parts\_PartNumber**, ServiceLabor, ServiceTotal, **Employee\_Id**, **Dealership\_BranchNo**, **ServiceCust\_Id**)

# **Use case Implementation (SQL Statements)**

**Dealership Entity: Insert, Delete, Update commands**

**Insert command for “Dealership” table**

select \* from dealership;

insert into dealership

(BranchNo, DealerBrand, DealerPhone)

value

('787', 'Hyundai', '(622)-624-6666');

**Delete command for “Dealership” table**

select \* from dealership;

delete from dealership

where BranchNo = 787;

**Update Command for “Dealership” table**

select \* from dealership;

update dealership

set DealerBrand= 'TOYOTA'

where BranchNo = 126;

## **Sales Department: Insert, Update, Delete commands**

**Customer Entity: Insert, Delete, Update commands**

**Insert command for “Customers” table**

select \* from customers; insert into customers

(Cust\_Id, CustomerFirstName, CustomerLastName, CustomerSSN, CustAddress\_Id)

value ('20', 'Marcus', 'Bradley', '20XXXXXXX', '20');

**Delete command for “Customers” table**

select \* from customers;

delete from customers

where Cust\_id = 20;

**Update command for “Customers” table**

select \* from customers;

update customers

set CustomerFirstName = 'Jamie'

where Cust\_Id = 19;

**Sales Employees Entity: insert, update, and delete commands**

**Insert command for “SalesEmployees” table**

select \* from salesemployees;

insert into salesemployees

(SalesEmp\_Id, SalesFirstName, SalesLastName, SalesSSN, SalesAddress\_Id)

value ('25', 'Kalil', 'Brown', '56XXXXXXX', '25')

**Delete command for “SalesEmployees” table**

select \* from salesemployees;

delete from salesemployees

where SalesEmp\_Id = 25;

**Update command for “SalesEmployees” table**

select \* from salesemployees;

update salesemployees set SalesFirstName = 'Lu', SalesLastName = 'Xiao' where SalesEmp\_Id = 13;

**Sales Entity: insert, update, and delete commands**

**Insert command for “Invoice” table**

select \* from invoice;

insert into invoice

(Invoice\_Id, SalesDate, Cust\_Id, Payment\_Id, GrandTotal, SalesEmp\_Id, Dealership\_BranchNo, DealerVehicleId)

value ('20', '2022-03-15', '20', '20', '7538', '24', '847', '59');

**Delete command for “Invoice” table**

select \* from invoice;

delete from invoice

where Invoice\_Id = 20;

**Update command for “Invoice” table**

select \* from invoice;

update invoice

set SalesDate = '2022-04-29'

where Invoice\_Id = 19;

**Dealership Vehicle Entity: insert, update, and delete commands**

**Insert command for “DealerVehicles” table**

select \* from dealervehicles;

insert into dealervehicles

(DealerVehicleId, VehicleVIN, VehicleMake, VehicleModel, VehicleYear)

value ('61', '2FMPK4J96NBA89376', 'Ford', 'Edge', '2022');

**Delete command for “DealerVehicles” table**

select \* from dealervehicles;

delete from dealervehicles

where DealerVehicleId = 61;

**Update command for “DealerVehicles” table**

select \* from dealervehicles;

update dealervehicles

set VehicleYear = '2021' where DealerVehicleId = 60;

## **Service Department: insert, update, and delete commands**

**Service Customer Entity: insert, update, and delete commands**

**Insert command for “ServiceCustomer” table**

select \* from servicecustomer;

insert into servicecustomer

(ServiceCust\_Id, ServiceFirstName, ServiceLastName, CustAddress\_Id)

values ('19', 'Yesenia', 'Campos', '19');

**Delete command for “ServiceCustomer” table**

select \* from ServiceCustomer;

delete from ServiceCustomer

where SalesEmp\_Id = 25;

**Update command for “ServiceCustomer” table**

select \* from servicecustomer;

update servicecustomer

set ServiceLastName = 'Vasquez'

where ServiceCust\_Id = 11;

**Service Employees Entity: insert, update, and delete commands**

**Insert command for “ServiceEmployees” table**

select \* from serviceemployees;

insert into serviceemployees

(Employee\_Id, EmployeeFirstName, EmployeeLastName, EmployeeSSN, EmpAddress\_Id)

values ('25', 'Kenya', 'Esparza', '80XXXXXXX', '25')

**Delete command for “ServiceEmployees” table**

select \* from serviceemployees;

delete from serviceemployees

where Employee\_Id = 25;

**Update command for “ServiceEmployees” table**

select \* from serviceemployees;

update serviceemployees

set EmployeeFirstName = 'Yin', EmployeeLastName = 'Ten' where Employee\_Id = 8;

**Service Invoice Entity: insert, update, and delete commands**

**Insert command “ServiceInvoice” table**

select \* from serviceinvoice;

insert into serviceinvoice

(ServiceInvoice\_Id, TransactionDate, Service\_Id, Parts\_PartNumber, ServiceLabor, ServiceTotal, Employee\_Id, Dealership\_BranchNo, ServiceCust\_Id)

value ('20', '2022-12-02', '20','N/A', '25.50', '25.50', '25', '515', '19');

**Delete command “ServiceInvoice” table**

select \* from serviceemployees;

delete from serviceemployees

where Employee\_Id = 20;

**Update command “ServiceInvoice” table**

select \* from serviceinvoice;

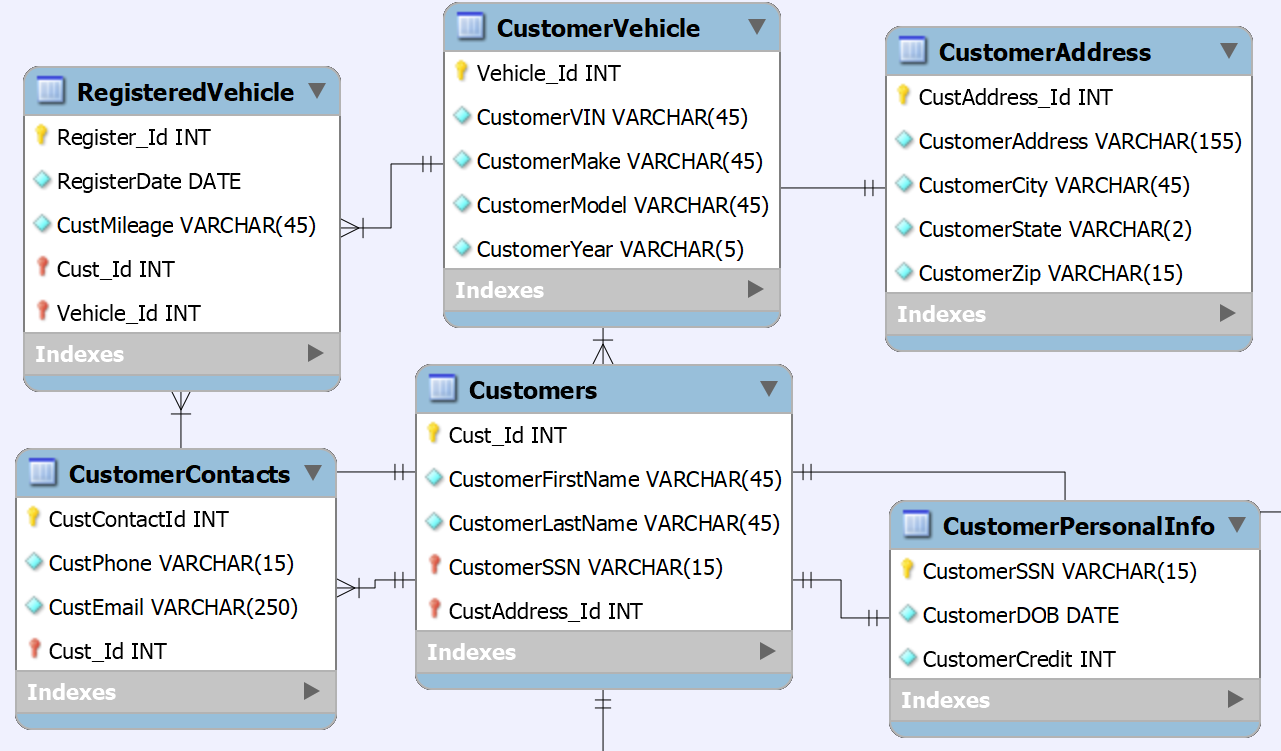
update serviceinvoice

set TransactionDate = '2022-11-12'

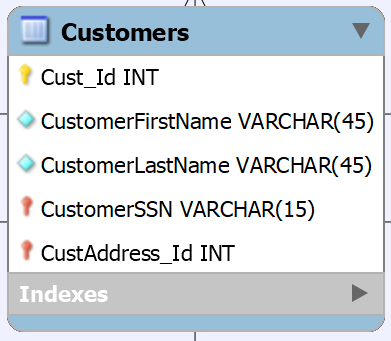
where ServiceInvoice\_Id = 19;

# **Normalization: Sales Department**

## **Customer entity data**

****

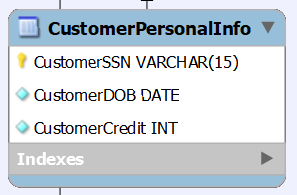
The “Customers” entity data contains six tables are dedicated to store customer information. The entity includes “Customers”, “CustomerPersonalInfo”, “CustomerAddress”, “CustomerContacts”, “RegisteredVehicle”, and “CustomerVehicle” tables.



The “Customers” table contains basic customer information. The table contains a primary key that holds customer’s ID number (Cust\_Id), customer’s first name (CustomerFirstName), customer’s last name (CustomerLastName), and two foreign keys that store customers social security number (CustomerSSN) and customer address (CustAddress\_Id). The data is does not contain any partial or transitive dependencies. The foreign keys only refer a primary key from another table, which in this case the “CustomerSSN” foreign key refers the “CustomerPersonalInfo” table, and “CustAddress” foreign key refers the “CustomerAddress” table. The “Customers” table is in a one-to-one relationship with “CustomerPersonalInfo” table because a customer can only have one personal information. The “Customers” table is in a many-to-one relationship with “CustomerAddress” table because many customers can live in an address. The “Customers” table is in a one-to-many relationship with “RegisterVehicle” table because a customer can have multiple registered vehicles. The “Customers” table is in a one-to-many relationship with “Payments” table because a customer can have multiple payments. The attributes are functionally dependent as there are no non-prime (non-key) attributes in the table.

* Cust\_Id → CustomerFirstName, CustomerLastName, CustomerSSN, CustAddress

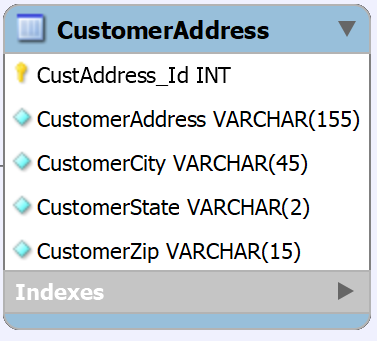
Therefore, “Customers” table is set in third normal form.



The “CustomerPersonalInfo” table contains a customer’s personal information. As stated before, “CustomerPersonalInfo” table is in a one-to-one relationship with “Customers” table. This table is in third normal form because there are no transitive and partial dependencies in the table. The table does have the proper functional dependencies that depend on a customer’s personal information such as the primary key that holds customer’s social security number (CustomerSSN) , customers date of birth (CustomerDOB), and customer’s credit score (CustomerCredit). In short:

* CustomerSSN → CustomerDOB, CustomerCredit

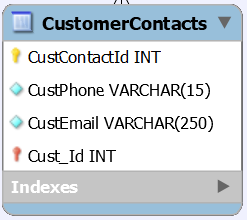
The table does not contain any non-prime attributes and transitive dependencies. Therefore, “CustomerPersonalInfo” table is in third normal form.



The “CustomerAddress” table contains a customer’s address information. As shown in the table above, there are five attributes that are functionally dependent of each other. The table includes a primary key that holds a customer address number ID (CustAddress\_Id), customer’s address (CustomerAddress), customer’s city (CustomerCity), customer’s state (CustomerState), and customer’s zip code (CustomerZip). The “CustomerAddress” table is in a one-to-many relationship with “Customers” table since many an address could belong to multiple customers. The “CustomerAddress” table does not contain any non-prime attributes or transitive dependencies as every attribute is functionally dependent:

* CustAddress\_Id → CustomerAddress, CustomerCity, CustomerState, CustomerZip

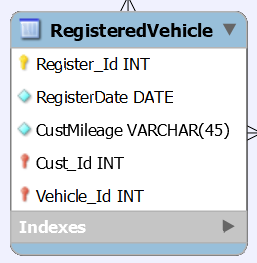
Therefore, the “CustomerAddress” table is in third normal form.



The “CustomerContacts” table contains customer’s contact information. The table contains three attributes and one foreign key that refers to “Customers” table. The table includes a primary key that holds the customer’s contact ID number (CustContactId), customer’s phone number (CustPhone), customer’s email (CustEmail), and the foreign key that holds the customer’s ID number (Cust\_Id). The “CustomerContacts” table is in a many-to-one relationship with “Customers” table because a customer can have multiple contacts or use the same contact information. The attributes in the “CustomerContacts” table are functionally dependent and no non-prime attributes exist in the table. Also, the table does not contain any transitive dependencies.

* CustContactId → CustPhone, CustEmail, Cust\_Id

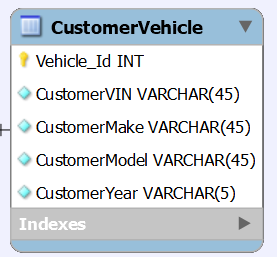
Therefore, “CustomerContacts” table is in third normal form.



The “RegisteredVehicle” table is also part of “Customers” entity because it contains information for customers that want to trade in their vehicles for a new one. The table contains three attributes and two foreign keys. The table includes a primary key that holds the registration ID number (Register\_Id), date of registration (RegisterDate), the vehicle mileage (CustMileage), and two foreign keys that are known as “Cust\_Id” and “Vehicle\_Id”. The foreign key “Cust\_Id” refers to the “Customers” table and “Vehicle\_Id” refers to the “CustomerVehicle” table (which will be discussed next). The “RegisteredVehicle” table is in a many-to-one relationship with “Customers” table because a customer can have many registered vehicles. Also, the “RegisteredVehicle” table is in a many-to-one relationship with “CustomerVehicle” table because many registrations can contain a vehicle information. The attributes in the “RegisteredVehicle” table are functionally dependent since the table does not contain any non-prime attributes or transitive dependencies:

* Register\_Id → RegisterDate, CustMileage, Cust\_Id, Vehicle\_Id

**Note:** CustMileage is functionally dependent with the attributes in this table because a vehicle will never have the same mileage from the day that it was registered. Hence, it is important that mileage is recorded properly based on date because it plays a determination of a vehicle’s value when it is traded in. Therefore, the “RegisteredVehicle” table is in third normal form.

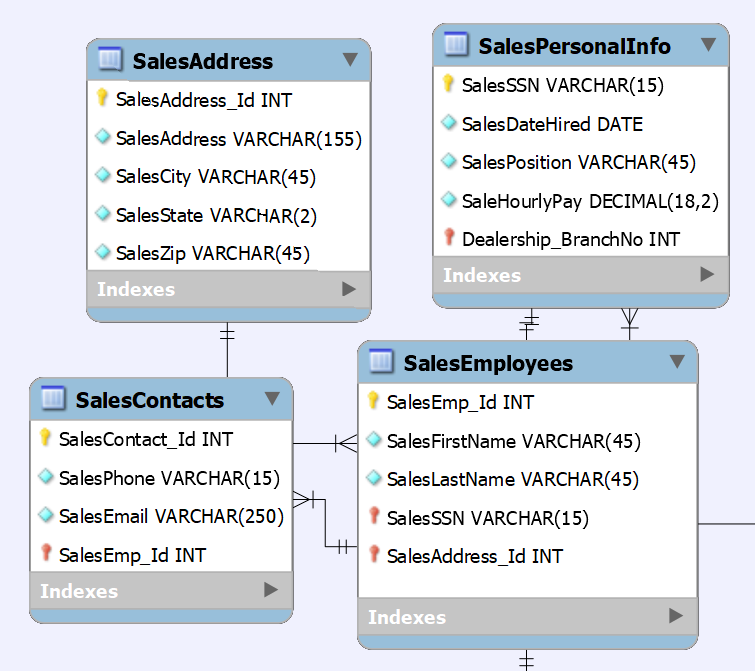


The “CustomerVehicle” table contains a customer’s vehicle information. There are five attributes that depend on each other. The table includes a primary key that holds the vehicle number ID (Vehicle\_ID), customer’s vehicle identification number (CustomerVIN), customer’s make (CustomerMake), customer’s model (CustomerModel), and customer’s year (CustomerYear). As stated before, the “CustomerVehicle” table is in a one-to-many relationship with “RegisteredVehicle” table because a vehicle can have multiple registrations. The attributes in the “CustomerVehicle” table are functionally dependent since the table does not contain any non-prime attributes or transitive dependencies:

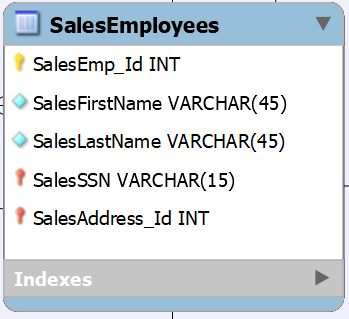
* Vehicle\_Id → CustomerVIN, CustomerMake, CustomerModel, CustomerYear

Therefore, “CustomerVehicle” table in in third normal form.

## **Sales Employees entity**

****

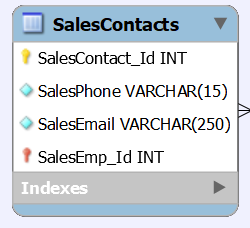
The “Sales Employees” entity contains four dedicated tables that store sales employees information. The entity includes “SalesEmployees”, “SalesContacts”, “SalesAddress”, and “SalesPersonalInfo” tables. The “Sales Entity” tables are in third normal form since none of the tables any partial or transitive dependencies. Here are the reasons why:



The “SalesEmployees” table contains the names of employees. As shown above, the table contains three attributes and two foreign keys. The table contains a primary key that holds sales employee’s ID number (SalesEmp\_Id), sales employee’s first name (SalesFirstName), sales employee’s last name (SalesLastName), and two foreign keys that includes sales employees’ social security number (SalesSSN), and sales employees addresses ID number (SalesAddress\_Id). The “SalesSSN” foreign key refers the “SalesPersonalInfo” table, and “SalesAddress\_Id” refers the “SalesAddress” table. The “SalesEmployees” is in a many-to-one relationship with “SalesAddress” table because many employees can have an address. The “SalesEmployees” is in a one-to-one relationship with “SalesContacts” table because an employee has one personal information. The “SalesEmployees” is in a one-to-many relationship with “Invoice” table because an employee can be included in multiple invoices. All these attributes are functionally dependent as there are no partial dependencies found in the table:

* SalesEmp\_Id → SalesFirstName, SalesLastName, SalesSSN, SalesAddress

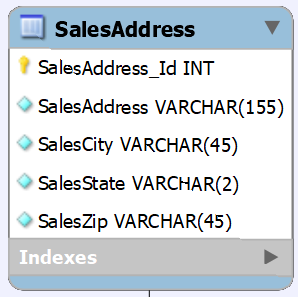
Therefore, “SalesEmployees” table is in third normal form.



The “SalesContacts” table contains information about a sales employee’s contact information. The table contains a sales employee contact ID number (SalesContact\_Id) as a primary key, sales employee’s phone number (SalesPhone), sales employee’s email (SalesEmail), and a foreign key known as “SalesEmp\_Id” that refers the “SalesEmployees” table. In addition, the “SalesContacts” table is in a many-to-one relationship with “SalesEmployees” table because many contacts information can belong to a sales employee. The attributes in the “SalesContacts” table are functionally dependent as there are no partial or transitive dependencies shown in the table:

* SalesContact\_Id → SalesPhone, SalesEmail, SalesEmp\_Id

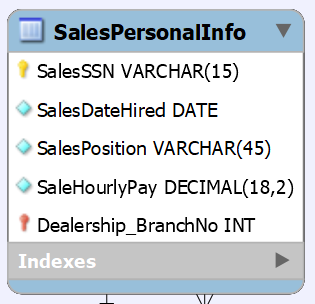
Therefore, the “SalesContacts” table is in third normal form.



The “SalesAddress” table’s purpose is to contain sales employees addresses. The table contains a primary key that is a salesperson address number ID (SalesAddress\_Id), salesperson address (SalesAddress), salesperson city (SalesCity), salesperson state (SalesState), salesperson zip code (SalesZip), and a foreign key that displays salesperson ID number (SalesEmp\_ID). The foreign key “SalesEmp\_Id” refers to “SalesEmployees” table. The “SalesAddress” table is in a one-to-many relationship with “SalesEmployees” table because an employee could have multiple addresses or live in a same address. The attributes in the “SalesAddress” table are functionally dependent since the table does not contain any non-prime attributes or transitive dependencies:

* SalesAddress\_Id → SalesAddress, SalesCity, SalesState, SalesZip

Therefore, “SalesAddress” table is in third normal form.

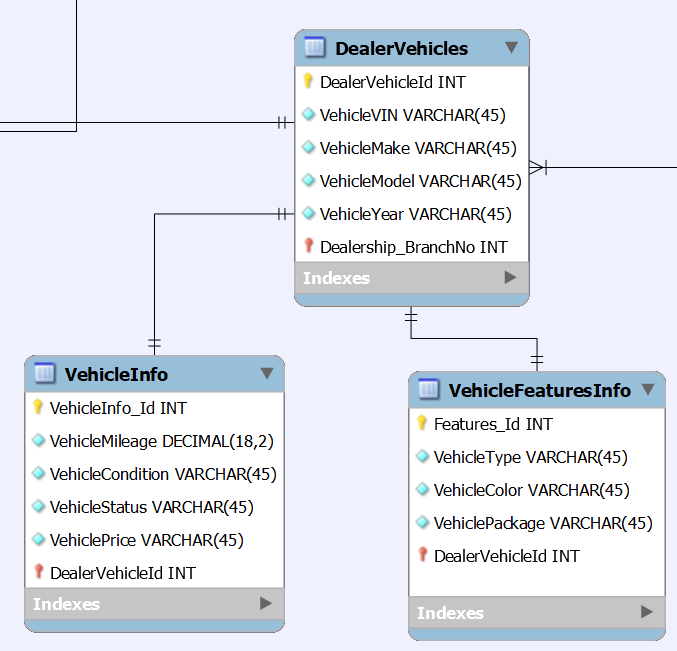


The “SalesPersonalInfo” is a table that contains a salesperson’s personal information table. The table includes four attributes and a foreign key. The table include a primary key that holds a sales employees’s social security number (SalesSSN), an employee’s hiring date (SalesDateHired), sales employee position (SalesPosition), sales employees’s hourly pay (SaleHourlyPay), and a foreign key that stores dealership branch number (Dealership\_BranchNo). The “Dealership\_BranchNo” foreign key refers the “Dealership” table. The “SalesPersonalInfo” table is in a one-to-one relationship with “SalesEmployees” table because a sales employee’s personal information belongs to one sales employee. The “SalesPersonalInfo” table is in a many-to-one relationship with “Dealership” table (which will be discussed towards the end) because many sales employee’s personal information belongs to a dealership. The attributes in the “SalesPersonalInfo” table are functionally dependent since the table does not contain any non-prime attributes or transitive dependencies:

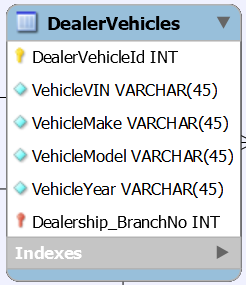
* SalesSSN → SalesDateHired, SalesPosition, SaleHourlyPay, Dealership\_BranchNo

Therefore, the “SalesPersonalInfo” table is in third normal form.

## **Dealership Vehicle entity**



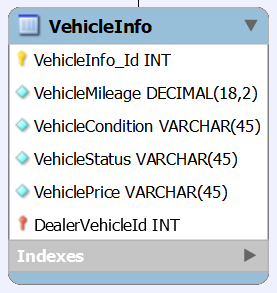
The “Dealership Vehicle” entity contains three tables that are dedicated for the vehicles that are up to sale in a dealership. The entity includes “DealerVehicles”, “VehicleInfo”, and “VehicleFeaturesInfo” tables as shown above.



The “DealerVehicles” table contains five attributes and one foreign key that refers the “Dealership” table. The “DealerVehicles” table contains a vehicle ID number (DealerVehicleId) as a primary key, vehicle VIN number (VehicleVIN), vehicle manufacturer (VehicleMake), vehicle model (VehicleModel), vehicle year (VehicleYear), and a foreign key that contains the dealership’s branch number (Dealership\_BranchNo). The “DealerVehicles” table is in a many-to-one relationship with the “Dealership” table because many dealer vehicles belong in a dealership. The “DealerVehicles” table is in a one-to-many relationship with the “Invoice” table because a dealer vehicle can be placed in many invoices. The “DealerVehicles” table is in a one-to-one relationship with “VehicleInfo” table because a dealer vehicle contains an information. The “DealerVehicles” table is in a one-to-one relationship because each vehicle contains a feature. The table has functional dependencies since there are no partial or transitive dependencies in the table:

* DealerVehicleId → VehicleVIN, VehicleMake, VehicleModel, VehicleYear, Dealership\_BranchNo

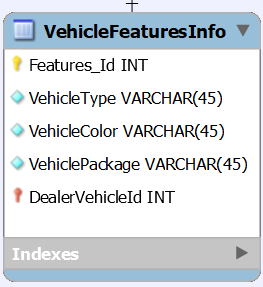
Therefore, the “DealerVehicles” table is in third normal form.



The “VehicleInfo” table contains information about the vehicle that is on sale. The table contains five attributes and one foreign key that refers the “DealerVehicles” table. The “VehicleInfo” table contains a primary key that holds the vehicle information ID number (VehicleInfo\_Id), vehicle mileage (VehicleMileage), vehicle condition (VehicleCondition), vehicle status (VehicleStatus), vehicle price (VehiclePrice), and a foreign key known as “DealerVehicleId” that comes from “DealerVehicles” table. The “VehicleInfo” table in a one-to-one relationship with “DealerVehicles” table because a vehicle from the dealer can only have one information. The attributes in the “VehicleInfo” tables are functional dependent on each other. Also, there are no partial or transitive dependencies in this table:

* VehicleInfo\_Id → VehicleMileage, VehicleCondition, VehicleStatus, VehiclePrice, DealerVehicleId

Therefore, “VehicleInfo” table is in third normal form.

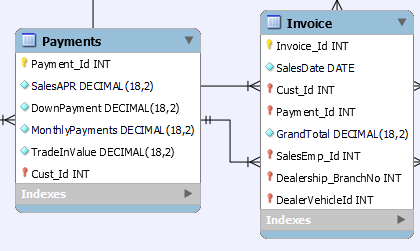


The “VehicleFeaturesInfo” table contains information about a vehicle’s features. The table contains four attributes and one foreign key from “DealerVehicles” table. The table includes primary key known as feature number ID (Features\_Id), the type of vehicle (VehicleType), the color of a vehicle (VehicleColor), the name of the type of package a vehicle (VehiclePackage), and a foreign key known as the vehicle number ID (DealerVehicleId). The “VehicleFeaturesInfo” table is in a one-to-one relationship with “DealerVehicles” table because a vehicle feature information can only be applied to a vehicle. Partial and transitive dependencies do not exist in this table:

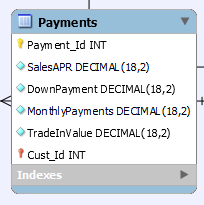
* Features\_Id → VehicleType, VehicleColor, VehiclePackage, DealerVehicleId

Therefore, the table is in third normal form.

## **Sales entity**

****

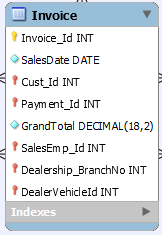
The sales entity contains two tables that are dedicated to sales data from each vehicle being sold. The names of the two tables are “Payments” and “Invoice”.

****

The “Payments” table contains five attributes and one foreign key. The “Payments” table includes primary key known as the payment ID number (Payment\_Id), the annual percentage rate (SalesAPR), a down payment (DownPayment), monthly payments (MontlyPayments), a trade in value (TradeInValue), and a foreign key containing customer number ID (Cust\_Id) that comes from “Customers” table. The “Payments” table is in a many-to-one relationship with the “Customers” table because many payments can come from a customer if they are purchasing multiple vehicles. The attributes functionally depend on each other as the “Payments” table does not contain any non-prime attributes or transitive dependencies:

* Payment\_Id → SalesAPR, DownPayment, MontlyPayments, TradeInValue, Cust\_Id

Therefore, the table is in third normal form.

****

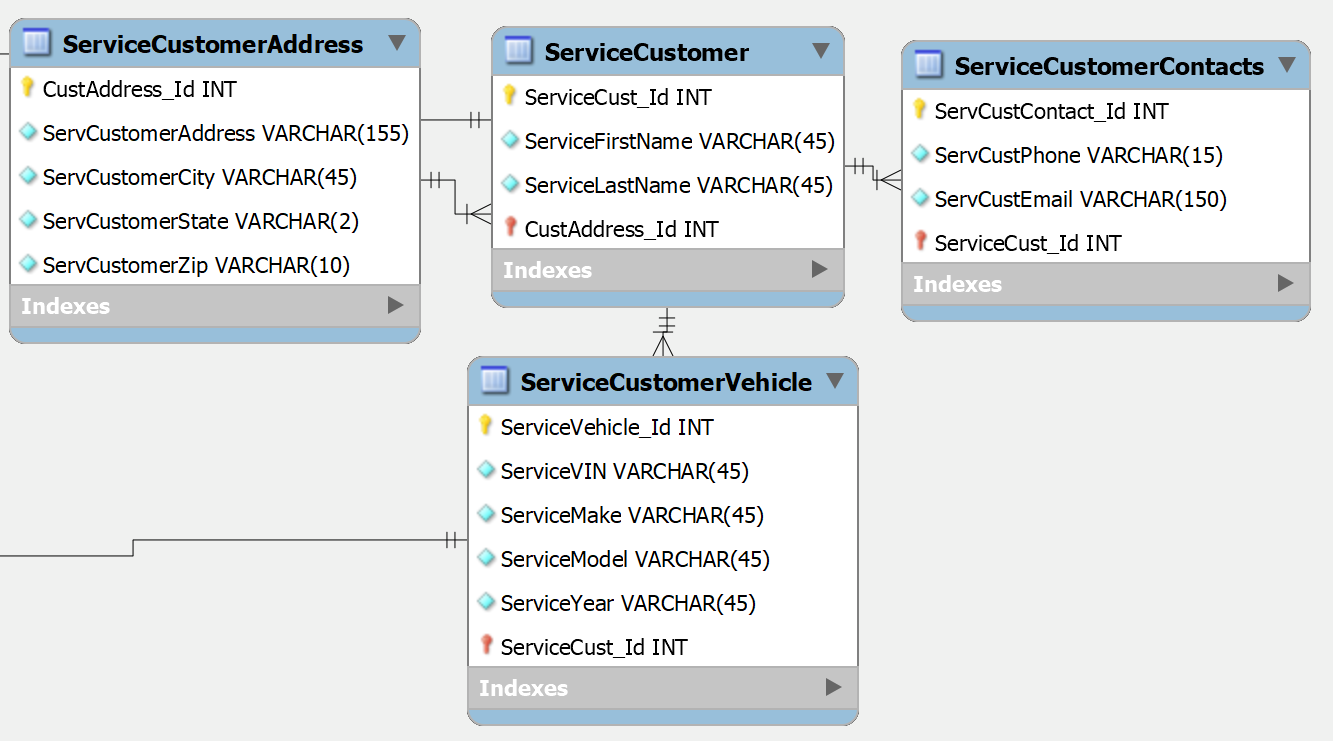
The “Invoice” table contains three attributes and five foreign keys. The “Invoice” table includes a primary key known as invoice ID number (Invoice\_Id), the date of sale (SalesDate), the grand total (GrandTotal), and five foreign keys known as the “Cust\_Id” that refers the “Customers” table, “Payment\_Id” that refers the “Payments” table, “SalesEmp\_Id” that refers the “SalesEmployees” table, “Dealership\_BranchNo” that refers the “Dealership” table, and “DealerVehicleId” that refers the “DealerVehicles” table. The “Invoice” table is in a many-to-one relationship with “Customers” table because many invoices can contain a customer. The “Invoice” table is in a many-to-one relationship with “Payments” table because many invoices can contain a payment information. The “Invoice” table is in a many-to-one relationship with “SalesEmployees” table because many invoices can include an employee’s information for contributing to making sales. The “Invoice” table is in a many-to-one relationship with “DealerVehicles” table because many invoices can contain a vehicle. The “Invoice” table is in a many-to-one relationship with “Dealership” table because many invoices can contain a dealership’s information. The attributes in the “Invoice” table functionally depend on each other since the table does not contain any non-prime attributes or transitive dependencies:

* Invoice\_Id → SalesDate, Cust\_Id, Payment\_Id, GrandTotal, SalesEmp\_Id, Dealership\_BranchNo, DealerVehicleId

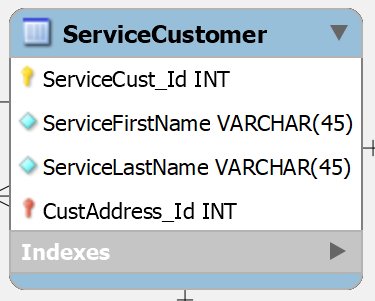
Therefore, the table is in third normal form.

# **Normalization Service Department**

## **Service Customer Entity**



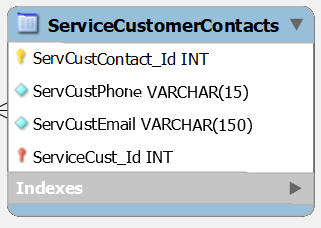
The “Service Customer” entity contains four dedicated tables that contain information about a customer in service department. The names of the tables that are utilized in this entity are “ServiceCustomer”, “ServiceCustomerContacts”, “ServiceCustomerVehicle”, and “ServiceCustomerAddress”.



The “ServiceCustomer” table contains three attributes that are based on customer’s name. The “ServiceCustomer” table contains a primary that holds the service customer ID number (ServiceCust\_Id), service customer’s first name (ServiceFirstName), and service customer’s last name (ServiceLastName). The “ServiceCustomer” table also contains a foreign key that stores customer’s address id (CustAddress\_Id) and it refers the “ServiceCustomerAddress” table. The “ServiceCustomer” table is in a one-to-many relationship with “ServiceCustomerContacts” and “ServiceCustomerVehicle” tables. The reason why “ServiceCustomer” table is in a one-to-many relationship with “ServiceCustomerContacts” table is because a customer can have multiple contacts. The reason why “ServiceCustomer” table is in a one-to-many relationship with “ServiceCustomerVehicle” table is because a customer in service can have many vehicles for service. The reason why “ServiceCustomer” table is in a one-to-many relationship with “ServiceCustomerAddress” is because a customer could in a same address. The attributes in the “ServiceCustomer” table functionally depend on each other since the table does not contain any non-prime attributes or transitive dependencies:

* ServiceCust\_Id → ServiceFirstName, ServiceLastName, CustAddress\_Id

Therefore, the table is in third normal form.



The “ServiceCustomerContacts” table contains the address information from a customer in the service department. The table contains three attributes and a foreign key that refers the “ServiceCustomer” table. The table contains a primary key that holds the service customer contact ID number (ServCustContact\_Id), service customer’s phone number (ServCustPhone), service customer’s email (ServCustEmail), and the foreign key known as the service customer ID number (ServiceCust\_Id). The “ServiceCustomerContacts” table is in a many-to-one relationship with “ServiceCustomer” table because there can be multiple contacts for a customer in the service department. There are no transitive dependencies, partial dependencies, or non-prime attributes found in the table:

* ServCustContact\_Id → ServCustPhone, ServCustEmail, ServiceCust\_Id

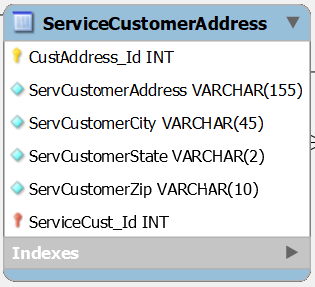
Therefore, the table is in third normal.



The “ServiceCustomerVehicle” table contains a service customer’s vehicle information. The table contains five attributes and a foreign key. The table includes a primary key that holds the customer’s service vehicle ID number (ServiceVehicle\_Id), a service vehicle’s VIN number (ServiceVIN), the vehicle’s manufacturer (ServiceMake), the vehicle’s model (ServiceModel), vehicle’s year (ServiceYear), and a foreign key known as “ServiceCust\_Id” and it refers “ServiceCustomer” table. The “ServiceCustomerVehicle” table is in a many-to-one relationship with “ServiceCustomer” table because many service vehicles can belong to a customer. The table’s attributes only relate to the service customer’s vehicle information. There are no attributes that are partially or transitively dependent as they are all functionally dependent in the “ServiceCustomerVehicle” table:

* ServiceVehicle\_Id → ServiceVIN, ServiceMake, ServiceModel, ServiceYear, ServiceCust\_Id

Therefore, the table is in third normal form.

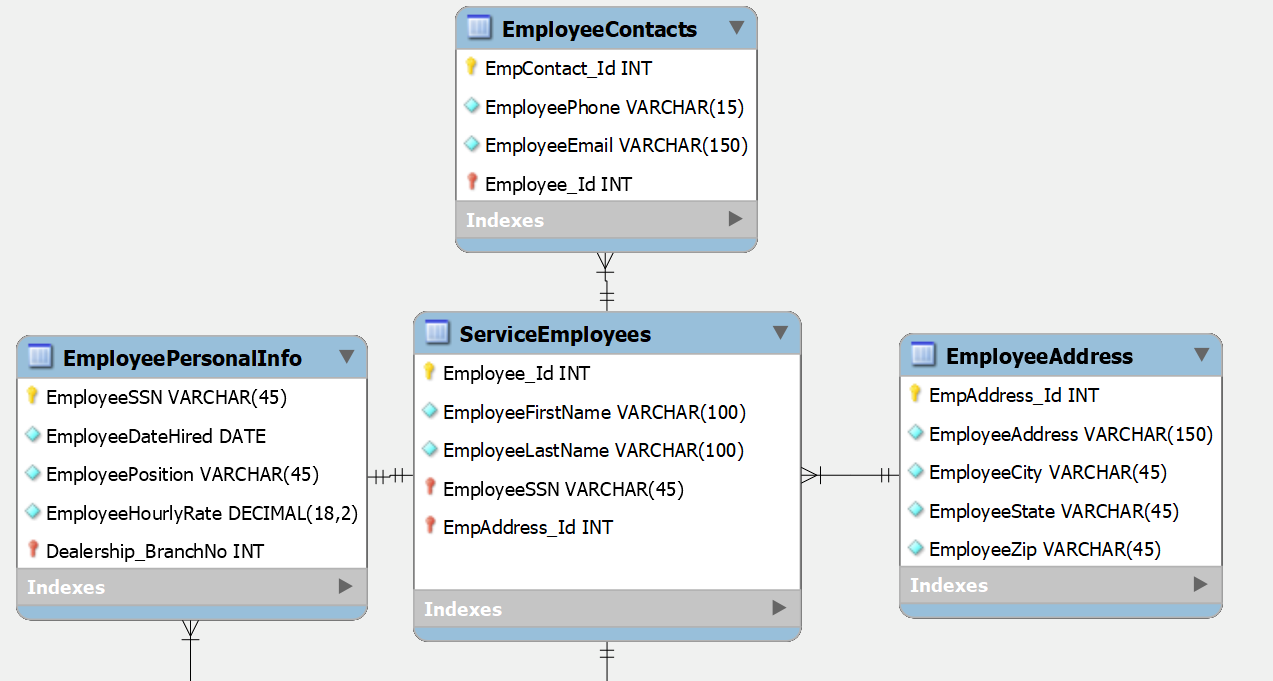


The “ServiceCustomerAddress” table contains information about a customer’s address for the service department. The table contains five attributes that includes a primary key that holds the service customer’s address ID number (CustAddress\_Id), service customer’s address (ServCustomerAddress), service customer’s city (ServCustomerCity), service customer’s state (ServCustomerState), service customer’s zip (ServCustomerZip). The “ServiceCustomerAddress” table is in a one-to-many relationship with “ServiceCustomer” table because an address can belong to multiple customers in the service department, or a customer may live in a same address. The attributes are all functionally dependent since the table does not contain any non-prime attributes. In addition, there are no transitive or partial dependencies found in the table:

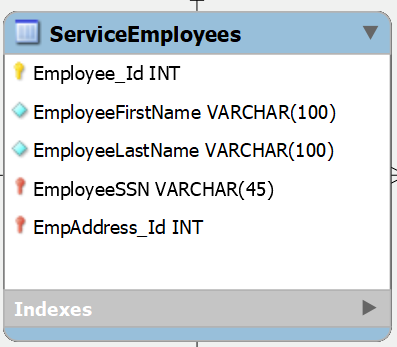
* CustAddress\_Id → ServCustomerAddress, ServCustomerCity, ServCustomerState, ServCustomerZip

Therefore, the table is set in third normal form.

## **Service Employees entity**

****

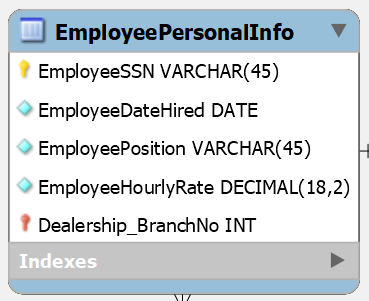
The “Service Employees” entity contains four tables that are dedicated to store service employee information in the database. The names of the tables that are utilized in this entity are “ServiceEmployees”, “EmployeeContacts”, “EmployeeAddress”, and “EmployeePersonalInfo”.



The “ServiceEmployees” table contains the names of the employees in the service department. The table contains three attributes that contain a service employee’s ID number (Employee\_Id), service employee’s first name (EmployeeFirstName), and service employee’s last name (EmployeeLastName). The table also contains two foreign keys that store service employee social security number (EmployeeSSN), and service employee address (EmpAddress\_Id). The “EmployeeSSN” foreign key refers the “SalesPersonalInfo” table, and “EmpAddress\_Id” refers the “EmployeeAddress” table. The “ServiceEmployees” table has a one-to-one relationship with “EmployeePersonalInfo” table since an employee can only have one personal info. The “ServiceEmployees” table is in a many-to-one relationship with “EmployeeAddress” table because many employees can live in an address. The “ServiceEmployees” table is in a one-to-many relationship with “EmployeeContacts” table because an employee can have multiple contacts. The attributes in the “ServiceEmployees” table does not contain any partial or transitive dependencies and there are no non-prime attributes found in the table:

* Employee\_Id → EmployeeFirstName, EmployeeLastName, EmployeeSSN, EmpAddress\_Id

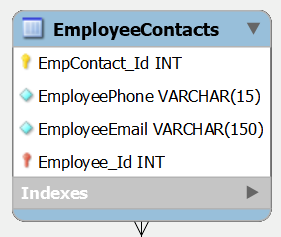
Therefore, the table is in third normal form.



The “EmployeePersonalInfo” contains service employee’s personal information. The table has four attributes and one foreign key. The table contains a primary key that holds the service employee’s social security number (EmployeeSSN), the service employee hiring date (EmployeeDateHired), the employee’s position (EmployeePosition), service employee’s hourly salary (EmployeeHourlyRate), and the foreign key known as “Dealership\_BranchNo”. The “Dealership\_BranchNo” foreign key refers to the “Dealership” table (comes from the “Dealership” entity and will be the last entity discussed towards the end). The “EmployeePersonalInfo” table is in a one-to-one relationship with “ServiceEmployees” because a personal information belongs to an employee. The “EmployeePersonalInfo” table is in a many-to-one relationship with “Dealership” table because many employees’ personal information belongs to a dealership. Looking at the table, the attributes are functionally dependent as there are no partial or transitive dependencies that exist in the table:

* EmployeeSSN → EmployeeDateHired, EmployeePosition, EmployeeHourlyRate, Dealership\_BranchNo

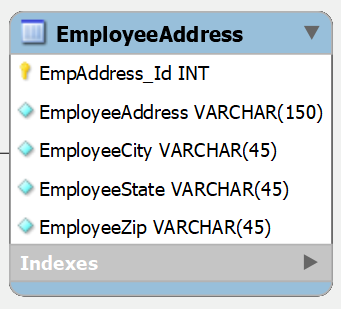
Therefore, the table is in third normal form.



The “EmployeeContacts” table contains the service employees contact information. The table contains three attributes and a foreign key that refers to “ServiceEmployees” table. The table has a primary key that is known to have a service employee’s ID number (EmpContact\_Id), service employee’s phone number (EmployeePhone), service employee’s email (EmployeeEmail), and a foreign key known as “Employee\_Id”. As stated before, the “EmployeeContacts” table is in a many-to-one relationship with “ServiceEmployees” table because there could be many contacts that come from an employee. The table does not have any partial or transitive dependencies since the attributes are functionally dependent. Also, non-prime attributes do not exist in the table:

* EmpContact\_Id → EmployeePhone, EmployeeEmail, Employee\_Id

Therefore, the table is in third normal form.

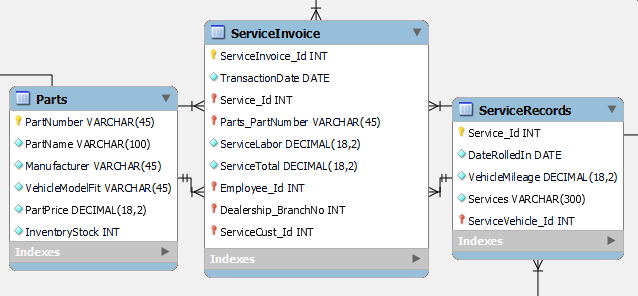


The “EmployeeAddress” table contains information about an employee’s address. The table has five attributes which contains a primary key that holds a service employee address ID number (EmpAddress\_Id), service employee address (EmployeeAddress), service employee city (EmployeeCity), service employee state (EmployeeState), and service employee zip code (EmployeeZip). As stated before, the table is in a one-to-many relationship with “ServiceEmployees” table because there could be an address that belongs to many employees. The table does not have any partial or transitive dependencies since the attributes are functionally dependent:

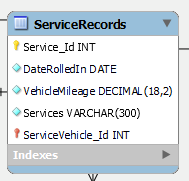
* EmpAddress\_Id → EmployeeAddress, EmployeeCity, EmployeeState, EmployeeZip

Therefore, the table is in third normal form.

## **Service Sales entity**



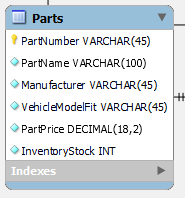
The “Service Sales” entity contains three tables that are dedicated on sales in the service department. The names of the tables that are used in this entity are “Parts”, “ServiceInvoice”, and “ServiceRecords”.



The “ServiceRecords” table contains records of a customer’s vehicle for any issues or requested services. The table contains four attributes and one foreign key that refers to “ServiceCustomerVehicle” table. The table contains a primary that is the service ID number (Service\_Id), the date that customer’s vehicle arrived at the dealership (DateRolledIn), customer’s vehicle mileage (VehicleMileage), customer issues or requested services (Services), and the foreign key known as “ServiceVehicle\_Id”. The “ServiceRecords” table has a many-to-one relationship with “ServiceCustomerVehicle” because there could be many services done for a vehicle. The “ServiceRecords” table is in a one-to-many relationship with “ServiceInvoice” table because a service record can be registered in many invoices. The “ServiceRecords” table does not contain any partial or transitive dependencies because all attributes are functional, and no non-prime attributes are found in the table:

* Service\_Id → DateRolledIn, VehicleMileage, Services, ServiceVehicle\_Id

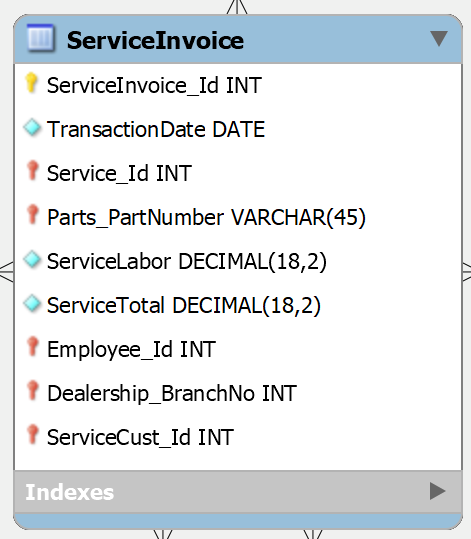
Therefore, the table is in third normal form.



The “Parts” table contains information about vehicle’s parts in the service department. The table contains six attributes. The table contains a primary key that gives a part number ID (PartNumber), part name (PartName), manufacturer (Manufacturer), parts compatibility (VehicleModelFit), part price (PartPrice), and inventory stock (InventoryStock). The “Parts” table has a one-to-many relationship with “ServiceInvoice” table because a part can be included in multiple invoices. The attributes in the “Parts” table functionally depend on each other since the table does not contain any non-prime attributes or transitive dependencies:

* PartNumber → PartName, Manufacturer, VehicleModelFit, PartPrice, InventoryStock

Therefore, the table is in third normal form.



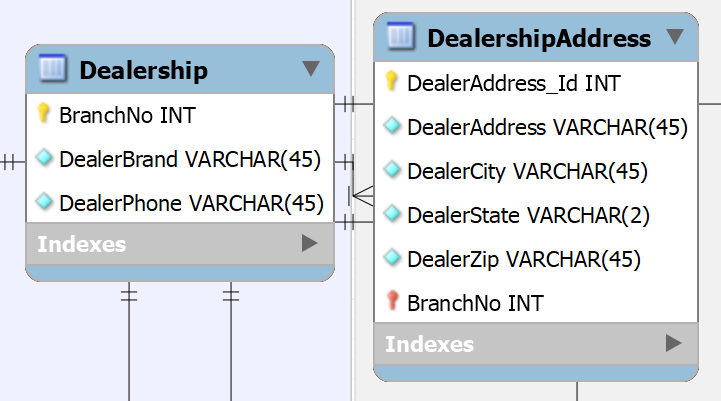
The “ServiceInvoice” table contains an invoice information from a serviced vehicle. The table contains four attributes and five foreign keys. The table contains primary key known as service invoice ID number (ServiceInvoice), the date of transaction (TransactionDate), a foreign key that refers the “ServiceRecords” table known as “Service\_Id”, a foreign key that refers the “Parts” table known as “Parts\_PartNumber”, service labor (ServiceLabor), service grand total (ServiceTotal), a foreign key that refers the “ServiceEmployees” table known as “Employee\_Id”, a foreign key that refers the “Dealership” table known as “Dealership\_BranchNo”, and a foreign key that refers the “ServiceCustomer” table known as “ServiceCust\_ID”. The “ServiceInvoice” table has a many-to-one relationship with five different tables. The “ServiceInvoice” table has a many-to-one relationship with “SalesEmployees” table because there can be many service invoices that contains a service employee information. The “ServiceInvoice” table has a many-to-one relationship with “ServiceRecords” table because there can be many invoices that contains a service record. The “ServiceInvoice” table has a many-to-one relationship with “Parts” table because there can be many invoices that contains a part. The “ServiceInvoice” table has a many-to-one relationship with “Dealership” table because because there can be many invoices that contains a dealership branch number. The “ServiceInvoice” table has a many-to-one relationship with “ServiceCustomer” table because there can be many service invoices that contains a customer’s information. The “ServiceInvoice” table does not contain any partial or transitive dependencies, and non-prime attributes since every attribute is functionally dependent:

* ServiceInvoice → TransactionDate, Service\_Id, Parts\_PartNumber, ServiceLabor, ServiceTotal,

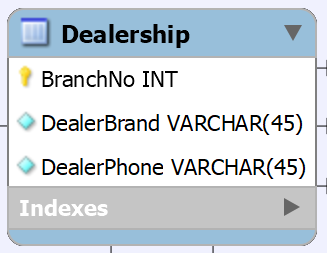
Employee\_Id, Dealership\_BranchNo, ServiceCust\_ID

Therefore, “ServiceInvoice” is in third normal form.

# **Normalization Dealership Entity**



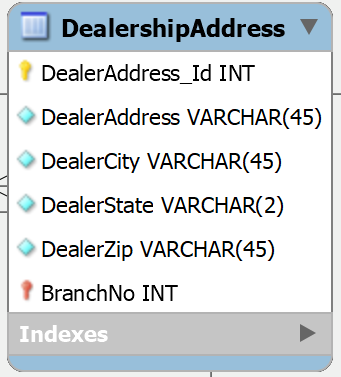
The “Dealership” entity contains two tables that are dedicated on dealership information. The names of the tables included in this entity are “Dealership” and “DealershipAddress” tables.



The “Dealership” table contains information about different dealerships. The table contains three attributes. The table contains a primary key known as the dealership branch number (BranchNo), the vehicle brands (DealerBrand), and dealership phone number (DealerPhone). The “Dealership” table is in a one-to-many relationship with “DealershipAddress” table because a dealership can contain many addresses. The “Dealership” table is in a one-to-many relationship with “DealerVehicle” table (from Sales department) because a dealership may contain many vehicles. The “Dealership” table is in a one-to-many relationship with “SalesPersonalInfo” table (from Sales department) because a dealership may contain many employees’ information in the sales department. The “Dealership” table is in a one-to-many relationship with “Invoice” table (from Sales department) because a dealership may contain many invoices. The “Dealership” table is in a one-to-many relationship with “EmployeePersonalInfo” table (from Service department) because a dealership may contain many employees’ information in the service department. The “Dealership” table is in a one-to-many relationship with “ServiceInvoice” table (from Service department) because a dealership may contain many service invoices. The attributes in the “Dealership” table are functionally dependent and do not contain any transitive or partial dependencies, and non-prime attributes:

* BranchNo → DealerBrand, DealerPhone

Therefore, the table is in third normal form.



The “DealershipAddress” table contains information about a dealership’s location. The table contains five attributes and one foreign that refers the “Dealership” table. The “DealershipAddress” table contains a primary key known as the dealership address number ID (DealerAddress\_Id), dealer’s address (DealerAddress), city (DealerCity), state (DealerState), zip code (DealerZip), and the dealership’s branch number foreign key (BranchNo). The “DealershipAddress” table is in a many-to-one relationship with “Dealership” table because there may be many addresses for a dealership. The table contains functional dependencies since it does not contain any partial or transitive dependencies, and non-prime attributes:

* DealerAddress\_Id → DealerAddress, DealerCity, DealerState, DealerZip, BranchNo

Therefore, the table is in third normal form.

# **Test plan and records**

## **Dealership Entity: Insert, Delete, Update, and Aggregate commands**

**Insert command “Dealership” table**

select \* from dealership;

insert into dealership

(BranchNo, DealerBrand, DealerPhone)

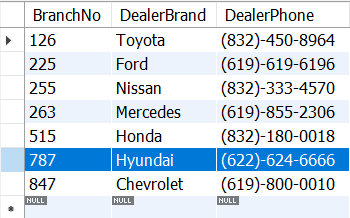
value

('787', 'Hyundai', '(622)-624-6666');

**Expected Output**



**Actual Output**

****

**Delete Command “Dealership” table**

select \* from dealership;

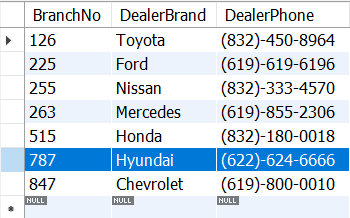
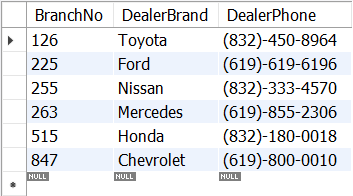
delete from dealership

where BranchNo = 787;

**Expected Output**



**Actual Output**

****

**Update Command “Dealership” table**

select \* from dealership;

update dealership

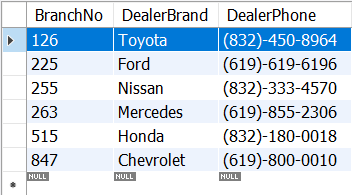
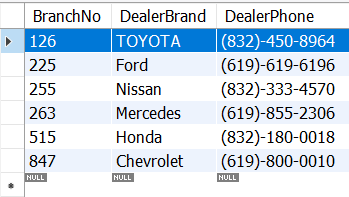
set DealerBrand= 'TOYOTA'

where BranchNo = 126;

**Expected Output**



**Actual Output**

****

**Dealership Entity: Aggregate Command for “Dealership” table**

select \* from dealership;

select count(\*) from dealership;

**Expected Output**



**Actual Output**

****

## **Sales Department: Insert, Update, Delete commands**

### **Customer Entity**

**“Customers” table**

select \* from customers; insert into customers

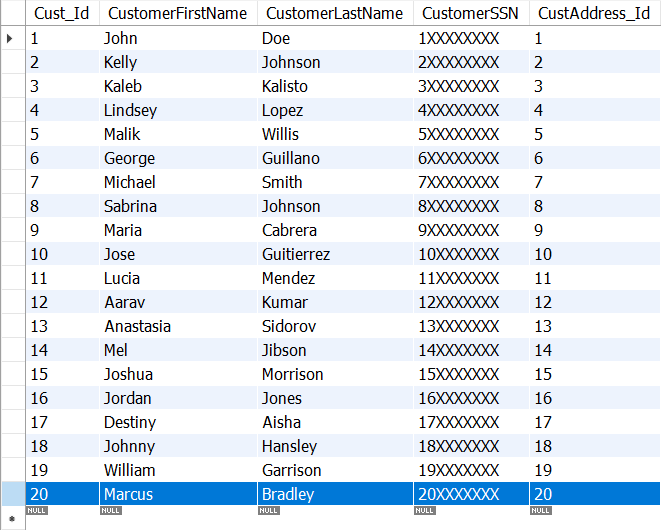
(Cust\_Id, CustomerFirstName, CustomerLastName, CustomerSSN, CustAddress\_Id)

value ('20', 'Marcus', 'Bradley', '20XXXXXXX', '20');

**Expected Output**



**Actual Output**

****

**DELETE Command**

**Delete “Customers” table**

select \* from customers;

delete from customers

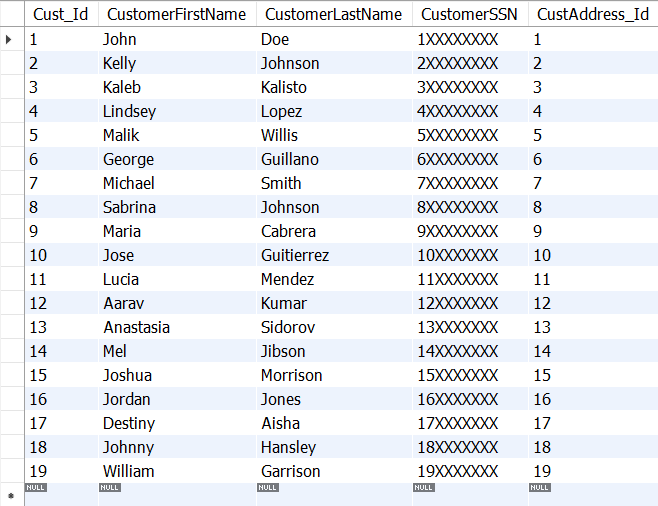
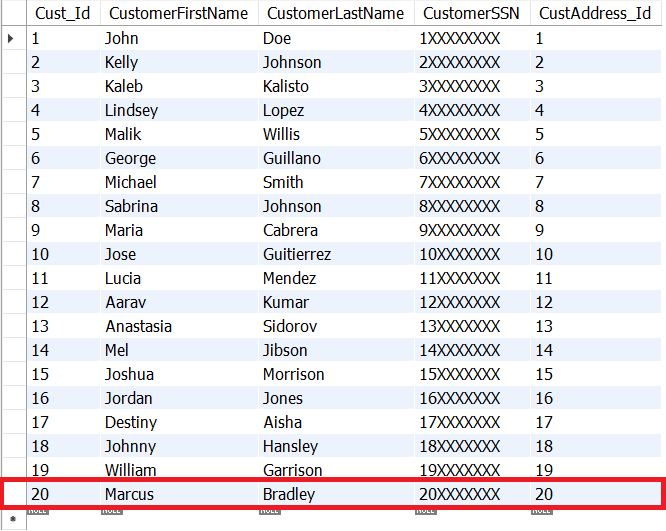
where Cust\_id = 20;

**Expected Output**





**Actual Output**

****

**Update “Customers” table**

select \* from customers;

update customers

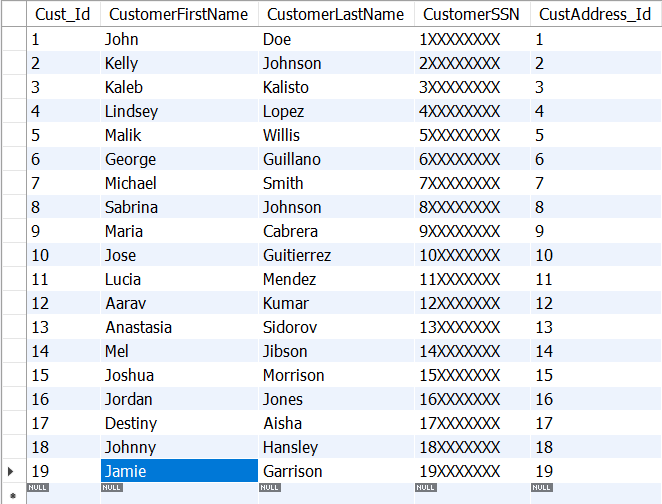
set CustomerFirstName = 'Jamie'

where Cust\_Id = 19;

**Expected Output**



**Actual result**

****

### **Sales Employees Entity**

**Insert command for “SalesEmployees” table**

select \* from salesemployees;

insert into salesemployees

(SalesEmp\_Id, SalesFirstName, SalesLastName, SalesSSN, SalesAddress\_Id)

value ('25', 'Kalil', 'Brown', '56XXXXXXX', '25')

**Expected Output**



**Actual Output**

****

**Delete command for “SalesEmployees” table**

select \* from salesemployees;

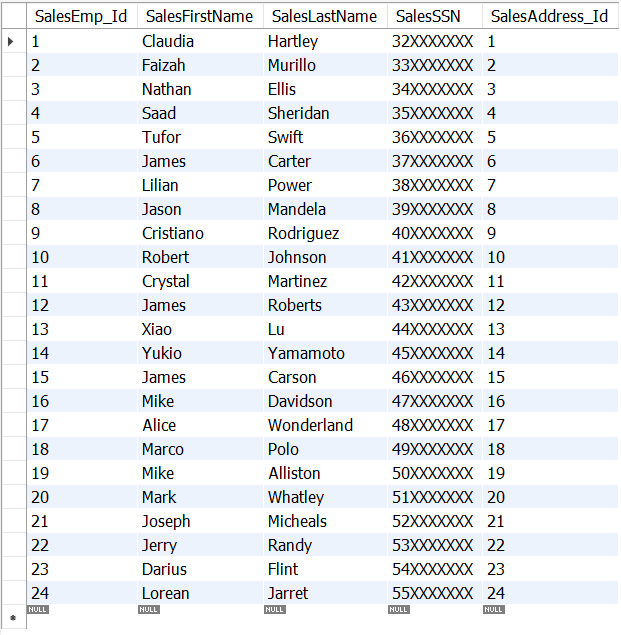
delete from salesemployees

where SalesEmp\_Id = 25;

**Expected Output**





******Actual Output**

**Update command for “SalesEmployees” table**

select \* from salesemployees;

update salesemployees

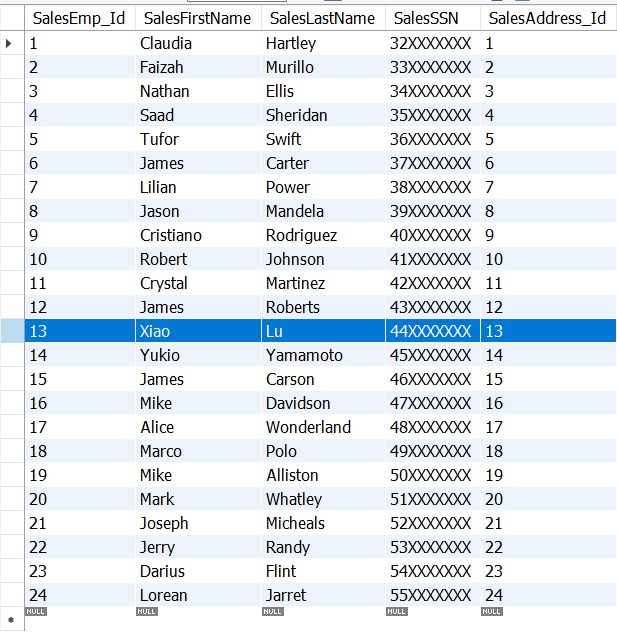
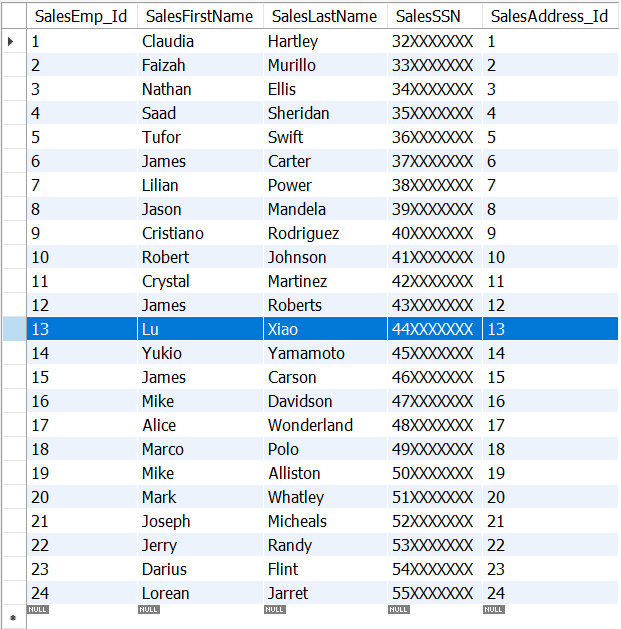
set SalesFirstName = 'Lu', SalesLastName = 'Xiao'

where SalesEmp\_Id = 13;

**Expected Output**



**Actual Output**



### **Sales Entity**

**Insert command for “Invoice” table**

select \* from invoice;

insert into invoice

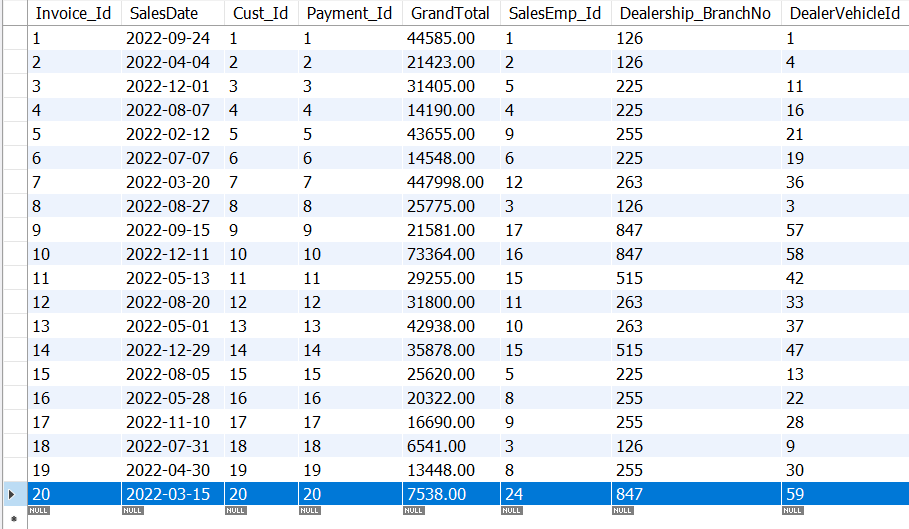
(Invoice\_Id, SalesDate, Cust\_Id, Payment\_Id, GrandTotal, SalesEmp\_Id, Dealership\_BranchNo, DealerVehicleId)

value ('20', '2022-03-15', '20', '20', '7538', '24', '847', '59');

**Expected Output**



**Actual Output**

****

**Delete command for “Invoice” table**

select \* from invoice;

delete from invoice

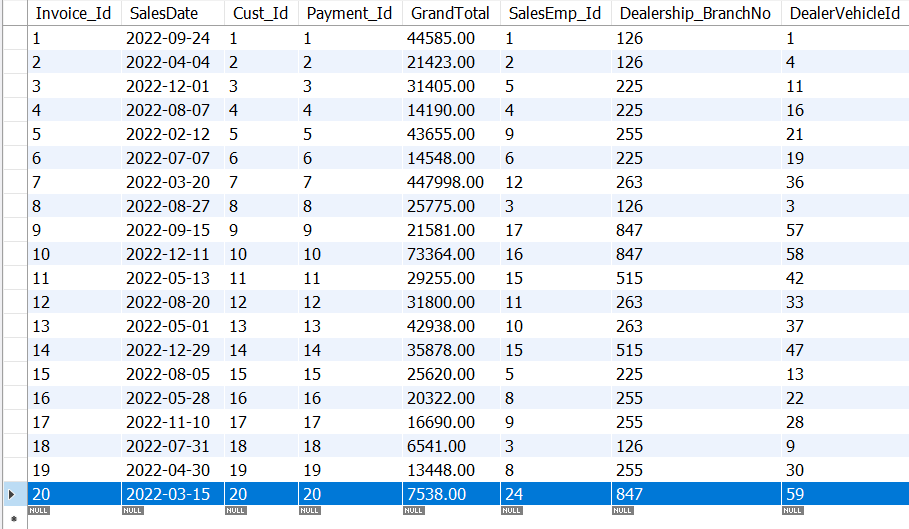
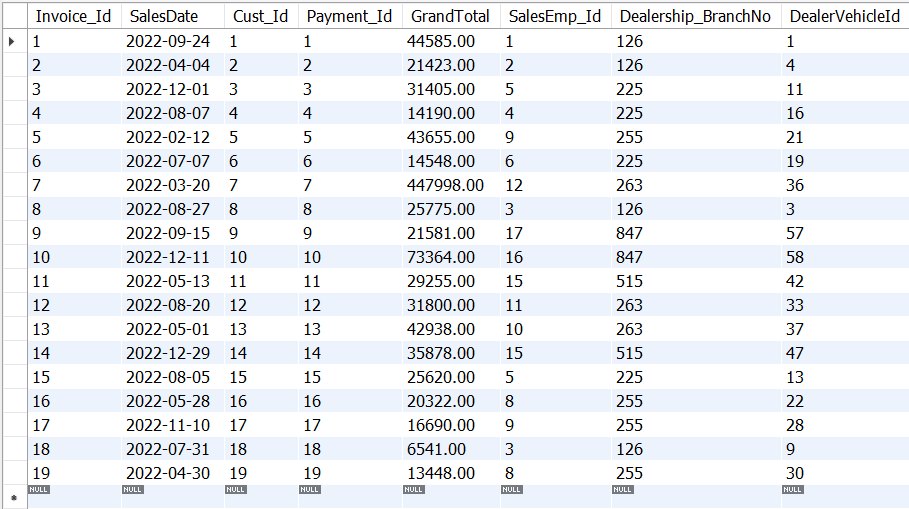
where Invoice\_Id = 20;

**Expected Output**





**Actual Output**

****

**Update command for “Invoice” table**

select \* from invoice;

update invoice

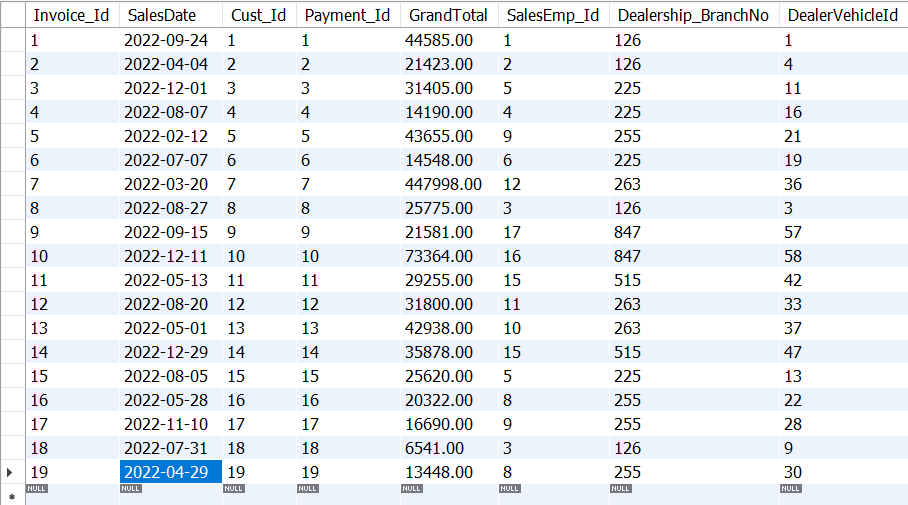
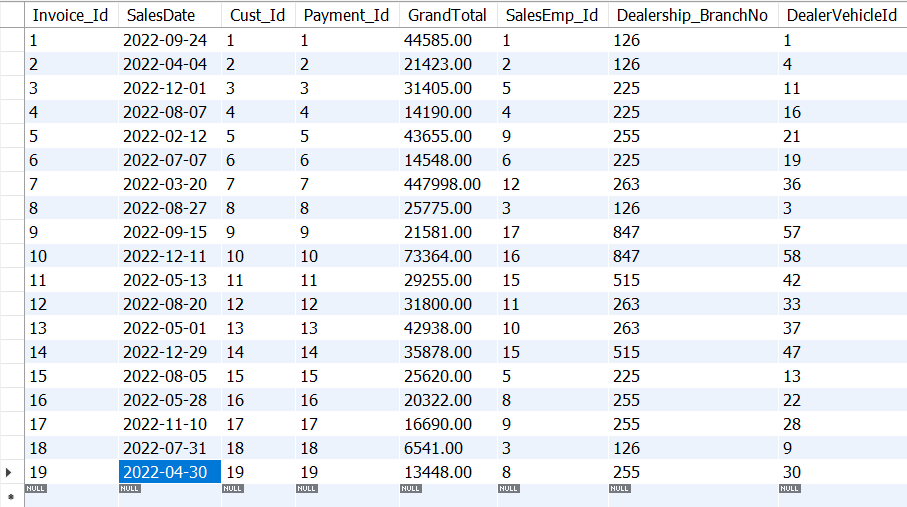
set SalesDate = '2022-04-29'

where Invoice\_Id = 19;

**Expected Output**



**Actual Output**

****

### **Dealership Vehicle Entity**

**Insert command for “DealerVehicles” table**

select \* from dealervehicles;

insert into dealervehicles

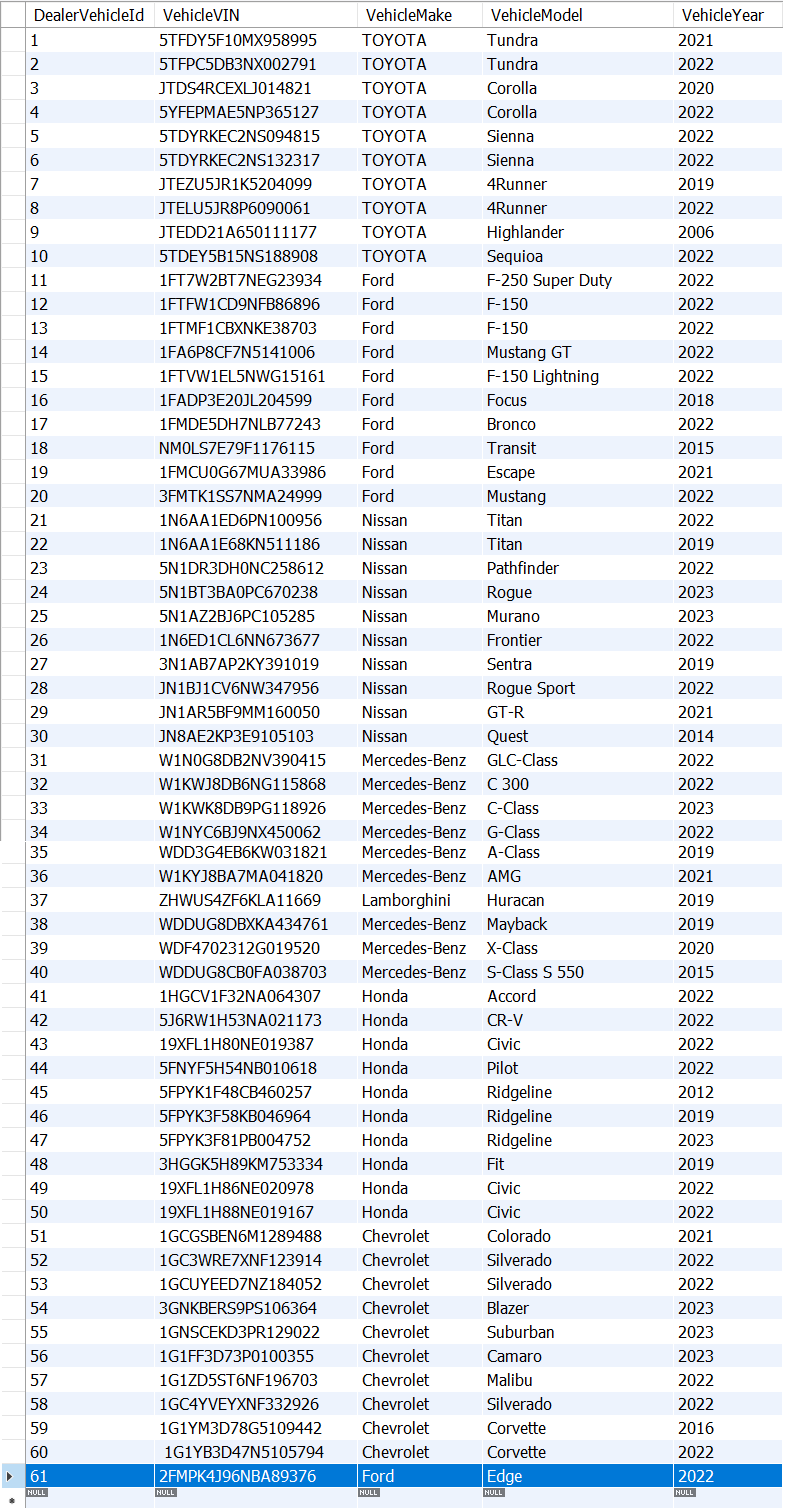
(DealerVehicleId, VehicleVIN, VehicleMake, VehicleModel, VehicleYear)

value ('61', '2FMPK4J96NBA89376', 'Ford', 'Edge', '2022');

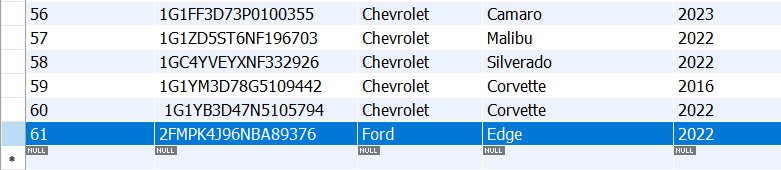
**Expected Output**



**Actual Output**

****

**Zoomed it:**

****

**Delete Command for “DealerVehicles” table**

select \* from dealervehicles;

delete from dealervehicles

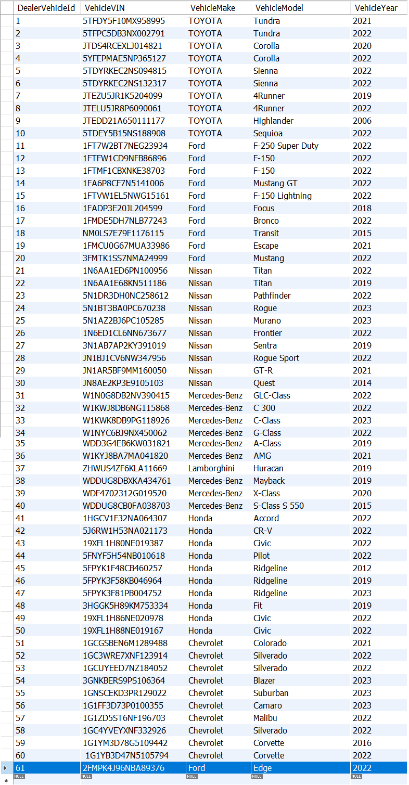
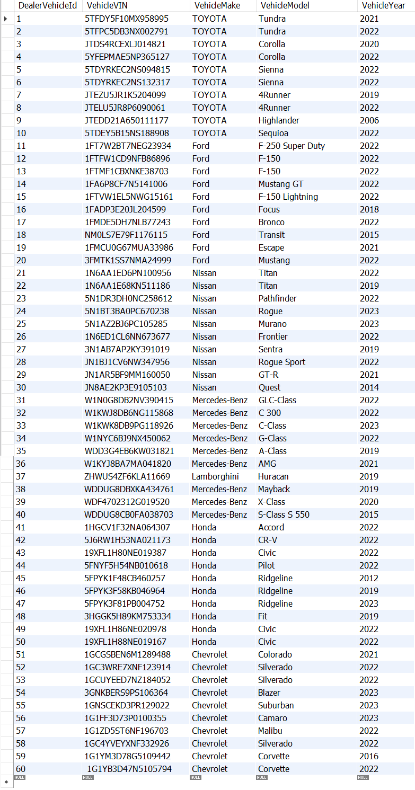
where DealerVehicleId = 61;

**Expected Output**

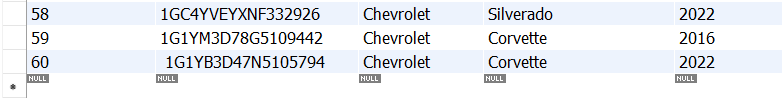
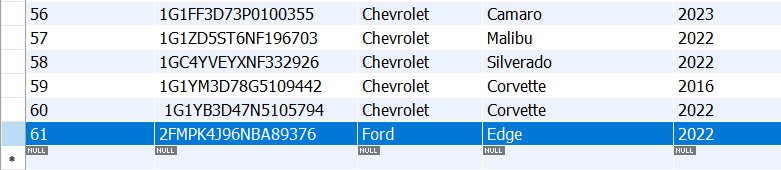




**Actual Output**

****

**Zoomed it:**

****

**Update command for “DealerVehicles” table**

select \* from dealervehicles;

update dealervehicles

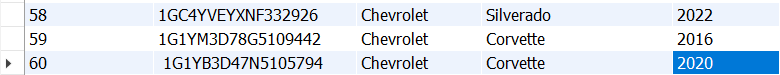
set VehicleYear = '2021'

where DealerVehicleId = 60;

**Expected Output**



**Actual Output**

****

**Zoomed it:**

## **Join Commands Sales Department**

**Join Customer and Sales Invoice**

select customers.Cust\_Id, customers.CustomerFirstName, customers.CustomerLastName,

invoice.Invoice\_Id, invoice.SalesDate, invoice.Payment\_Id, invoice.GrandTotal,

invoice.SalesEmp\_Id, invoice.Dealership\_BranchNo

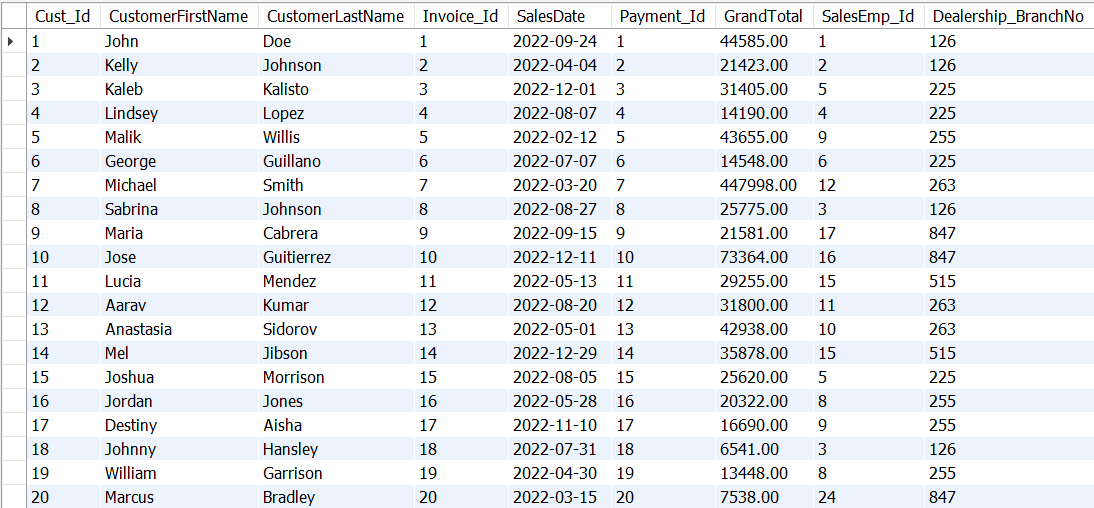
from customers

inner join invoice

on customers.Cust\_Id = invoice.Cust\_Id;

**Expected Output**

**Actual Output**



**Join Sales Employees and invoice**

select salesemployees.SalesEmp\_Id, salesemployees.SalesFirstName, salesemployees.SalesLastName,

invoice.Invoice\_Id, invoice.SalesDate, invoice.Payment\_Id, invoice.GrandTotal,

invoice.Dealership\_BranchNo

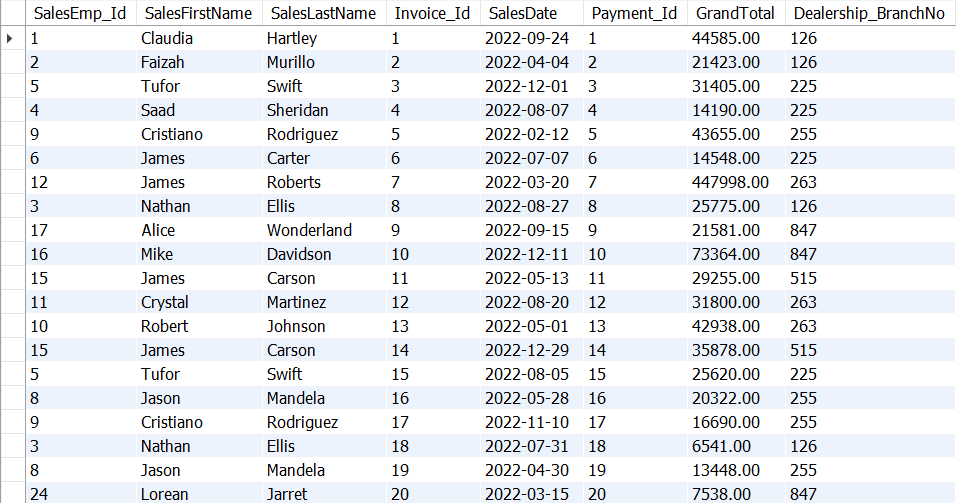
from salesemployees

inner join invoice

on salesemployees.SalesEmp\_Id = invoice.SalesEmp\_Id;

**Expected Output**

**Actual Output**

****

**Join Dealership and invoice**

select dealership.BranchNo, dealership.DealerBrand,

invoice.Invoice\_Id, invoice.SalesDate, invoice.Payment\_Id, invoice.GrandTotal

from dealership

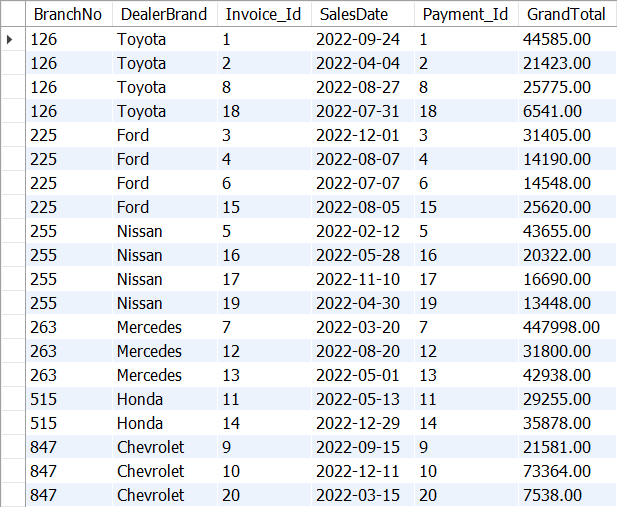
inner join invoice

on dealership.BranchNo = invoice.Dealership\_BranchNo;

**Expected Output**



**Actual Output**



**Join Dealer Vehicles and invoice**

select dealervehicles.VehicleVIN, dealervehicles.VehicleMake, dealervehicles.VehicleModel,

dealervehicles.VehicleYear,

invoice.Invoice\_Id, invoice.SalesDate, invoice.Payment\_Id, invoice.GrandTotal,

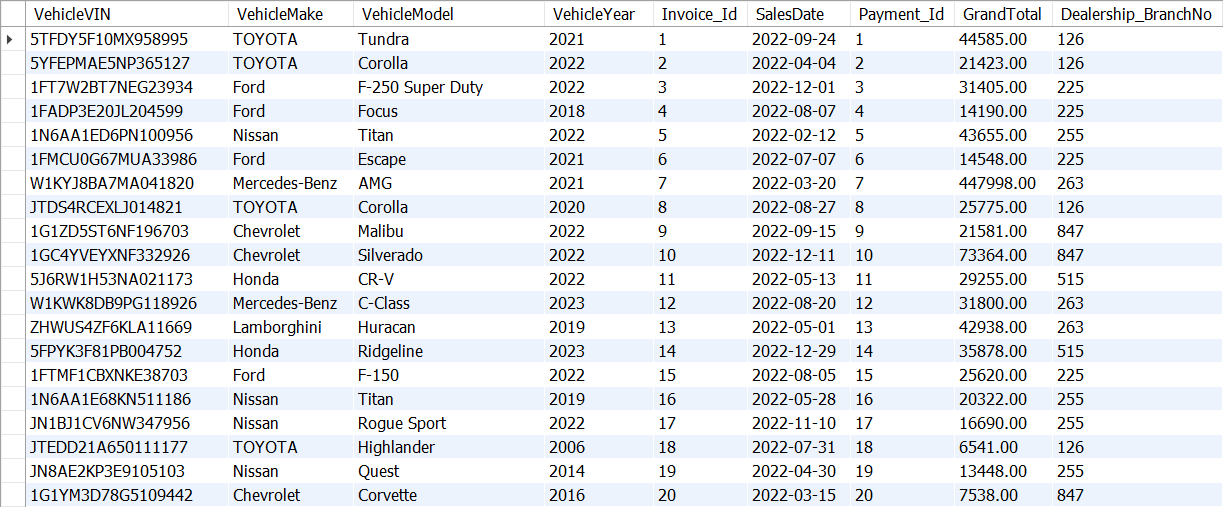
invoice.Dealership\_BranchNo

from dealervehicles

inner join invoice

on dealervehicles.DealerVehicleId = invoice.DealerVehicleId;

**Expected Output**

**Actual Output**

## **Aggregate Commands Sales Department**

**Aggregate Customers Entity**

**“Customers” table count**

select \* from customers;

select count(\*) from customers;

**Expected Output**



**Actual result**

****

**Aggregate Sales Employees Entity**

select \*

from salesemployees;

select count(\*) from salesemployees;

**Expected Output**



**Actual result**

****

**Aggregate Sales Entity**

**“Invoice” table sum**

select \* from invoice;

select sum(GrandTotal) from invoice;

**Expected Output**



**Actual result**

****

**Aggregate Dealership Vehicle Entity**

**“DealerVehicles” table count**

select \* from dealervehicles;

select count(\*) from dealervehicles;

**Expected Output**



**Actual result**

****

## **Service Department: Insert, Update, Delete commands**

### **Service Customer Entity**

**Insert command for “ServiceCustomer” table**

select \* from servicecustomer;

insert into servicecustomer

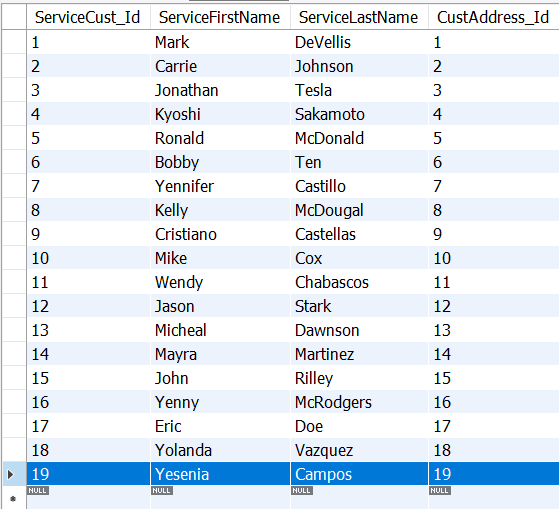
(ServiceCust\_Id, ServiceFirstName, ServiceLastName, CustAddress\_Id)

values ('19', 'Yesenia', 'Campos', '19');

**Expected Output**



**Actual Output**

****

**Delete command for “ServiceCustomer” table**

select \* from ServiceCustomer;

delete from ServiceCustomer

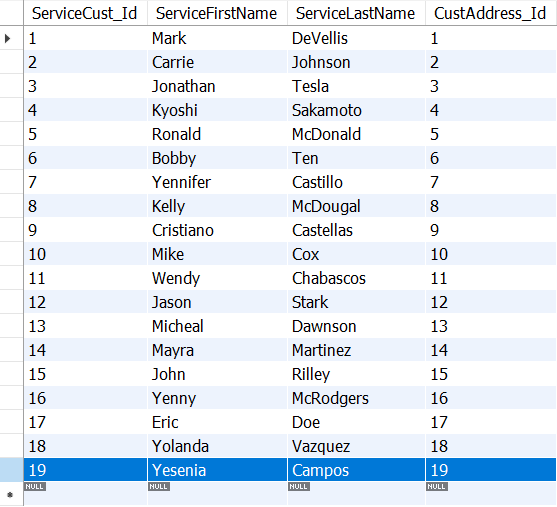
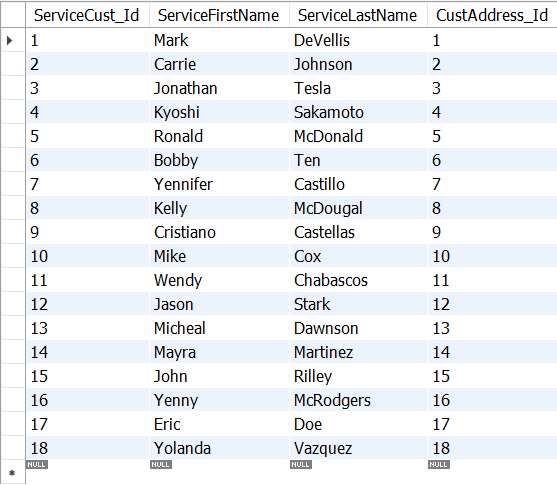
where SalesEmp\_Id = 25;

**Expected Output**





**Actual Output**

****

**Update command for “ServiceCustomer” table**

select \* from servicecustomer;

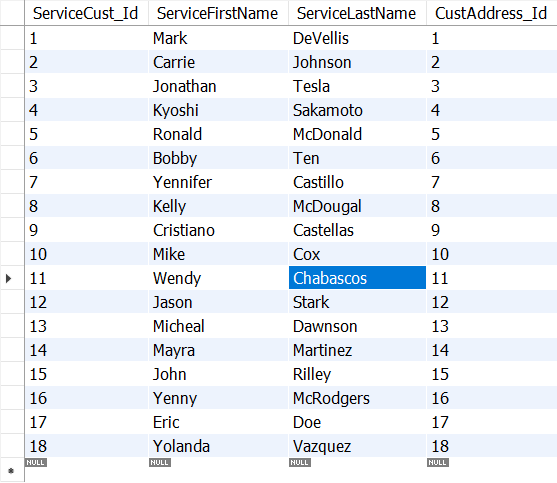
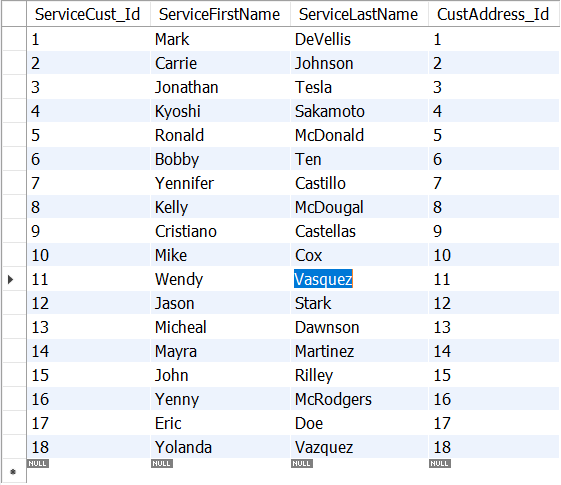
update servicecustomer

set ServiceLastName = 'Vasquez'

where ServiceCust\_Id = 11;

**Expected Output**

**Actual Output**



### **Service Employees Entity**

**Insert command for “ServiceEmployees” table**

select \* from serviceemployees;

insert into serviceemployees

(Employee\_Id, EmployeeFirstName, EmployeeLastName, EmployeeSSN, EmpAddress\_Id)

values

('25', 'Kenya', 'Esparza', '80XXXXXXX', '25')

**Expected Output**



**Actual Output**

****

**Delete command for “ServiceEmployees” table**

select \* from serviceemployees;

delete from serviceemployees

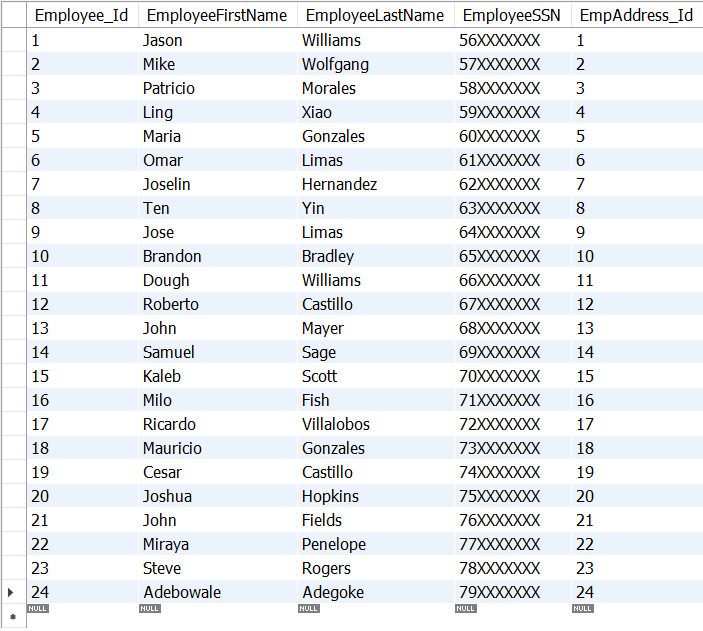
where Employee\_Id = 25;

**Expected Output**





**Actual Output**

****

**Update command for “ServiceEmployees” table**

select \* from serviceemployees;

update serviceemployees

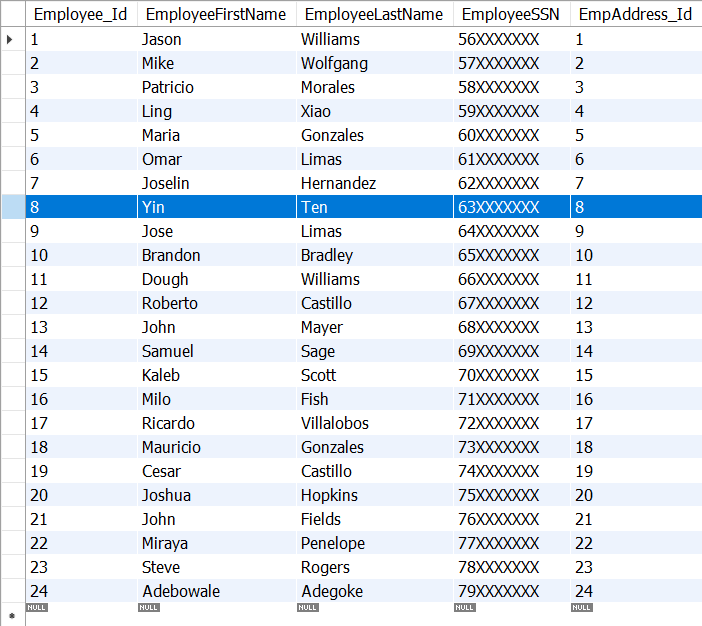
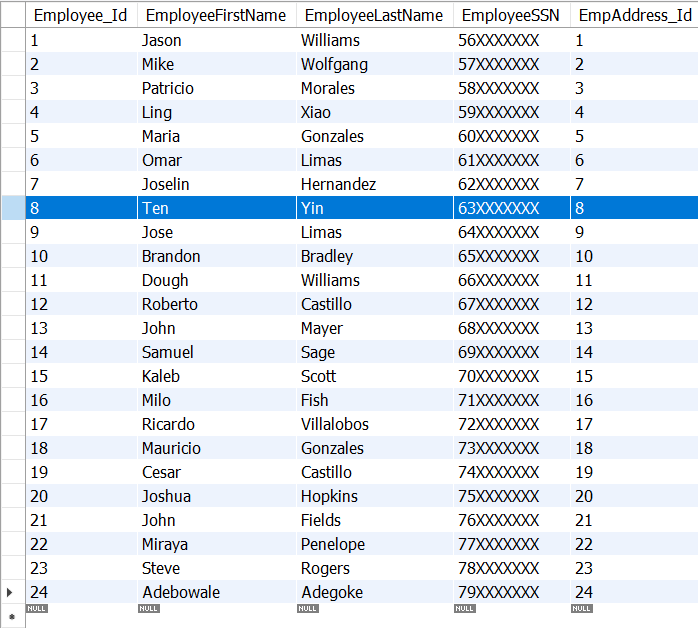
set EmployeeFirstName = 'Yin', EmployeeLastName = 'Ten'

where Employee\_Id = 8;

**Expected Output**



**Actual Output**



### **Service Invoice Entity**

**Insert command “ServiceInvoice” table**

select \* from serviceinvoice;

insert into serviceinvoice

(ServiceInvoice\_Id, TransactionDate, Service\_Id, Parts\_PartNumber, ServiceLabor, ServiceTotal, Employee\_Id, Dealership\_BranchNo, ServiceCust\_Id)

value ('20', '2022-12-02', '20','N/A', '25.50', '25.50', '25', '515', '19');

**Expected Output**

**Actual Output**

****

**Delete command “ServiceInvoice” table**

select \* from serviceemployees;

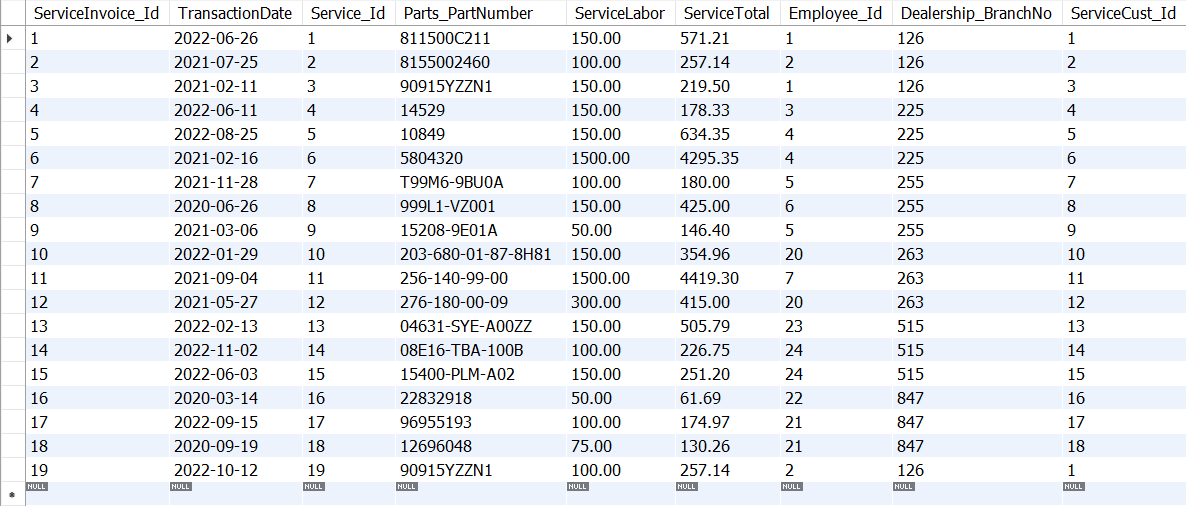
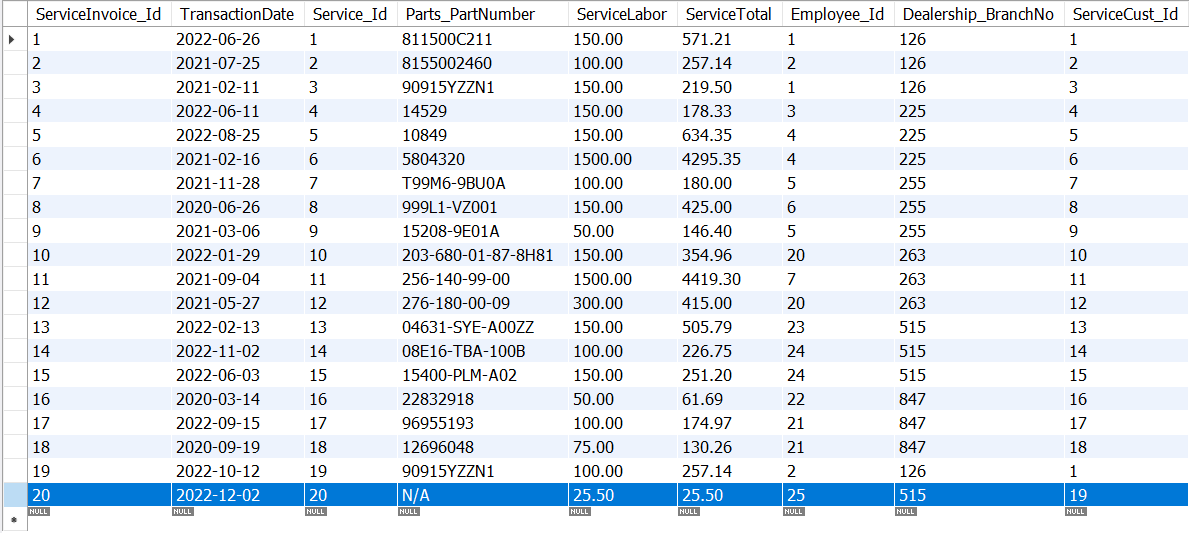
delete from serviceemployees

where Employee\_Id = 20;

**Expected Output**



**Actual Output**



**Update command “ServiceInvoice” table**

select \* from serviceinvoice;

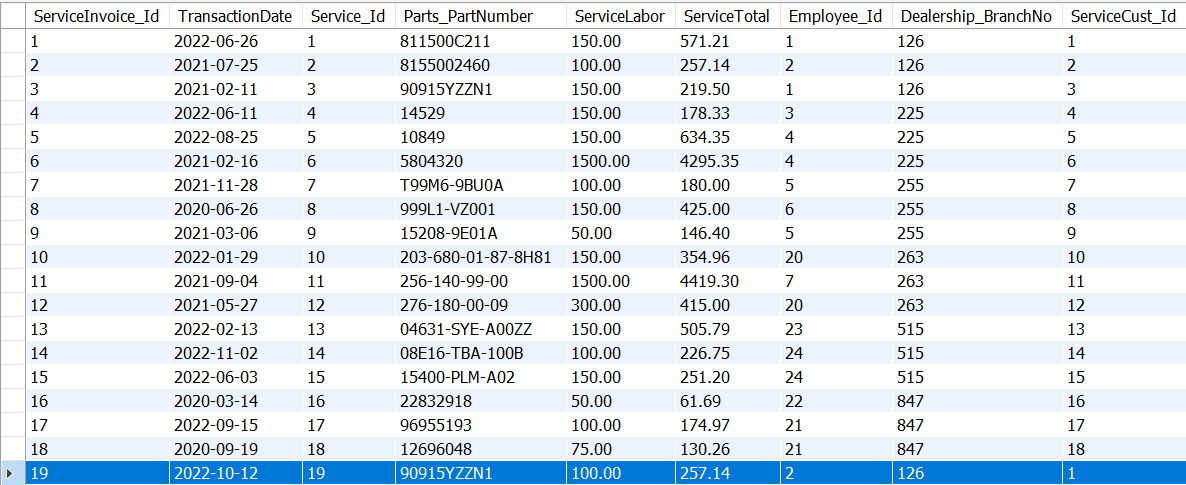
update serviceinvoice

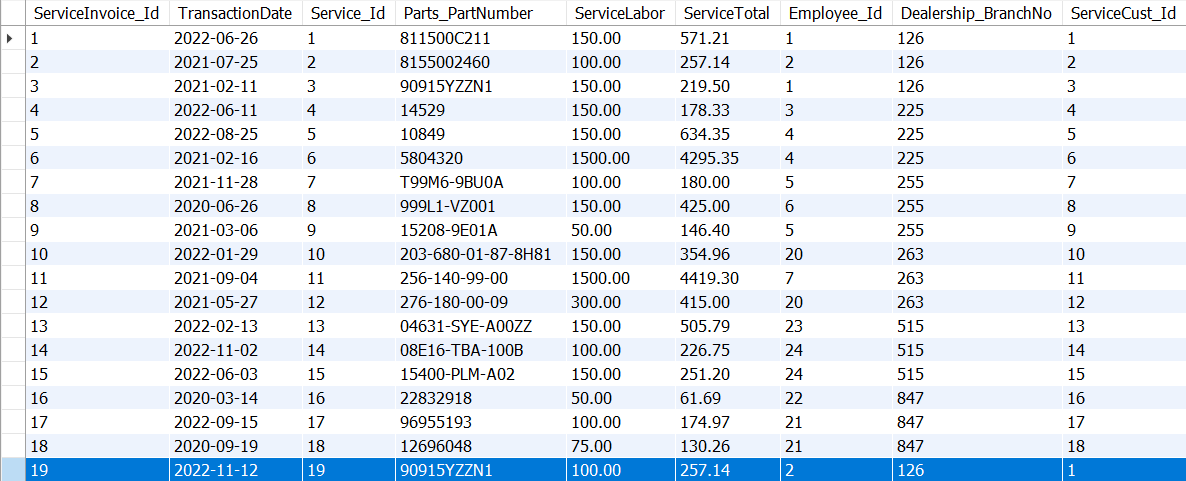
set TransactionDate = '2022-11-12'

where ServiceInvoice\_Id = 19;

**Expected Output**



**Actual Output**



## **Join Commands Service Department**

**Join “ServiceCustomer” table with “ServiceInvoice” table.**

select servicecustomer.ServiceCust\_Id, servicecustomer.ServiceFirstName, servicecustomer.ServiceLastName,

serviceinvoice.TransactionDate, serviceinvoice.Service\_Id, serviceinvoice.Parts\_PartNumber,

serviceinvoice.ServiceLabor, serviceinvoice.ServiceTotal,

serviceinvoice.Employee\_Id, serviceinvoice.Dealership\_BranchNo

from servicecustomer

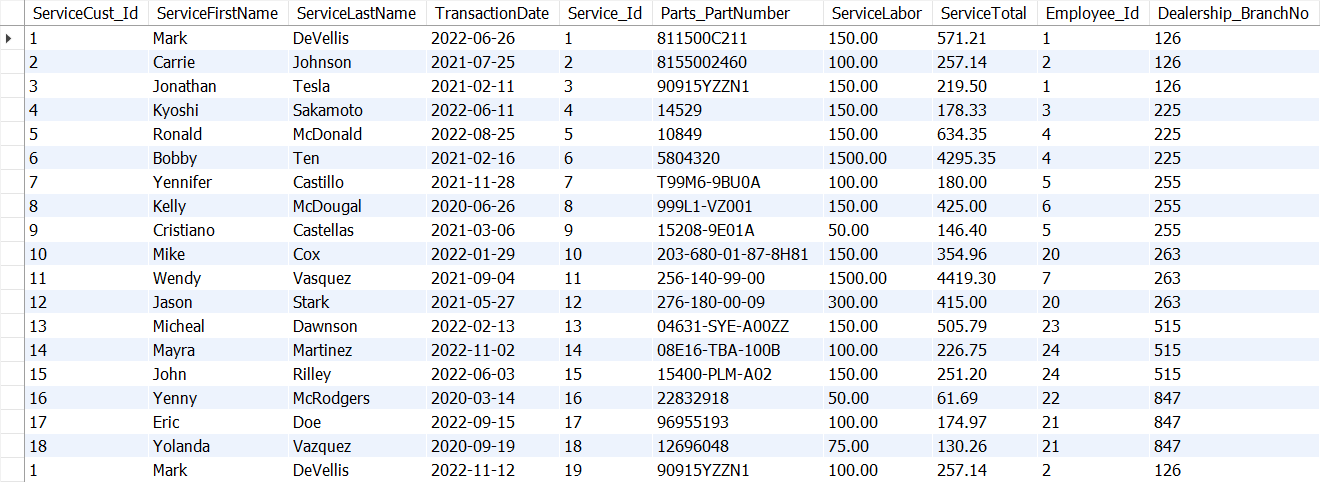
inner join serviceinvoice

on servicecustomer.ServiceCust\_Id = serviceinvoice.ServiceCust\_Id;

**Expected Output**



**Actual Output**



**Join “ServiceEmployees” table with “ServiceInvoice” table.**

select serviceemployees.Employee\_Id, serviceemployees.EmployeeFirstName, serviceemployees.EmployeeLastName,

serviceinvoice.TransactionDate, serviceinvoice.Service\_Id, serviceinvoice.Parts\_PartNumber,

serviceinvoice.ServiceLabor, serviceinvoice.ServiceTotal,

serviceinvoice.Dealership\_BranchNo

from serviceemployees

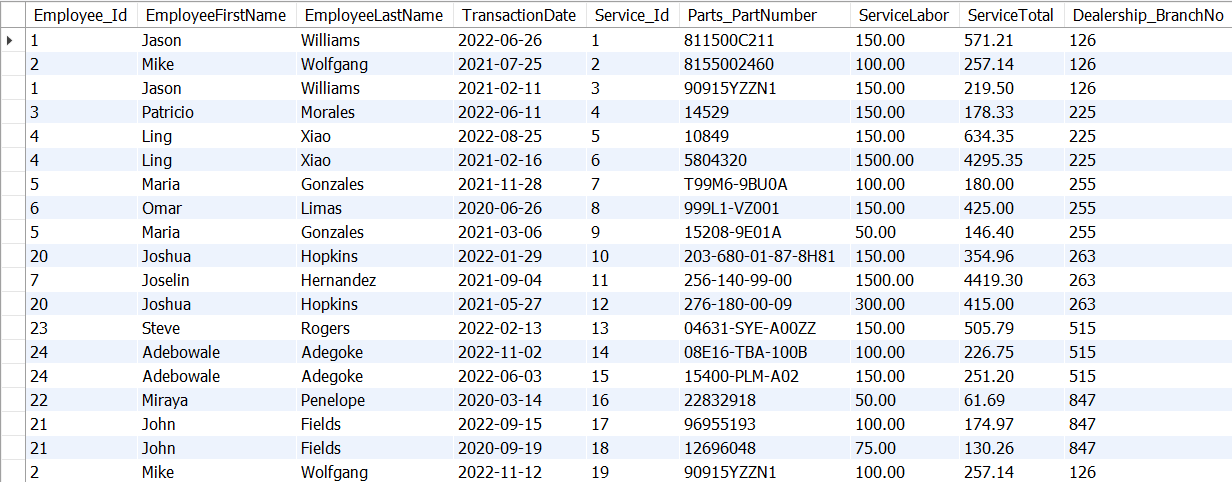
inner join serviceinvoice

on serviceemployees.Employee\_Id = serviceinvoice.Employee\_Id;

**Expected Output**



**Actual Output**



**Join “Dealership” table with “ServiceInvoice” table.**

select dealership.BranchNo, dealership.DealerBrand, dealership.DealerPhone,

serviceinvoice.TransactionDate, serviceinvoice.Service\_Id, serviceinvoice.Parts\_PartNumber,

serviceinvoice.ServiceLabor, serviceinvoice.ServiceTotal

from dealership

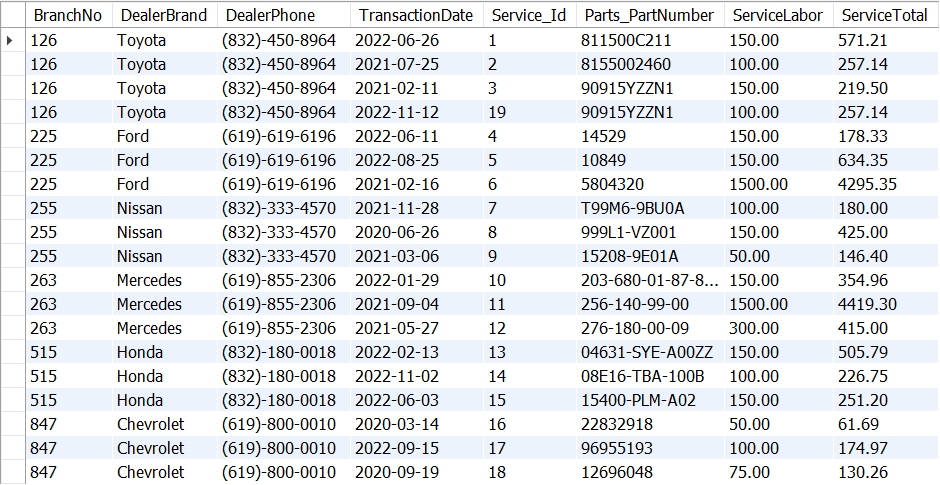
inner join serviceinvoice

on dealership.BranchNo = serviceinvoice.Dealership\_BranchNo;

**Expected Output**



**Actual Output**



## **Aggregate Commands Service Department**

**Aggregate Command Service Customer Entity**

**“ServiceCustomer” table count**

select \* from servicecustomer;

select count(\*)

from servicecustomer;

**Expected Output**



**Actual Output**



**Aggregate Command Service Employees Entity**

**“ServiceEmployees” table count**

select \* from serviceemployees;

select count(\*) from serviceemployees;

**Expected Output**



**Actual Output**

****

**Aggregate Command Service Invoice Entity**

**“ServiceInvoice” table count**

select \* from serviceinvoice;

select count(\*) from serviceinvoice;

**Expected Output**



**Actual Output**

****

# **Conclusion**

In conclusion, this database was built for the purpose of maintaining data for dealerships. Originally, I wanted to simulate how a dealership typically holds data in a database. However, I assumed that all tables had to be in third normal form. Therefore, I went through many redesigns in my database in order to achieve third normal form for all tables. I went through extensive research in understanding the concept of normalization and attempted to make all my tables in third normal form, and as a result I ended up having a total of 28 tables. I was able to implement the use cases and SQL statements for every table in this database. Some the data in this database was made by me (such as fake social security numbers, phone numbers, addresses, etc.). Some data I had to pick online (such as VIN numbers, car model, etc.) to enhance the dealership experience. I have provided many functions for manipulating data and achieved all the listed objectives. Since there are too many tables in this database, there was a limited amount of time to implement all the functions for every table. However, in the future, I plan to implement more functions for every table in this database.

# **Resources**

<https://randomvin.com/>

<https://www.hausautogroup.com/documents-to-bring-when-buying-a-car.htm#:~:text=Proof%20of%20Residence,home's%20lease%20or%20mortgage%20agreement>.

<https://www.cars.com/>

<https://www.cargurus.com/Cars/inventorylisting/viewDetailsFilterViewInventoryListing.action?entitySelectingHelper.selectedEntity=d2361&zip=77090&pid=directVDPSimilarListing#listing=333675160>

<https://houston.craigslist.org/search/cta#search=1~gallery~0~0>

<https://www.classicchevysugarland.com/searchnew.aspx?search=new&utm_campaign=dds-branding-namekeywords&utm_content=branding-search&utm_medium=cpc&utm_source=dominion-google-ads>

<https://www.toyota.com/gst/deals-incentives/?viewAllOffers=1&srchid=SEM:700000002435463:GOOGLE:71700000089852642:58700007613107096:p68780760605:574633314601&gclid=Cj0KCQiAveebBhD_ARIsAFaAvrGguj4LzegI6hsC3cD6MiJWYKvdUmTa3bHMMPtmJpMobAnWtMmkpikaAnJLEALw_wcB&gclsrc=aw.ds>

<https://www.geeksforgeeks.org/third-normal-form-3nf/>

<https://parts.ford.com/en.html>

<https://www.sterlingmccalltoyota.com/parts/>

<https://www.hondapartsnow.com/>

<https://parts.chevrolet.com/>

<https://www.mbusa.com/en/owners/parts?sd_campaign_type=Search&sd_digadprov=Resolution&sd_campaign=Service_Google_Brand_Genuine+Parts&sd_channel=GOOGLE&sd_adid=General+Parts&sd_digadkeyword=mercedes+parts&gclid=Cj0KCQiAveebBhD_ARIsAFaAvrHLdy8QF8XGYcqReB17q1dzxsMu-8XItpCw_sHgopliUo-8a7nT1-kaAmvpEALw_wcB&gclsrc=aw.ds>

<https://parts.nissanusa.com/>

<https://www.mbusa.com/en/cpo/inventory?gclid=Cj0KCQiAveebBhD_ARIsAFaAvrEj2a_kfJ2soVLdCYwUTcvs5biIlI38YgrLc8iwEo4m0JXQsqWmwVwaAlSJEALw_wcB&gclsrc=aw.ds>

<https://uhd.photoshelter.com/galleries/C0000k1pYEiR6HDE/G0000Xd1IeCdCGY4/Main-UHD-Logos-Mark>

<https://www.uhd.edu/administration/university-relations/Pages/UHD-Logos.aspx>

# **Video Links**

**Google Drive:**

<https://drive.google.com/file/d/1nImgQ7Yz7c-ashvd7jSTXH4FdGJmMWuQ/view?usp=share_link>

**One Drive:**

<https://1drv.ms/v/s!AsJmZPxfinREnA9pf-S8A1dY2GMI?e=xdrULB>