Project Design Report

Car Dealership Database System

TEAM L.I.T.

04/27/2020

Ivan Shapirov

Tommy Lim

LaKeesha Patterson

Introduction

For our project, we designed a database system for a car dealership with multiple locations. The dealership needs our help to keep track of all its employees, locations and inventory at each location. We wanted our project to be fun and interesting so we decided to model a car dealership. We can all relate to needing to purchase a vehicle since the majority of us rely heavily on our vehicle as our main mode of transportation. Also, we found modeling a car dealership fairly straightforward as we already have some idea of how it operates which helped us to visualize and give emphasis to the design of our database. Our database allows the user to search for a vehicle within inventory based on specific features of interest such as color, make, model, gas mileage, etc. Also, the user can track all services performed on any customers' vehicle as well as the technician who worked on the vehicle. Additionally, the user can track all sales of vehicles for all customers and the salesperson that sold the vehicle. With our help the car dealership will now have an organized way to keep up with their inventory, service, sales and employees for all locations. An important aspect of our database is the ISA relationship between employee and the manager, technician and salesperson. This special relationship allows us to track all the activities for each employee function.

Requirement Analysis

This section is regarding the database requirement analysis. Describe the requirements one by one. Additionally, include specific constraints related to your database. Another important part is the functional requirements. State at least 10 functional requirements (including insertions and queries) of your database. Note that, these functional requirements should be sophisticated (inserting a tuple in your database does not need to be reported as a functional requirement).

Data Requirements

- An EMPLOYEE is uniquely identified by their SSN. The dealership also stores each employee's name (first name, last name), address (street, city, state, zip code), birthdate, and phone number. SALESPEOPLE, BRANCH MANAGERs, and TECHNICIANs are EMPLOYEEs.
- Each BRANCH MANAGER receives a salary and a bonus, each SALESPERSON receives a salary and a commission, and each TECHNICIAN receives an hourly wage.
- Each DEALERSHIP BRANCH is identified by its branch id. Each DEALERSHIP BRANCH also has an address and a phone number.
- All INVENTORIEs are identified by their inventory id which is unique for its DEALERSHIP BRANCH. INVENTORY has a derived attribute holding the number of VEHICLEs in its possession.
- A MANUFACTURER is uniquely identified by its name.
- A VEHICLE is uniquely identified by its vehicle id (V_ID). A vehicle has a color,mileage, and model id referring to its model.
- A Model is uniquely identified by its model_id. A model has a certain horsepower, mpg, make, type, MSRP, and Manufacturer Name.
- A CUSTOMER is uniquely identified by their customer id. The database also stores a CUSTOMER's name (first and last), phone number, birthdate and address (street, city, state, zip code).
- A SERVICE is uniquely identified by its service id. A SERVICE has an hourly cost and a type.
- A RECEIPT is uniquely identified by its receipt id.
- All PARTs have a unique id number and a name.
- SUPPLIERs have a unique supplier id, name, phone number, and address (street, city, state, zip code).
- Our database has a DEALERSHIP BRANCH entity which will represent each open location of our client. Each DEALERSHIP BRANCH must have 1 BRANCH MANAGER and 1 or more EMPLOYEEs.

- A BRANCH MANAGER manages a DEALERSHIP BRANCH. A BRANCH MANAGER can manage only 1 DEALERSHIP BRANCH at a time. The start date of when a manager began working is stored.
- Each DEALERSHIP BRANCH has multiple INVENTORIES which is where they store the data on the number of unsold VEHICLES in their possession. An INVENTORY has only 1 associated DEALERSHIP BRANCH.
- VEHICLEs are sold to a CUSTOMER with the help of a SALESPERSON for a specific price, on a certain date, with an invoice number. Each individual purchase involves 1 CUSTOMER, 1 SALESPERSON, and 1 VEHICLE. After purchase, a CUSTOMER owns a VEHICLE. Every VEHICLE can only have 1 CUSTOMER at most, but a CUSTOMER can own many VEHICLEs.
- VEHICLEs are stored in the inventory. Even after a vehicle is SOLD its original inventory is kept track of to keep track of the performance of inventories.
- A CUSTOMER can bring a broken VEHICLE to receive a SERVICEs to be done by a TECHNICIAN. The date this transaction happened is stored and the CUSTOMER will receive a RECEIPT indicating all the SERVICEs they receives on that day
- Some SERVICEs require PARTs. The quantity of each PART needed is stored. PARTs are bought from a SUPPLIER at a certain price.

Functional Requirements

- Customers can query all the services they have received on each of their vehicles, and how much this cost them.
- A branch manager can query how much revenue each salesperson has generated for the dealership, and decide which salespeople deserve promotions and raises.
- A technician will be able to query all of the services they may have to do, look at how long it takes to complete the service, and find all the parts and their cost for the service.
- A branch manager will be able to query the number of vehicles in their inventory, allowing them to plan whether or not they should make another purchase.
- The manager can view the performance of each technician by looking at how many services each technician has completed.
- Salespeople will be responsible for adding in transactions for customers and altering them/ deleting them if an error has been made or some other problem occurred.
- Customers will be able to filter available cars online by any of the vehicle’s or model’s attributes.
- A Salesperson will be able to query all of their sales, allowing them to calculate the commission they have earned, and see their performance over time.

- A branch manager can query which models of cars are being sold and which are not, allowing them to decide which purchases to make from manufacturers and which purchases not to make.
- The dealership company will be able to compare the performance between branches, comparing revenues based on sales to see which managers are performing poorly and which managers need to be replaced.

Conceptual Design

Entities and Attributes

Entity 1: DEALERSHIP BRANCH

Entity: A particular branch of a dealership that has employees, customers, and an inventory of vehicles.

Attributes

- B_ID: A unique identification code for a particular dealership branch.
- Phone_num: A particular dealership branch's phone number.
- Address(Street, City, State, Zipcode): Describes the street, city, state, and zip code where a particular branch is located.
- Mgr_ESSN: Identifies the branch manager of a particular branch.

Relationships

- WORKS_AT: A dealership branch has one or more employees who *work at* its location.
- MANAGES: A Branch manager *manages* a dealership branch. We keep track of the start date of a branch manager who manages a dealership branch.
- HAS_INVENTORY: Every dealership branch *has an inventory*.

Primary Key

- Branch_ID uniquely identifies a particular dealership branch instance. Semantically, it makes sense that a branch is identified by a unique ID. Designating an address or phone number as a branch's primary key doesn't convey the same context.

Entity 2: EMPLOYEE

Entity: Represents an Employee who works at a dealership branch.

Attributes

- ESSN: An employee's social security number.
- Name(FName, LName): Describes the first and last name of an employee.
- Address(Street, City, State, Zipcode): Describes the street, city, state, and zip code where an employee lives.
- Birthdate: Describes the date of birth of an employee.
- Phone_No: Describes an employee's phone number.
- B_ID: Identifies which branch the employee works at.

Relationships

- WORKS_AT: every EMPLOYEE *works at* one DEALERSHIP BRANCH, but a dealership branch can have many employees.
- ISA: BRANCH MANAGERS, SALESPERSON, and TECHNICIANS are all EMPLOYEEs, but an employee can be only one of those at a time.

Primary Key

- ESSN, an employee's social security number, uniquely identifies a particular employee because more than one employee could have the same, address, birth date, and/or phone number, which would not allow users to identify a unique employee.

Entity 3: BRANCH MANAGER

Entity: A BRANCH MANAGER *isa* employee that manages a DEALERSHIP BRANCH

Attributes

- Salary: The salary of a branch manager.
- Inherits from EMPLOYEE: ESSN, Name, Birthdate, Phone_No, and Address.
- Start Date: The start date of the branch manager managing a dealership branch.

Relationships

- ISA: A BRANCH MANAGER *isa* EMPLOYEE.
- MANAGES: A BRANCH MANAGER must *manage* a DEALERSHIP BRANCH and a dealership must be managed by one branch manager. We keep track of the branch managers' start date of managing a branch.

Primary Key

- ESSN, an employee's social security number, uniquely identifies a particular employee because more than one employee could have the same, address, birth date, and/or phone number, which would not allow users to identify a unique employee.

Entity 4: SALESPERSON

Entity: A SALESPERSON *isa* EMPLOYEE who sells VEHICLES to CUSTOMERS.

Attributes

- Salary: The salary of a salesperson.
- Work Phone_No: A salesperson has a direct line of contact to better facilitate business with customers.
- Inherits from EMPLOYEE: ESSN, Name, Birthdate, Phone_No, and Address.
- Commission: A salesperson earns a commission from every vehicle he sells.

Relationships

- ISA: A SALESPERSON *isa* EMPLOYEE.
- SOLD: A VEHICLE is *sold* by a SALESPERSON to a CUSTOMER. We keep track of the sales date, invoice number, and price of the sale.

Primary Key

- ESSN: an employee's social security number, uniquely identifies a particular employee because more than one employee could have the same address, birthdate, and/or phone number, which would not allow users to identify a unique employee.

Entity 5: TECHNICIAN

Entity: A TECHNICIAN *isa* EMPLOYEE who services a CUSTOMER's VEHICLE.

Attributes

- Hourly Wage: Technicians are hourly wage employees.
- Inherits from EMPLOYEE: ESSN, Name, Birthdate, Phone_No, and Address.

Relationships

- ISA: A TECHNICIAN *isa* EMPLOYEE.
- REQ_SERVICE: A CUSTOMER requests a SERVICE for their VEHICLE which is then performed by a TECHNICIAN. We keep track of the date of the service request.

Primary Key

- ESSN, an employee's social security number, uniquely identifies a particular employee because more than one employee could have the same, address, birth date, and/or phone number, which would not allow users to identify a unique employee.

Entity 6: SERVICE

Entity: A SERVICE represents the work performed on a VEHICLE.

Attributes

- Service_ID: Identifies the type of service.
- Type: The work performed on a vehicle.
- Cost: The monetary cost for the service performed.
- Duration: Describes how long a service took.

Relationships

- REQ_SERVICE: A CUSTOMER *requests service* for their VEHICLE which is fulfilled by a TECHNICIAN performing a SERVICE on the vehicle.

- REQ_PART: every SERVICE will require one or more part(s). And every part can belong to one or more service(s).
- HAS_SERVICES: every RECEIPT *has services* and every SERVICE can belong to one or more receipt(s).

Primary Key

- Service_ID: Uniquely identifies each type of service.

Entity 7: PART

Entity: PART represents a vehicle part that may be needed for to perform a service.

Attributes

- Part_ID: Every part has a unique ID code.
- Name: A part's name e.g carburetor.

Relationships

- SUPPLIED_BY: PARTs are *supplied by* a SUPPLIER. We keep track of the price of the part supplied by the supplier.
- REQ_PART: Every SERVICE *requires* one or more PART(s), and every PART can be used in one or more SERVICE.

Primary Key

- Part_ID: There will be cases where more than one PART shares the same name and so to identify that particular PART, we would require each part to have a unique PART_ID.

Entity 8: SUPPLIER

Entity: A SUPPLIER supplies parts that may be needed for vehicle services.

Attributes

- SP_ID: A supplier's unique identification code.
- Name: A supplier's name.
- Phone_No: A supplier's phone number.
- Address(Street, City, State, Zipcode): Describes the street, city, state, and zip code where a supplier is located.

Relationships

- SUPPLIED_BY: PARTS are *supplied by* a SUPPLIER, we keep track of the price of the part supplied.

Primary Key

- SP_ID: Suppliers may have the same name, which alone would not uniquely identify a particular supplier. Extreme cases where suppliers with the same name swap addresses and phone numbers can also hinder us from properly identifying a particular supplier, but having a unique supplier ID would resolve this.

Entity 9: CUSTOMER

Entity: A CUSTOMER can purchase a VEHICLE from a dealership and/or have their vehicle serviced at the dealership. A customer need not own a vehicle specifically purchased at the dealership.

Attributes

- C_ID: A unique identification code that allows a dealership to identify a customer.
- Name(FName, LName): Describes the first and last name of a customer.
- Birthdate: Describes the date of birth of a customer.
- Address(Street, City, State, Zip code: Describes the street, city, state, and zip code where an employee lives.

Relationships

- OWNS: A CUSTOMER can OWN many VEHICLEs, but a vehicle can belong to only one customer.
- SOLD: A VEHICLE is *sold* by a SALESPERSON to a CUSTOMER. We keep track of the sales date, Invoice_No, and price of sale.
- REQ_SERVICE: A customer can *request service* for their vehicle, which will be fulfilled by a technician. We keep track of the service request date.

Primary Key

- C_ID: It's possible more than one customer can have the same name, phone number, birth date, and address so a unique customer identification code is necessary to uniquely identify a specific one.

Entity 10: INVENTORY

Entity: An INVENTORY encompasses the vehicle stock of a dealership branch.

Attributes

- INV_ID: The identification code for an inventory.

Relationships

- HAS_INVETORY: Every DEALERSHIP BRANCH must have an INVENTORY and every inventory must belong to a dealership branch.
- INV_CONTAINS: Every INVENTORY must contain one or more VEHICLE(S).

Primary Key

- (BRANCH_ID, INV_ID): An inventory's ID is unique for its particular branch only. It's possible that two inventories have the same id, thus we need a combination of its respective branch_id and its own inventory_id to uniquely identify a particular inventory

Entity 11: MANUFACTURER

Entity: MANUFACTURERS make MODELS.

Attributes

- Name: The name of a particular Manufacturer.

Relationships

- MADE_BY: Every Model must be made by a manufacturer.

Primary Key

- M_name: Modeled after the real world, manufacturer names are inherently unique.

Entity 12: MODEL

Entity: Every VEHICLE is a MODEL and every model is MADE BY a MANUFACTURER.

Attributes

- M_ID: A model identification model used to uniquely identify a particular model.
- Make: Describes the make of a model e.g Accord, Camry, Tundra, etc.
- Year: The year the vehicle was produced.
- Type: Describes the vehicle type e.g Coupe, Sedan, Jeep, Truck, etc.
- HP: The horsepower of a vehicle.
- MPG: Describes the distance in miles a vehicle can travel using one gallon of fuel.
- MSRP: The Manufacturer Suggested Retail Price. A "recommended" selling price for a vehicle.
- Man_name: The name of the manufacturer that made this model.

Relationships

- MADE_BY: Each Model must be made one manufacturer.
- IS_MODEL: Every Vehicle *is* a *model*.

Primary Key

- M_ID: Every model will have a unique model ID number.

Entity 13: VEHICLE

Entity: Vehicles can be contained in a branch inventory, sold to customers, owned by a customer, and serviced by technicians.

Attributes

- V_ID: A vehicle identification number used to uniquely identify a particular vehicle.
- Model: Describes the model of a vehicle e.g Accord, Camry, Tundra, etc.
- Color: The color of a vehicle.
- Curr_mileage: The current mileage a vehicle has accrued.
- Year: The year the vehicle was produced.
- MSRP: The Manufacturer Suggested Retail Price. A "recommended" selling price for a vehicle.
- Type: Describes the vehicle type e.g Coupe, Sedan, Jeep, Truck, etc.

Relationships

- IS_MODEL: Every VEHICLE is a MODEL.
- REQ_SERVICE: A CUSTOMER can *request service* for their vehicle, the SERVICE is fulfilled by a TECHNICIAN.
- OWNS: A CUSTOMER can own multiple VEHICLES but each vehicle belongs to, at most, one customer.
- SOLD: A VEHICLE is *sold* to a CUSTOMER by a SALESPERSON. We keep track of the sales date, invoice_no, and selling price.
- INV_CONTAINS: An INVENTORY contains certain vehicles.

Primary Key

- V_ID It is possible for multiple vehicles to have the same values for all of their respective attributes, so assigning a unique V_ID to every vehicle allows us to identify a particular vehicle.

Entity 14: RECEIPT

Entity: Every instance of a customer requesting a service for their vehicle will generate a receipt that contains the services performed.

Attributes

- Receipt_ID: A positive integer that tracks every instance of a customer requesting a service for their vehicle. Automatically incremented.

Relationships

- REQ_SERVICE: Every instance of a customer requesting a service for their vehicle will generate a receipt that contains the services performed.
- HAS_SERVICES: Every receipt will have one or more services listed on it.

Primary Key

- Receipt_ID: A positive integer that tracks every instance of a customer requesting a service for their vehicle. Automatically incremented.

Relationships

Relationship 1: WORKS_AT

Relation: Shows the relation between Employee and Dealership Branch

Attributes

ESSN(Primary Key of Employee): An employee's social security number.

Name(FName, LName): Describes the first and last name of an employee.

Address(Street, City, State, Zipcode): Describes the street, city, state, and zip code where an employee lives.

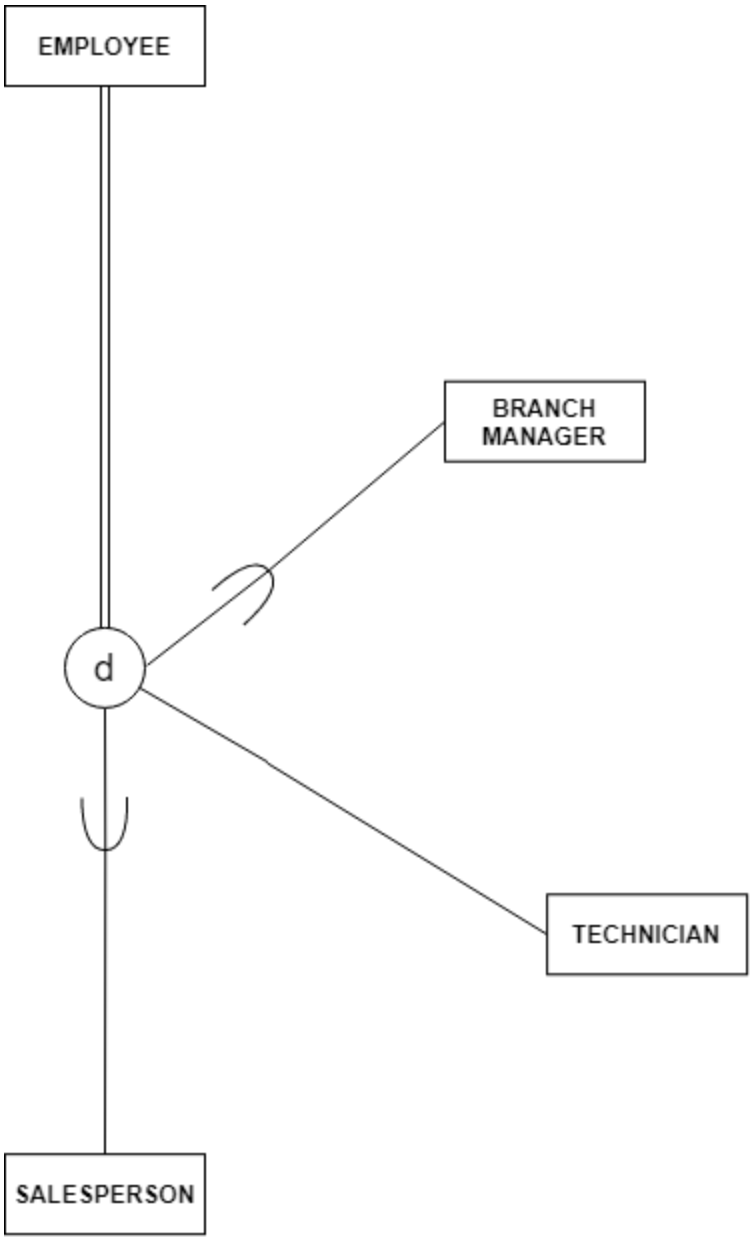
Birthdate(Employee Attribute): Describes the date of birth of an employee.

Phone_No(Employee lAttribute): Describes an employee's phone number.

B_ID(Foreign Key from Dealership_Branch): Identifies which branch the employee works at.

Cardinalities: Every employee must work at a dealership branch, and every dealership branch must have at least 1 employee but can have more.

Relationship 2: EMPLOYEE SPECIALIZATION



Relation: Shows the relation between Employee, Branch Manager, Technician and Salesperson

Attributes

ESSN (Foreign Key from Employee): Uniquely identifies employees by their social security number.

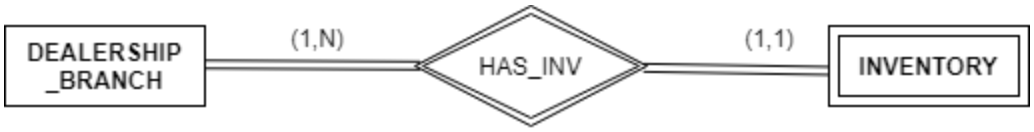
Branch Manager (Subclass of Employee)

Technician (Subclass of Employee)

Salesperson (Subclass of Employee)

Cardinalities:Every employee must participate in one of its subclasses. The subclasses inherit all of the superclass's (Employee) attributes as well as any relationship the superclass participates in e.g. WORKS_AT.

Relationship 3: HAS_INVENTORY



Relation: Shows the relation between Inventory and Dealership Branch

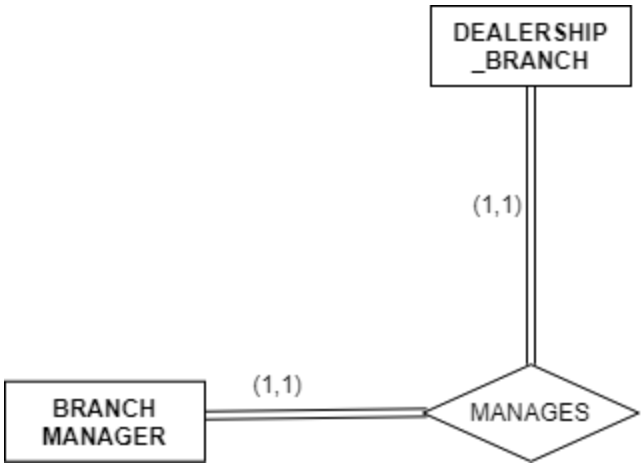
Attributes

INV_ID (Foreign Key from Inventory) – Uniquely identifies the inventory at each branch location.

B_ID (Foreign Key from Dealership Branch): Uniquely identifies which branch in which an employee works.

Cardinalities:Every dealership must have at least one inventory and each inventory must belong to a dealership branch.

Relationship 4: MANAGES



Relation: Shows the relation between Branch Manager and Dealership Branch

Attributes

B_ID(Primary Key of Dealership_Branch): A unique identification code for a particular dealership branch.

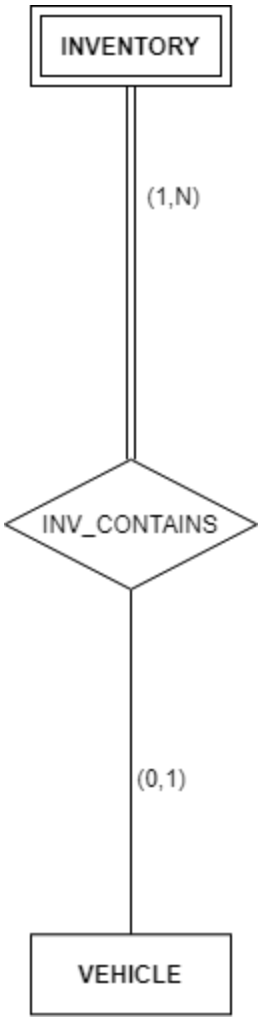
Phone_num(Dealership_Branch Attribute): A particular dealership branch's phone number.

Address(Street, City, State, Zipcode): Describes the street, city, state, and zip code where a particular branch is located.

Mgr_ESSN(Foreign Key from Branch Manager): Identifies the branch manager of a particular branch.

Cardinalities: Each Branch Manager must manage a Dealership Branch and each branch must be managed by a branch manager..

Relationship 5: INV_CONTAINS



Relation: Each Dealership branch can have one or more inventory that contains vehicles.

Attributes

INV_ID (Foreign Key from Inventory): Uniquely identifies inventory at each branch location.

B_ID (Foreign Key from Dealership_Branch): Uniquely identifies a particular branch.

V_ID (Foreign Key from Vehicle): Uniquely identifies each vehicle.

Cardinalities: Each inventory must have at least one vehicle, but not every vehicle needs to be in an inventory.

Relationship 6: MADE_BY

Relation: Every MODEL is MADE_BY at most one MANUFACTURER.

Attributes

M_ID (Primary Key of Model): Uniquely identifies each model.

Make(Model Attribute): Describes the make of the model

Year(Model Attribute):: Describes the year the model was made by the manufacturer.

Type(Model Attribute):: Describes the type of the model e.g Truck, sedan, coupe.

HP(Model Attribute):: Describes the horsepower of the vehicle.

MPG(Model Attribute):: Describes the miles the model can travel using one gallon of gas.

MSRP(Model Attribute):: Describes the manufacturer suggested retail price.

Man_name(Foreign Key from Manufacturer): The name of the Manufacturer that made this model..

Cardinalities: Each Model must be made by a Manufacturer but every Manufacturer can make many models (but must make at least one).

Relationship 7: IS_MODEL

Relation: Every vehicle is a model.

Attributes

V_ID(Primary key of Vehicle): Uniquely identifies a vehicle.

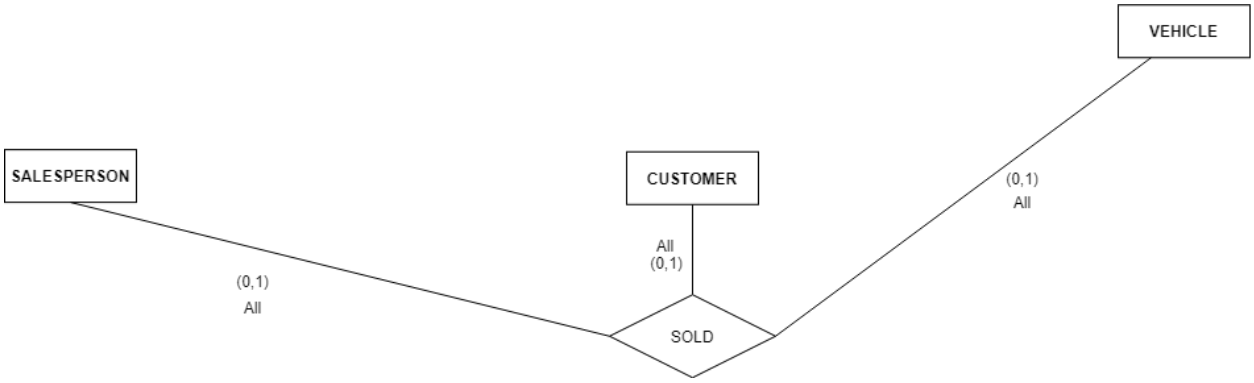
Color(Vehicle Attribute): Describes the color of a vehicle.

Curr_mileage(Vehicle Attribute): Describes the current mileage of a vehicle.

M_ID(Foreign Key from Model): The model of a particular vehicle, describing certain features and attributes of said vehicle.

Cardinalities: Every Vehicle must be a Model, but a Model can be associated with many vehicles..

Relationship 8: SOLD



Relation: Shows the relation between Customer, Vehicle and Salesperson

Attributes

C_ID (Foreign Key from Customer) – Uniquely identifies each customer.

V_ID(Foreign Key from Vehicle) - Uniquely identifies each vehicle.

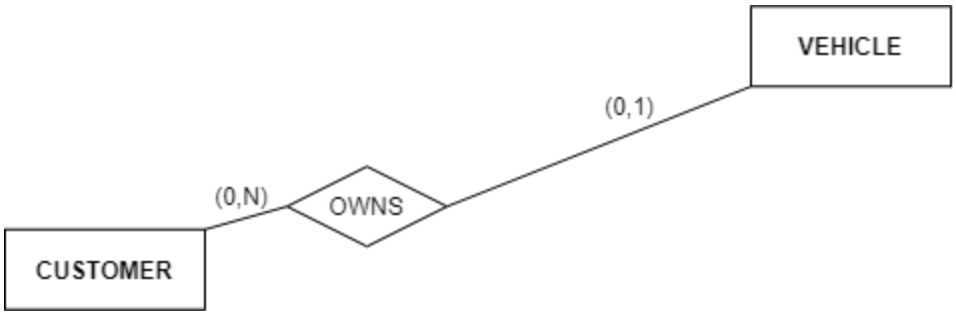
ESSN (Foreign key from Salesperson) – Uniquely identifies a salesperson by their social security number.

Price(Relational Attribute): The price that a vehicle was sold to a customer.

Date(Relational Attribute): The date of the sale.

Cardinalities: Each instance of a sale involves one customer, one salesperson, and one vehicle.

Relationship 9: OWNS



Relation: Shows the relation between Customer and Vehicle

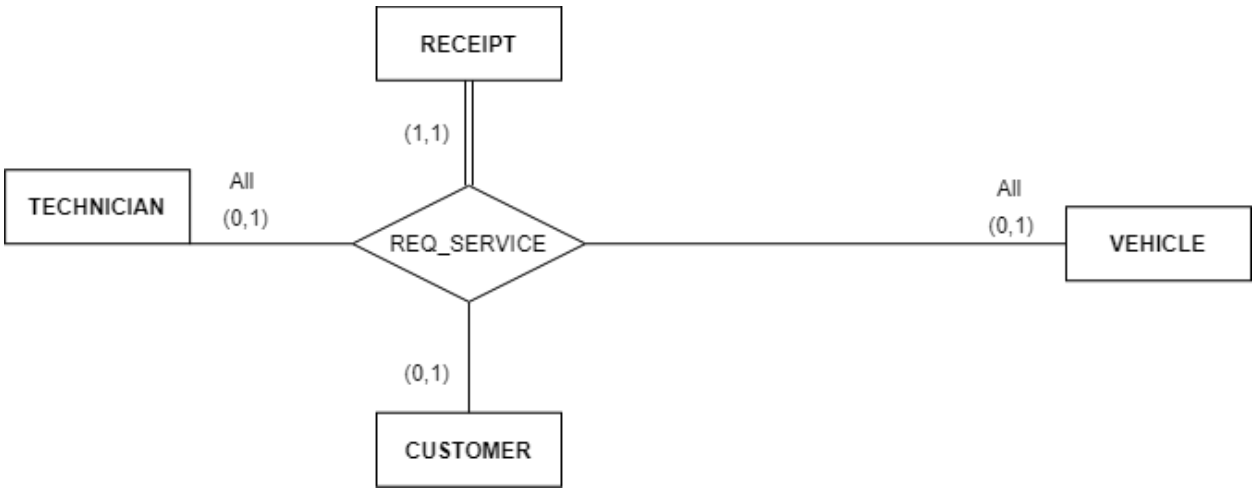
Attributes

C_ID (Foreign Key from Customer) – Uniquely identifies each customer.

V_ID (Foreign Key from Vehicle) - Uniquely identifies each vehicle.

Cardinalities: A customer can own many vehicles, but each vehicle can belong to only one customer.

Relationship 10: REQ_SERVICE



Relation: Shows the relation between Customer, Vehicle, Technician and Receipt

Attributes

C_ID (Foreign Key from Customer) – Uniquely identifies each customer.

V_ID (Foreign Key from Vehicle) - Uniquely identifies each vehicle.

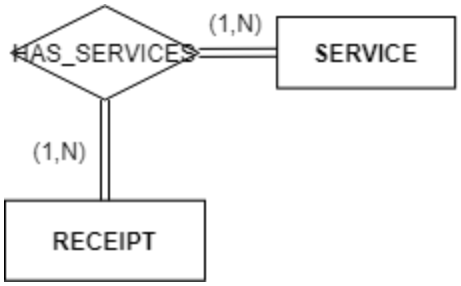
ESSN (FK from Technician) - Uniquely identifies a technician by their social security number.

Receipt_ID (Foreign Key from Receipt) – Will list the services performed for each instance of this relationship.

Date (Relational Attribute) – Identifies the date of the service request.

Cardinalities: For every time a customer goes to receive services on his car, they will receive 1 receipt, and the services will be performed by 1 technician.

Relationship 11: HAS_SERVICES



Relation: Each receipt has at least one or more services listed on it and every service can belong to one or more receipt.

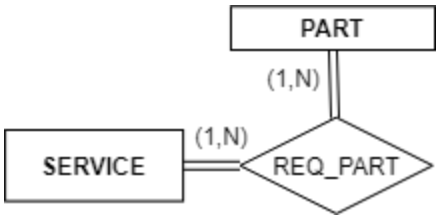
Attributes

Receipt_ID(Foreign Key from Inventory): Uniquely identifies which receipt has these services..

Service_ID (Foreign Key from Service): Uniquely identifies the service(s) each receipt has.

Cardinalities: Each receipt has at least one or more services listed on it and each service can belong to one or more receipt.

Relationship 12: REQ_PART



Relation: Shows the relation between Service and Parts

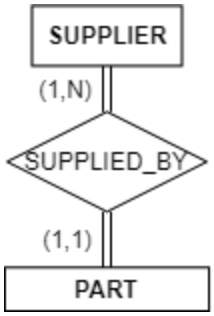
Attributes

Service_ID (Foreign Key from Service) – Identifies the service that will require parts.

Part_ID (Foreign Key from Parts) - Uniquely identifies each part.

Cardinalities: Every service requires at least one part and each part will belong to one or more services.

Relationship 13: SUPPLIED_BY



Relation: Shows the relation between Part and Supplier

Attributes

Part_ID (Foreign Key from Parts) - Uniquely identifies each part.

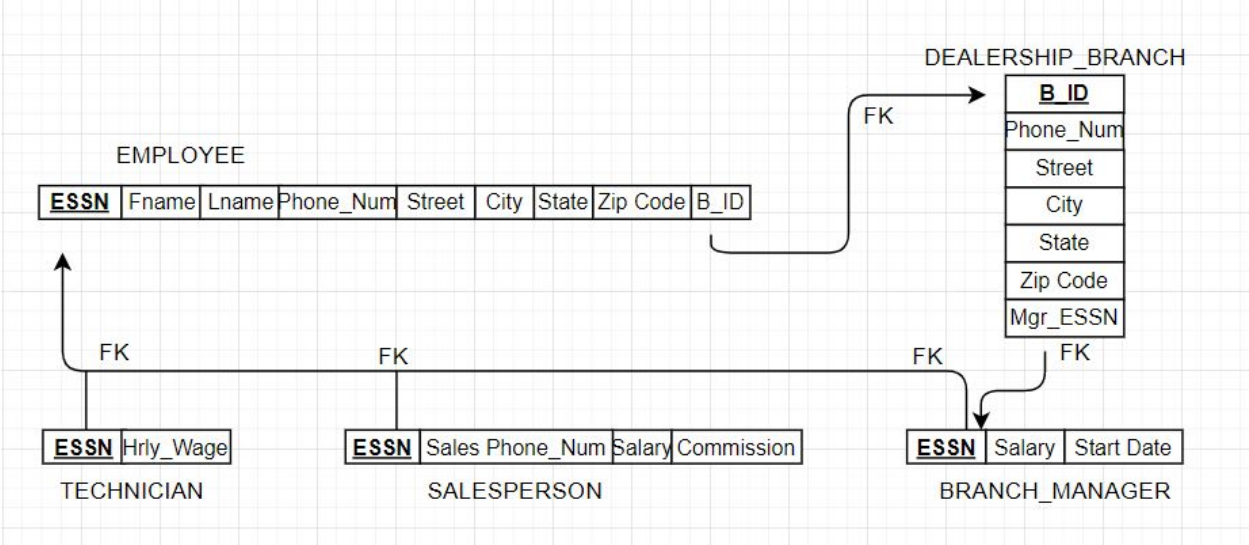
SP_ID (Foreign Key from Parts) - Uniquely identifies each supplier.

Price(Relational attribute) - The price of the Part supplied by the supplier.

Cardinalities: Every part must be supplied by a supplier but every supplier can supply one or more part(s).

Relational Model

Employees working at a branch and the ISA relationships

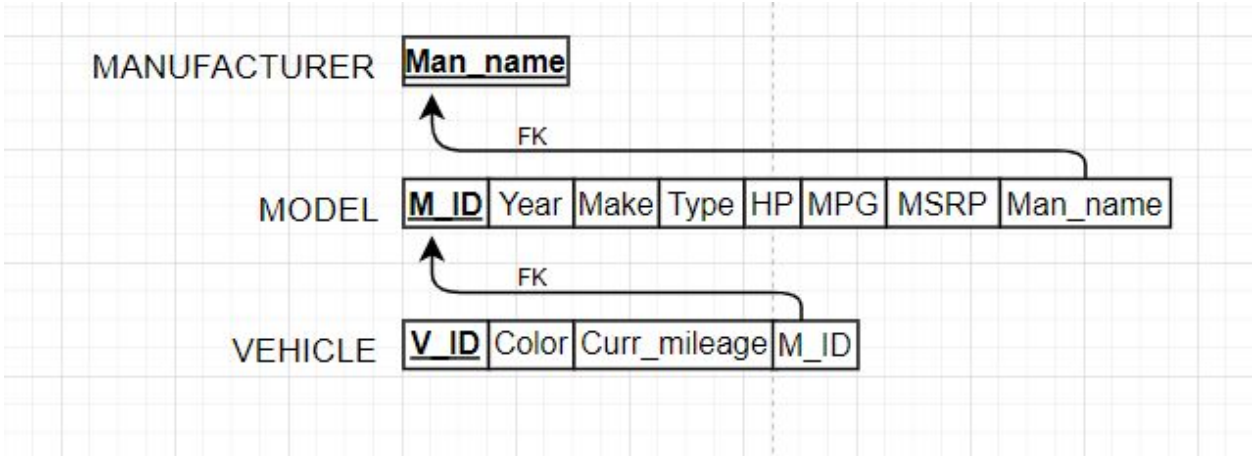


These relations show our EMPLOYEE, SALESPERSON, TECHNICIAN, BRANCH_MANAGER, and DEALERSHIP_BRANCH entities along with the ISA relationship between employees and the different categories of employees they can be, and the WORKS_AT relationship between EMPLOYEE and DEALERSHIP_BRANCH.

Modeling the entities was easy, as we first simply added all the attributes to each relation. We modeled the ISA relationship by having each category's primary key be the ESSN, so if more information was needed about a specific employee, they could be joined with the employee table on their ESSN. Then, for the WORKS_AT relationship, since the cardinality was 1:N, we gave a B_ID foreign key from EMPLOYEE to DEALERSHIP_BRANCH.

All of these relations are in BCNF as the primary key determines all of the non prime attributes, and there are no other functional dependencies present.

Manufacturer, Model, Vehicle

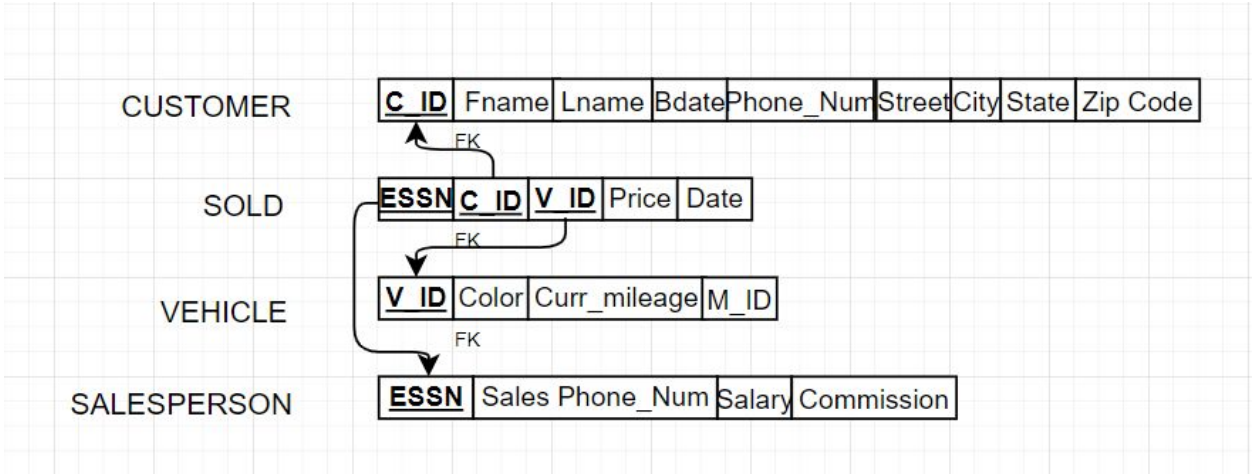


These relations show the MANUFACTURER, MODEL, and VEHICLE entities and how we modeled the IS_MODEL and MADE_BY relationships. In modeling the entities, it was really easy for us since we did not have any multi-valued or composite attributes, so we easily added all the non-prime attributes to each entity relation.

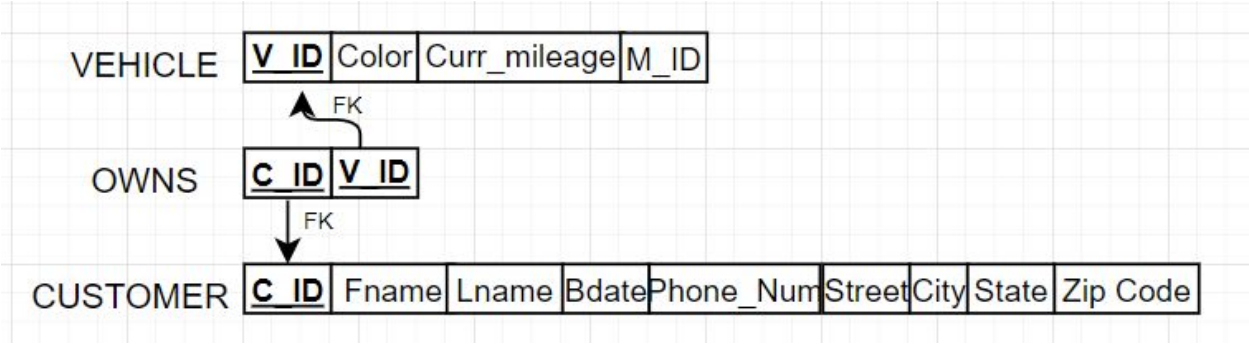
When we were mapping the relationships between these entities, since the cardinality of VEHICLE to MODEL was 1:N and the cardinality of MODEL to MANUFACTURER was also 1:N, we decided to simply add a foreign key M_ID to VEHICLE and Man_name to MODEL, as this would allow us to connect the tables together without losing any data.

All of these relations are in BCNF, as the only functional dependencies in these tables are from the primary key to the non-prime attributes.

Salespeople selling Vehicles to Customers



Customers owning Vehicles

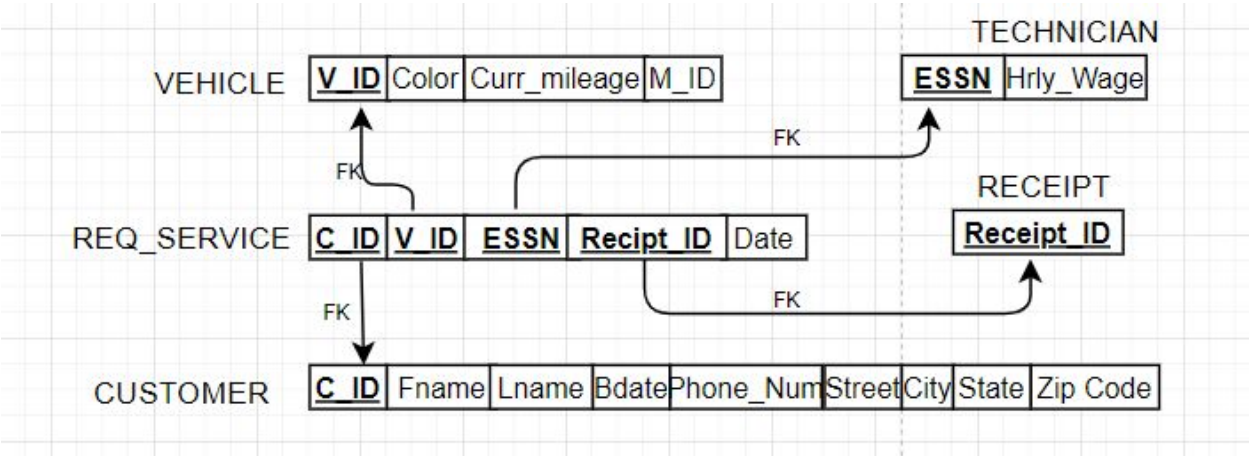


These relations show our VEHICLE and CUSTOMER entities along with the OWNS relationship.
The CUSTOMER entity was very straightforward to make, as we simply took all the attributes on the entity from our ER diagram and added them as attributes on the relation.

The cardinality of CUSTOMER to VEHICLE was 1:N, so we had the option of putting a customer_id foreign key on VEHICLE or creating a separate OWNS relation. We went with the latter because we thought this made the relational model easier to follow. We also chose this because VEHICLE was involved in many 1:N relationships and it would become a very large relation which we wanted to avoid to just keep everything simpler and easier to manage.

These relations are in BCNF as the only functional dependencies present in these tables are from the primary key to all the other attributes.

Customers requesting services from Technician on Vehicles



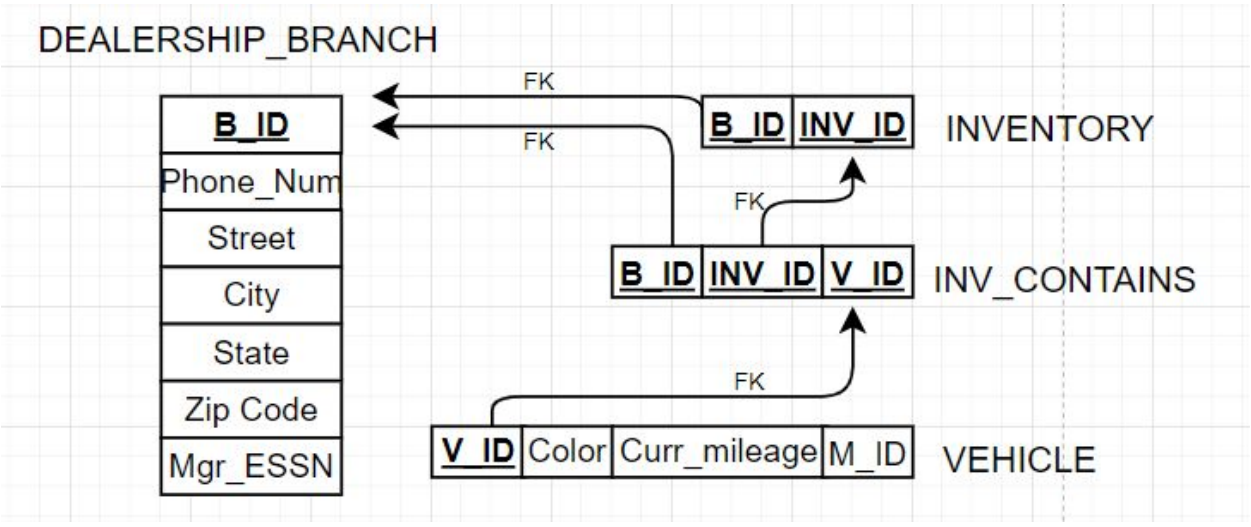
These relations show our VEHICLE, TECHNICIAN, CUSTOMER, and RECEIPT entities along with the REQ_SERVICE relationship.

The TECHNICIAN entity was simply the SSN and the hourly wage as they are a part of the ISA relationship with EMPLOYEE, so all their other information can be found by joining on ESSN with the EMPLOYEE relation. The RECEIPT was simply its ID, as we had already put the date on REQ_SERVICE.

We needed to make REQ_SERVICE its own relation because of the cardinalities since customers could come in and require service on their vehicles many times.

These relations are in BCNF as the only functional dependencies present in these tables are from the primary key to all the other attributes. At first we thought that since every vehicle has one customer, the functional dependency from vehicle to customer would not follow 2NF, but we realized that since vehicles can change owners, if we did not put customer ID in this table and the customer no longer owns the vehicle, they would no longer be able to track how much they paid in services at that time. This means that the customer is not functionally dependent on the vehicle.

Dealership branches and their Inventories of Vehicles

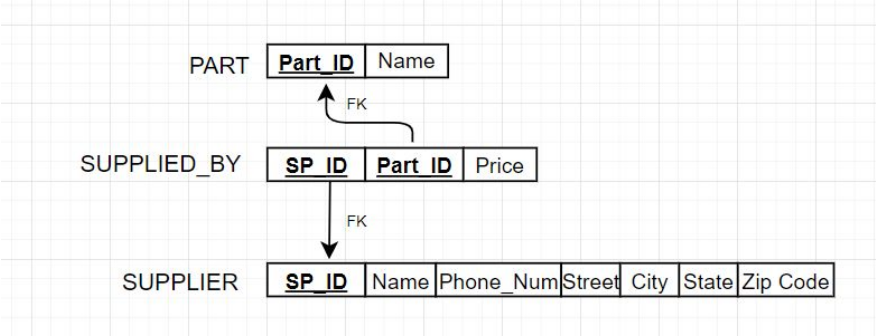


These relations show the DEALERSHIP_BRANCH, INVENTORY, and VEHICLE entities along with the HAS_INV relationship between INVENTORY and DEALERSHIP_BRANCH and the INV_CONTAINS relationship between INVENTORY and VEHICLE.

The DEALERSHIP_BRANCH and VEHICLE entities were already described earlier. For inventory, since it is a weak entity, we made its primary key be the combination of the branch ID and the inventory ID so that we could uniquely identify each inventory. For the INV_CONTAINS relationship we made it its own table as the cardinality was 1:N and so making the relationship its own table was a valid possibility. We also chose this because VEHICLE was involved in many 1:N relationships and it would become a very large relation which we wanted to avoid to just keep everything simpler and easier to manage.

These relations are in BCNF because the only functional dependencies present are from the whole primary key to each of the other attributes (or the trivial primary key determines itself).

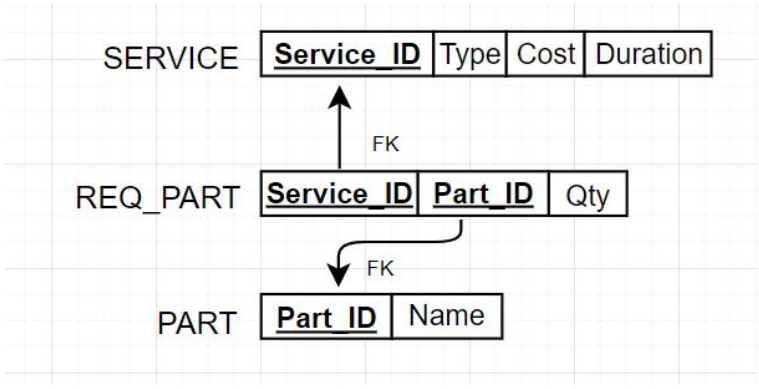
Parts are Supplied-by Suppliers



These relations show our PART and SUPPLIER entities and the SUPPLIED_BY relationship between them. These relations were straightforward to make as we just added the attributes of each entity and relationship to each relation, and we made SUPPLIED_BY its own relation because the cardinality of PART to SUPPLIER was 1:N.

These relations are in BCNF and each of these relations only has the functional dependency of the primary key to the other attributes. We did have multiple candidate keys (Name for PART and SUPPLIER), but we chose to use their respective IDs instead as the primary key.

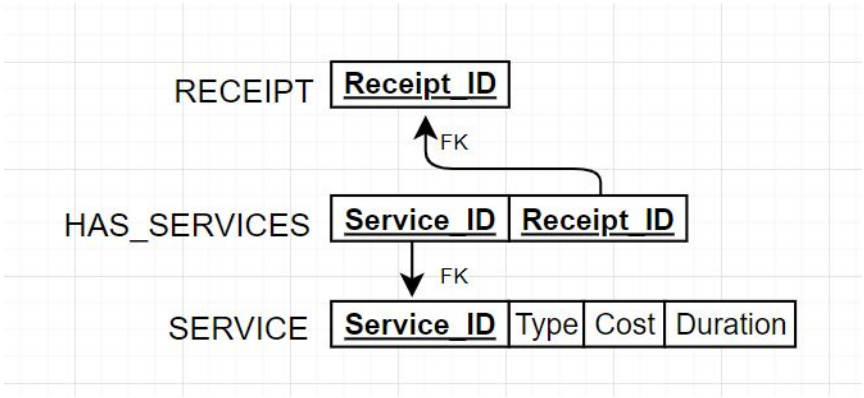
Services Requiring Parts



These relations show the SERVICE and PART entities along with the REQ_PART relationship between them. This was done in a similar way to the SUPPLIER supplying PARTs relations above, the only difference being we had to make REQ_PART its own relation due to the M:N cardinality.

These relations are in BCNF and each of these relations only has the functional dependency of the primary key to the other attributes.

Receipts containing many Services



These relations show how all of our RECEIPTs are composed of multiple SERVICEs. We made HAS_SERVICES its own relation due to the M:N cardinality between RECEIPT and SERVICEs.

These relations are in BCNF and each of these relations only has the functional dependency of the primary key to the other attributes.

Data Dictionary

Table	Attribute	Data Type	Primary Key	Foreign Key	Constraints
BRANCH_MANAGER	ESSN	VARCHAR(11)	YES	EMPLOYEE(ESSN)	11 characters. 3 digits followed by a dash, followed by 2 digits and a dash, and then 4 digits.
BRANCH_MANAGER	Salary	DECIMAL(9,2)			Positive, 2 floating points NOT NULL DEFAULT 75000.00
BRANCH_MANAGER	Start_Date	DATE			< current date
CUSTOMER	C_ID	VARCHAR(10)	YES		Randomly Generated
CUSTOMER	Fname	VARCHAR(25)			
CUSTOMER	Lname	VARCHAR(25)			
CUSTOMER	Birthdate	DATE			<current date
CUSTOMER	Phone-Num	VARCHAR(12)			
CUSTOMER	Street	VARCHAR(50)			
CUSTOMER	City	VARCHAR(50)			
CUSTOMER	State	VARCHAR(15)			
CUSTOMER	Zipcode	VARCHAR(5)			5 digits, positive
DEALERSHIP_BRANCH	B_ID	INT	YES		Positive, Incremented automatically.

DEALERSHIP_BRANCH	Phone-Num	VARCHAR(12)			
DEALERSHIP_BRANCH	Street	VARCHAR(50)			
DEALERSHIP_BRANCH	City	VARCHAR(50)			
DEALERSHIP_BRANCH	State	VARCHAR(15)			
DEALERSHIP_BRANCH	Zipcode	VARCHAR(5)			5 digits, positive
DEALERSHIP_BRANCH	mgr_ESSN	VARCHAR(11)		BRANCH MANAGER(ESSN)	11 characters. 3 digits followed by a dash, followed by 2 digits and a dash, and then 4 digits.
EMPLOYEE	ESSN	VARCHAR(11)	YES		11 characters. 3 digits followed by a dash, followed by 2 digits and a dash, and then 4 digits.
EMPLOYEE	Fname	VARCHAR(25)			
EMPLOYEE	Lname	VARCHAR(25)			
EMPLOYEE	Phone-Num	VARCHAR(12)			XXX-XXX-XXXX
EMPLOYEE	Birthdate	DATE			<current date
EMPLOYEE	Street	VARCHAR(50)			
EMPLOYEE	City	VARCHAR(50)			
EMPLOYEE	State	VARCHAR(15)			

EMPLOYEE	Zipcode	VARCHAR(5)			5 digits, positive
EMPLOYEE	B_ID	INT		DEALERSHIP_BRANCH(B_ID)	
HAS_SERVICES	Receipt_ID	INT	YES	RECEIPT(Receipt_ID)	Positive, Incremented automatically.
HAS_SERVICES	Service_ID	INT	YES	SERVICE(Service_ID)	Positive, Incremented automatically.
INV_CONTAINS	B_ID	INT	YES	DEALERSHIP_BRANCH(B_ID)	Positive, Incremented automatically.
INV_CONTAINS	INV_ID	INT	YES	INVENTORY(INV_ID)	Positive, Incremented automatically.
INV_CONTAINS	V_ID	VARCHAR(7)	YES	VEHICLE(V_ID)	3-7 Alphanumeric Characters
INVENTORY	B_ID	INT	YES	DEALERSHIP_BRANCH(B_ID)	Positive, Incremented automatically.
INVENTORY	INV_ID	INT	YES		Positive, Incremented automatically.
MANUFACTURER	Man_name	VARCHAR(30)	YES		
MODEL	M_ID	INT	YES		Positive, auto-incremented
MODEL	Make	VARCHAR(15)			
MODEL	Year	INT			Positive, >1960
MODEL	Type	VARCHAR(15)			
MODEL	HP	INT			Positive
MODEL	MPG	INT			Positive

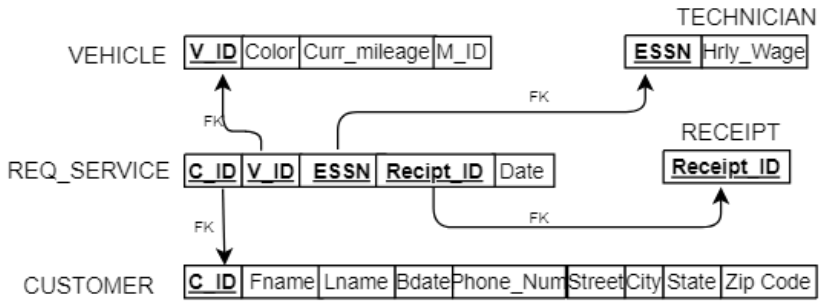
MODEL	MSRP	DECIMAL(8,2)			Positive
MODEL	Man_name	VARCHAR(30)		MANUFACTURER(Man_name)	
OWNS	C_ID	VARCHAR(10)	YES	CUSTOMER(CID)	Randomly Generated
OWNS	V_ID	VARCHAR(7)	YES	VEHICLE(V_ID)	3-7 Alphanumeric Characters
PART	Part_ID	INT	YES		Positive, Auto-incremented
PART	P_name	VARCHAR(30)			
PART	Price	DECIMAL(6,2)			Positive
RECEIPT	Receipt_ID	INT	YES		Positive, Auto-incremented
REQ_PARTS	Service_ID	INT	YES	SERVICE(Service_ID)	Positive, auto-incremented
REQ_PARTS	Part_ID	INT	YES	PART(Part_ID)	Positive, Auto-incremented
REQ_PARTS	Qty	INT			Positive
REQ_SERVICE	C_ID	VARCHAR(10)	YES	CUSTOMER(CID)	Randomly Generated
REQ_SERVICE	V_ID	VARCHAR(7)	YES	VEHICLE(V_ID)	3-7 Alphanumeric Characters
REQ_SERVICE	ESSN	VARCHAR(11)	YES	TECHNICIAN(ESSN)	11 characters. 3 digits followed by a dash, followed by 2 digits and a dash, and then 4 digits.
REQ_SERVICE	Receipt_ID	INT	YES	RECEIPT(Receipt_ID)	Positive, auto-incremented

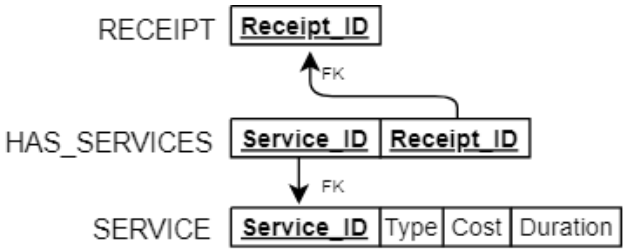
REQ_SERVICE	Date	DATE			
SALESPERSON	ESSN	VARCHAR(11)	YES	EMPLOYEE(ESSN)	11 characters. 3 digits followed by a dash, followed by 2 digits and a dash, and then 4 digits.
SALESPERSON	Sales_Phone_Num	VARCHAR(12)			10 digits, positive
SALESPERSON	Salary	DECIMAL(9,2)			Positive, 2 floating points
SALESPERSON	Commission	DECIMAL(4,2)			Represents the percent of a sale that goes to the salesperson. Positive, NOT NULL Default .25
SERVICE	Service_ID	INT	YES		Positive, Incremented automatically.
SERVICE	Type	VARCHAR(30)			Description of service e.g Oil Change, Brake pad replacement, etc.
SERVICE	Cost	DECIMAL(7,2)			Positive
SERVICE	Duration	INT			Positive. Duration of service in minutes
SOLD	ESSN	VARCHAR(11)	YES	SALESPERSON(ESSN)	11 characters. 3 digits followed by a dash, followed by 2 digits and a dash, and then 4 digits.
SOLD	C_ID	VARCHAR(10)	YES	CUSTOMER(C_ID)	
SOLD	V_ID	VARCHAR(7)	YES	VEHICLE(V_ID)	
SOLD	Price	DECIMAL(9,2)			Positive
SOLD	Date	DATE			

SUPPLIED_BY	SP_ID	INT	YES	SUPPLIER(SP_ID)	Positive, auto-incremented
SUPPLIED_BY	Part_ID	INT	YES	PART(Part_ID)	Positive, Auto-incremented
SUPPLIED_BY	Price	DECIMAL(8,2)			Positive
SUPPLIER	SP_ID	INT	YES		Positive, auto-incremented
SUPPLIER	S_name	VARCHAR(50)			
SUPPLIER	Phone-Num	VARCHAR(12)			
SUPPLIER	Street	VARCHAR(50)			
SUPPLIER	City	VARCHAR(50)			
SUPPLIER	State	VARCHAR(15)			
SUPPLIER	Zipcode	VARCHAR(5)			5 digits, positive
TECHNICIAN	ESSN	VARCHAR(11)	YES	EMPLOYEE(ESSN)	11 characters. 3 digits followed by a dash, followed by 2 digits and a dash, and then 4 digits.
TECHNICIAN	Hourly_Wage	DECIMAL(5,2)			Positive, NOT NULL Default 18.75
VEHICLE	V_ID	VARCHAR(7)	YES		3-7 Alphanumeric Characters
VEHICLE	Color	VARCHAR(15)			
VEHICLE	Curr_mileage	INT			Positive
VEHICLE	M_ID	INT		MODEL(M_ID)	Positive, auto-incremented

Implementation Of Functional Requirements

- Customers can query all the services they have received on each of their vehicles, and how much this cost them.





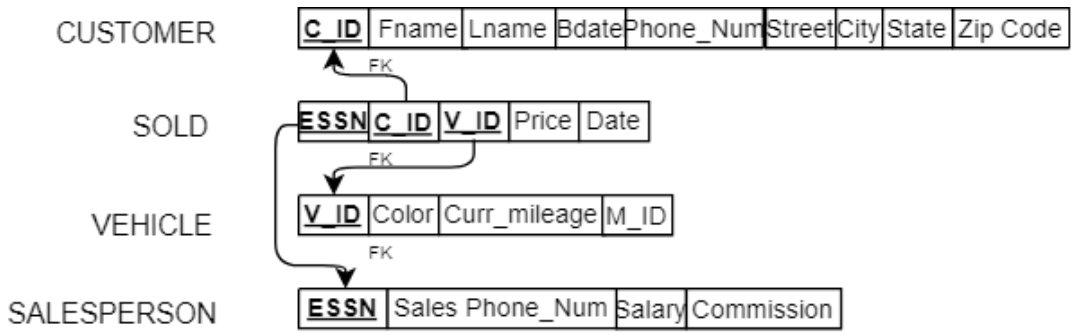
Since each receipt will lists the services performed for that particular service request, the customer can view all the services, as well as their associated costs, performed on their vehicle.

```
SELECT SUM(SERVICE.cost) AS TotalServiceCost
FROM SERVICE
WHERE SERVICE.Service_ID IN (SELECT HAS_SERVICES.Service_ID
                             FROM HAS_SERVICES
                             WHERE HAS_SERVICES.Receipt_ID IN (SELECT REQ_SERVICE.Receipt_ID
                                                               FROM REQ_SERVICE
                                                               WHERE REQ_SERVICE.C_ID = '007' AND REQ_SERVICE.V_ID = '2Q814' ));
```

```
SELECT SERVICE.Type,SERVICE.Cost
FROM SERVICE
WHERE SERVICE.Service_ID IN (SELECT HAS_SERVICES.Service_ID
                             FROM HAS_SERVICES
                             WHERE HAS_SERVICES.Receipt_ID IN (SELECT REQ_SERVICE.Receipt_ID
                                                               FROM REQ_SERVICE
                                                               WHERE REQ_SERVICE.C_ID = '007' AND REQ_SERVICE.V_ID = '2Q814'));
```

Every instance of REQ_SERVICE involves a customer (C_ID) & their vehicle (V_ID), and every service has a cost. So this query sums up the cost of all the services performed on the customer's vehicle through their receipts to find a total cost. The second query lists all of those services names and costs individually.

- A branch manager can query how much revenue each salesperson has generated for the dealership, and decide which salespeople deserve promotions and raises.

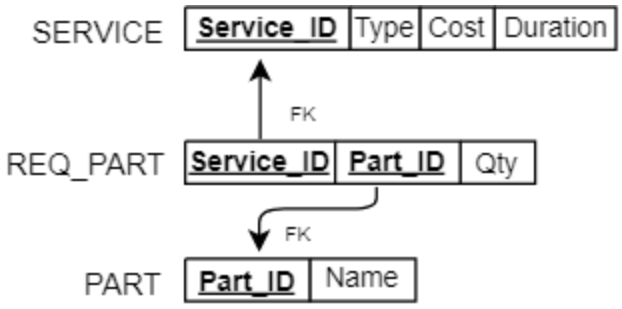


A branch manager can find how much revenue a particular salesperson has generated by querying (from the 'SOLD' table) the sum 'Price" for a particular ESSN (the salesperson).

```
SELECT SUM(SOLD.Price) AS Revenue,EMPLOYEE.ESSN,EMPLOYEE.FName,EMPLOYEE.LName
FROM SOLD,EMPLOYEE,SALESPERSON
WHERE SOLD.ESSN = SALESPERSON.ESSN AND SALESPERSON.ESSN = EMPLOYEE.ESSN
GROUP BY SOLD.ESSN;
```

This query sums the revenues of all salespeople and then joins the salesperson with their employee information to get their names. In order to get all of the salespeople revenues individually, the results are grouped by their SSNs.

- A technician will be able to query all of the services they may have to do, look at how long it takes to complete the service, and find all the parts and their cost for the service.



The SERVICE table provides all the information for every type of service. If the technician is scheduled to perform a service, they can view the type, cost, and duration

for that service, given its Service_ID; and the parts the service may require through the REQ_Part, as well as the Part's name through the PART table, given a particular Part_ID.

```
SELECT SERVICE.Type
FROM SERVICE;
```

```
SELECT SERVICE.Type,Part.P_name,SUPPLIED_BY.Price
FROM PART,REQ_PARTS,SERVICE,SUPPLIED_BY
WHERE REQ_PARTS.Service_ID = SERVICE.Service_ID AND REQ_PARTS.Part_ID = Part.Part_ID
AND PART.Part_ID = SUPPLIED_BY.Part_ID;
```

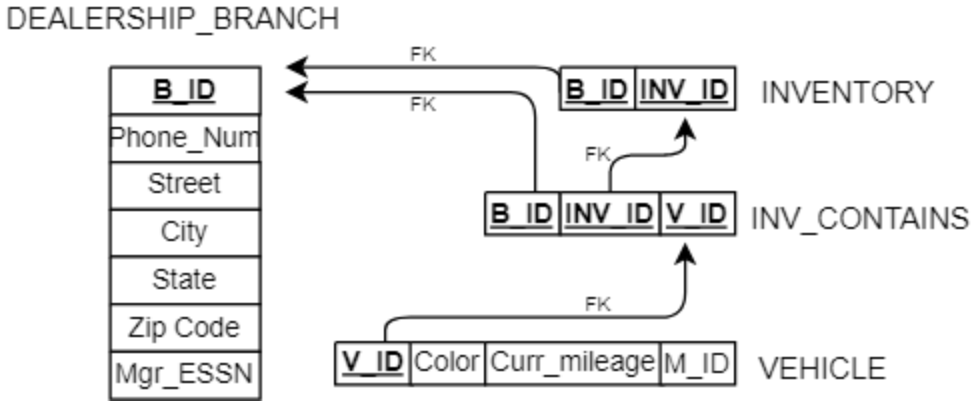
Since PARTs are supplied by SUPPLIERs at a specific price, to get their cost the SUPPLIED_BY relation is also needed.

```
SELECT SERVICE.duration
FROM SERVICE
WHERE SERVICE.Service_ID = 1;

SELECT SERVICE.duration
FROM SERVICE
WHERE SERVICE.Type = 'Serpentine Belt Replacement';
```

Querying the duration is also easy as the SERVICE table holds that information.

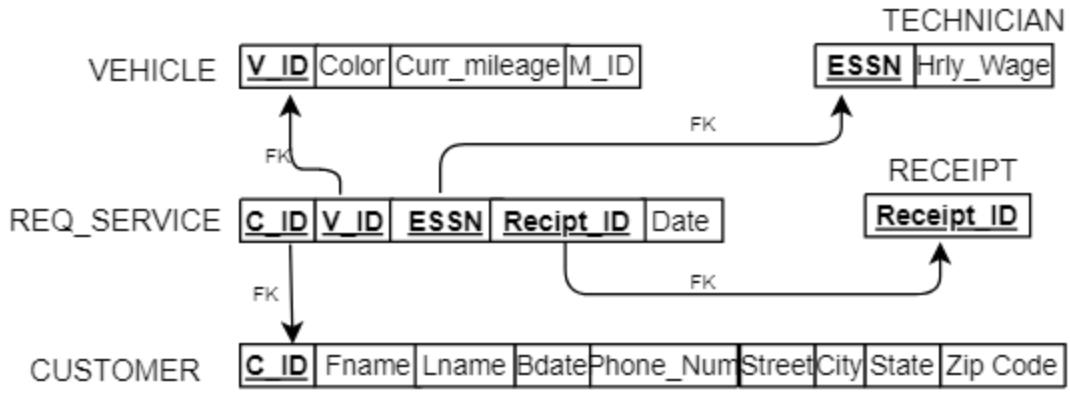
- A branch manager will be able to query the number of vehicles in their inventory, allowing them to plan whether or not they should make another purchase.




```
SELECT COUNT(INV_CONTAINS.V_ID) AS numVehicles
FROM INV_CONTAINS
WHERE INV_CONTAINS.B_ID =2 AND INV_CONTAINS.INV_ID = 2
AND INV_CONTAINS.V_ID NOT IN (SELECT OWNS.V_ID FROM OWNS);
```

The INV_CONTAINS table allows the branch manager to query the number of vehicles in their inventory. They can delve deeper into the specifics of a particular vehicle by querying the associated vehicles V_ID and its model (through M_ID), giving them insight into what type of vehicles are selling or not selling well.

- The manager can view the performance of each technician by looking at how many services each technician has completed.

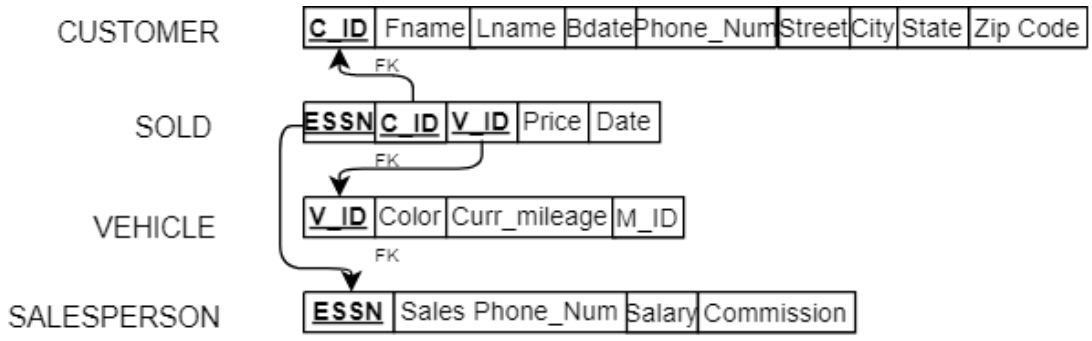


A branch manager can complete this query just by querying a count for a particular technician's ESSN in the REQ_SERVICE table.

```
SELECT COUNT(REQ_SERVICE.ESSN) AS numServices,TECHNICIAN.ESSN,EMPLOYEE.FName,EMPLOYEE.LName
FROM REQ_SERVICE,TECHNICIAN,EMPLOYEE
WHERE REQ_SERVICE.ESSN = TECHNICIAN.ESSN AND TECHNICIAN.ESSN = EMPLOYEE.ESSN AND EMPLOYEE.B_ID = 1
GROUP BY REQ_SERVICE.ESSN;
```

To filter by branch and get the name of the employee, the EMPLOYEE relation is also needed.

- Salespeople will be responsible for adding in transactions for customers and altering them/ deleting them if an error has been made or some other problem occurred.

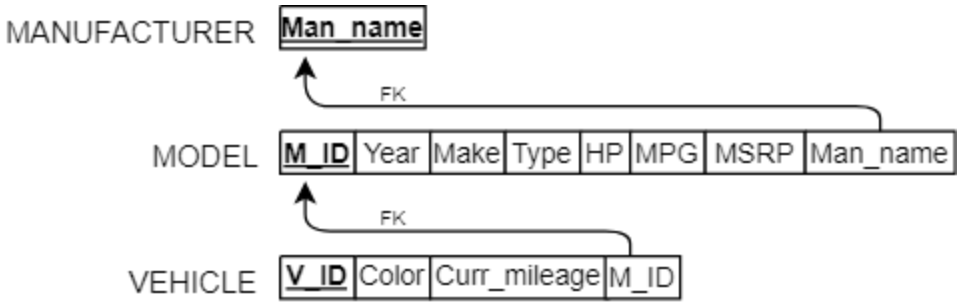


A salesperson just needs their own ESSN, a customer's C_ID, the vehicle's V_ID, the price of the sale, and the date of the sale to add a record to the SOLD table. To alter a record, the salesperson just needs to input their own ESSN, the customer's C_ID, and the V_ID-- from there, they can delete/alter the records information.

```
DELETE
FROM SOLD
WHERE SOLD.C_ID = '419' AND SOLD.ESSN = '427-30-5377' AND SOLD.V_ID = 'XPN';

UPDATE SOLD
SET SOLD.Price = 15000.00
WHERE SOLD.C_ID = '088' AND SOLD.ESSN = '427-30-5377' AND SOLD.V_ID = '6SKW';
```

- Customers will be able to filter available cars online by any of the vehicle's or model's attributes.



Since every vehicle is some type of model, a customer/salesperson just needs to query the vehicles that have that Model's particular M_ID for whatever attribute they're filtering by. The query also has to make sure the vehicle has not yet been sold.

Filtering by MSRP

```
SELECT *
FROM VEHICLE, MODEL
WHERE VEHICLE.M_ID = MODEL.M_ID AND MODEL.MSRP<20000 AND VEHICLE.V_ID NOT IN(SELECT SOLD.V_ID FROM SOLD);
```

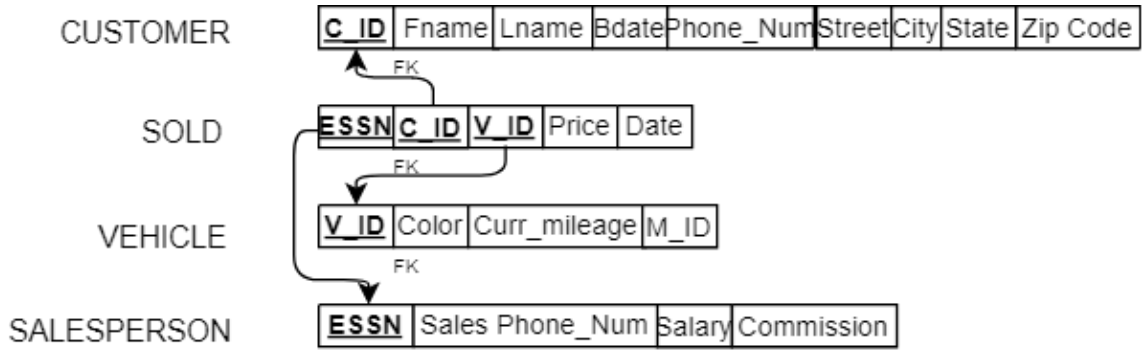
Filtering by Make

```
SELECT *
FROM VEHICLE, MODEL
WHERE VEHICLE.M_ID = MODEL.M_ID AND MODEL.Make='R8' AND VEHICLE.V_ID NOT IN(SELECT SOLD.V_ID FROM SOLD);
```

Filtering by Color

```
SELECT *
FROM VEHICLE, MODEL
WHERE VEHICLE.M_ID = MODEL.M_ID AND VEHICLE.Color = 'Red' AND VEHICLE.V_ID NOT IN(SELECT SOLD.V_ID FROM SOLD);
```

- A Salesperson will be able to query all of their sales, allowing them to calculate the commission they have earned, and see their performance over time.



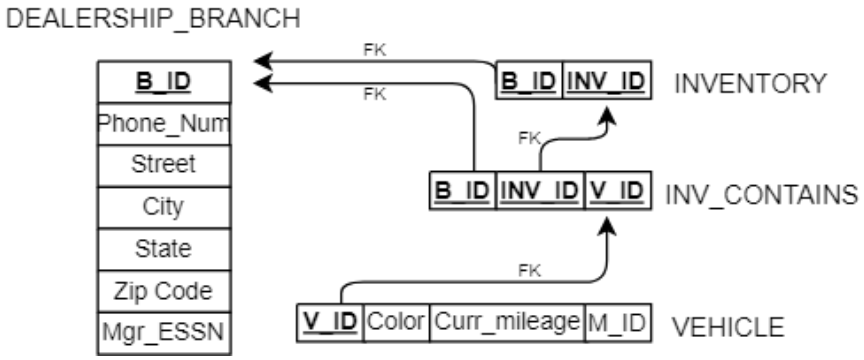
A salesperson just needs to query their ESSN from the SOLD table and SUM (Price * their commission rate) to calculate their total earned commission. They can also set performance ranges by including a date range.

```
SELECT SUM(SALESPERSON.Commission * SOLD.Price) AS CommissionEarned
FROM SOLD,SALESPERSON
WHERE SALESPERSON.ESSN = SOLD.ESSN AND SOLD.ESSN = '468-74-6913';
```

To track their performance, the salesperson can filter their commision by data and compare their commision from one time period to another.

```
SELECT SUM(SALESPERSON.Commission * SOLD.Price) AS CommissionEarned
FROM SOLD,SALESPERSON
WHERE SALESPERSON.ESSN = SOLD.ESSN AND SOLD.ESSN = '468-74-6913' AND SOLD.Date > '2010-01-01';
```

- A branch manager can query which models of cars are being sold and which are not, allowing them to decide which purchases to make from manufacturers and which purchases not to make.



Since every vehicle corresponds to some model, a branch manager can query what is being sold by matching a vehicle M_ID to a model M_ID and then matching that vehicle's V_ID to the SOLD table's vehicle V_ID. They can then group by a Model's make to gain some insight into which models are selling well, and which aren't.

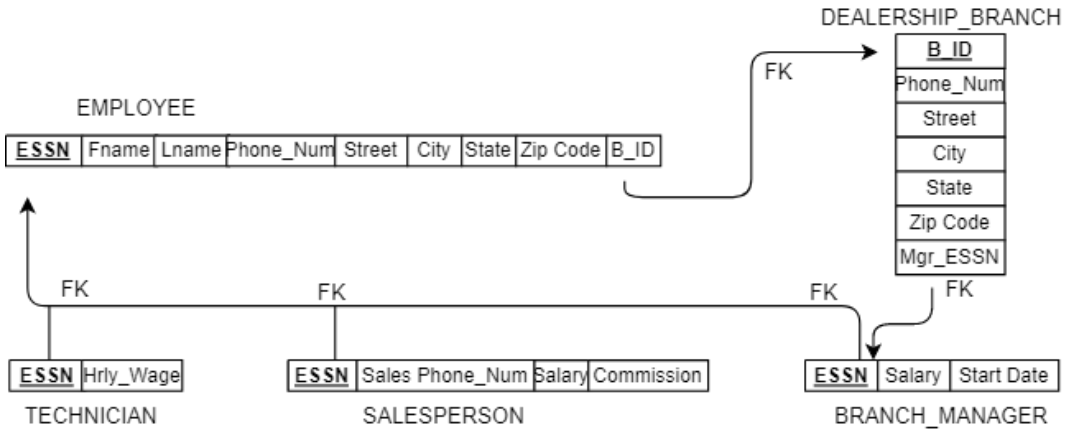
Models Sold

```
SELECT COUNT(MODEL.M_ID) AS numModels,MODEL.Make
FROM MODEL,VEHICLE,SOLD
WHERE VEHICLE.M_ID = MODEL.M_ID AND VEHICLE.V_ID = SOLD.V_ID
GROUP BY MODEL.Make;
```

Models Not Sold

```
SELECT COUNT(MODEL.M_ID) numModelsNotSold,MODEL.Make
FROM MODEL,VEHICLE
WHERE VEHICLE.M_ID = MODEL.M_ID AND
VEHICLE.V_ID NOT IN(SELECT SOLD.V_ID
                    FROM SOLD)
GROUP BY MODEL.Make;
```

- The dealership company will be able to compare the performance between branches, comparing revenues based on sales to see which managers are performing poorly and which managers need to be replaced.



Since every salesperson and branch manager is an employee and every employee works

at a dealership branch(identified by B_ID), the dealership company can view how each dealership branch is performing by calculating the total revenue from every salesperson, and then grouping by employee B_ID.

```
SELECT SUM(Sold.Price) AS Sales, EMPLOYEE.B_ID
FROM SOLD, SALESPERSON,EMPLOYEE
WHERE SOLD.ESSN = SALESPERSON.ESSN AND SALESPERSON.ESSN = EMPLOYEE.ESSN
GROUP BY EMPLOYEE.B_ID;
```

Summary & Teamwork

In this project we learned a lot about the complexity of designing an effective database. We spent days brainstorming and debating about the best way to simply model our ER diagram, and we still occasionally look back at it and think maybe there was some sort of better way to do it. We all learned about the importance of a good design and how without a good design there is no way to complete this sort of project without running into some major problems down the line. This was a valuable experience for us and this report shows the amount of work necessary to create a database which is efficient and useful.

Individual Contributions

Ivan Shapirov:
I worked on the functional requirements and data requirements sections. First by creating many of them and then updating them as we needed to. I helped in the creation of the EER diagram and the relational model. I also created the Relational Model and Data Dictionary portion of this document and contributed to the Implementation portion of the document. Finally, I helped in the implementation part of the project by generating a lot of the data for the SQL insert part, writing many of the insert queries, and writing most of the functional requirement queries.

Tommy Lim:
I worked on the EER diagram and the Entities and Attributes portion of the conceptual design during the initial phases of the project. With the groundwork laid down, I then converted the EER Diagram into a relational model. I worked closely with Ivan over dozens of hours and multiple days implementing our dealership DBMS and generating and inserting sensible mock data to ensure our product worked. I also created the power point presentation and recorded the functional requirements demo video for our DBMS.

LaKeesha Patterson:
I helped with writing the data requirements and mapping the ER diagram. I also helped create the relational model by mapping the relationships and entities into the relations. I worked to help create the entities and attributes portion of this document along with the data requirements sections. I also helped add many of the things we needed to add to the Appendix and helped with the powerpoint we used for our presentation.

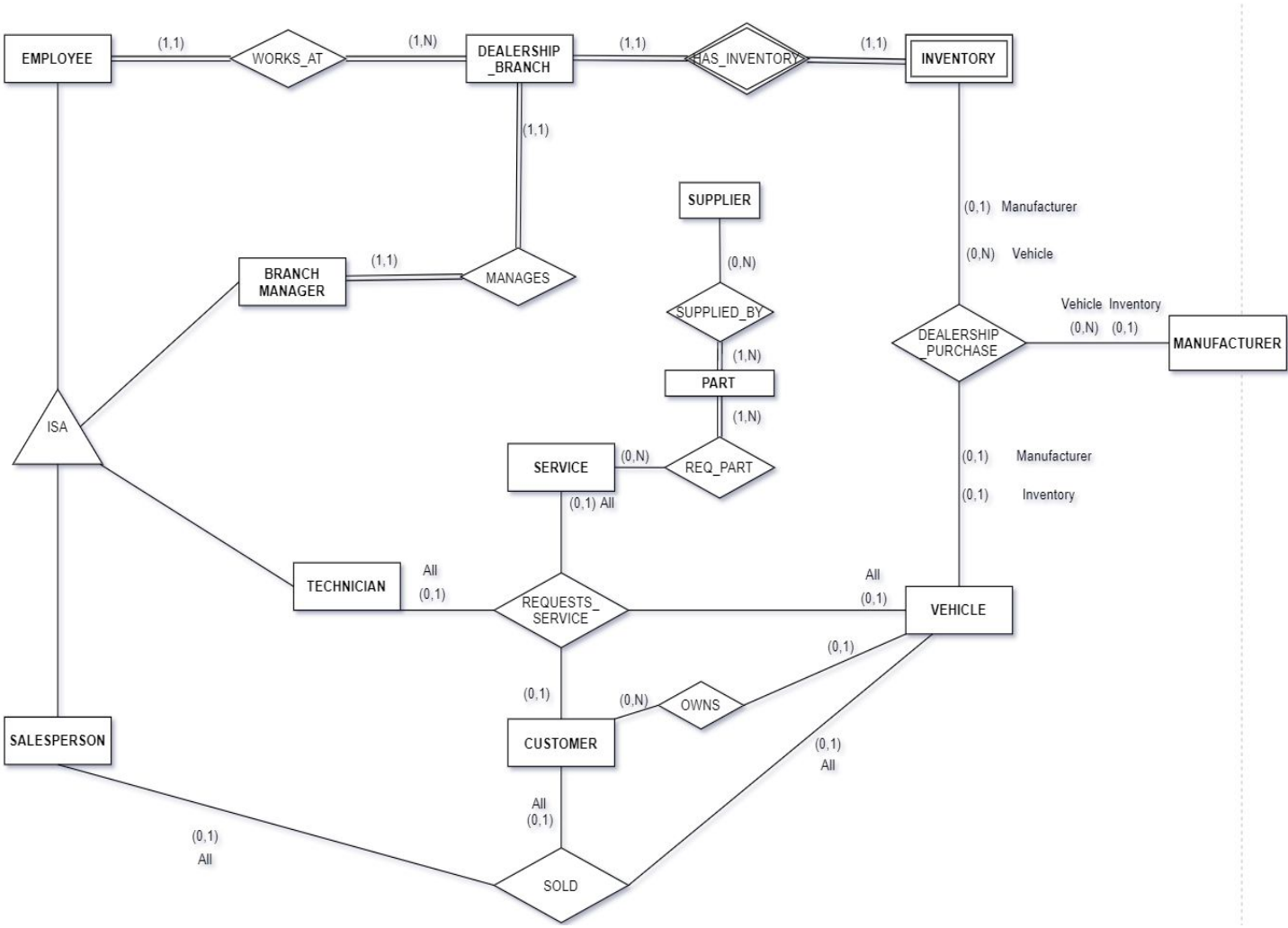
DEMO VIDEO LINK

Car Dealership Database Demo Video Link

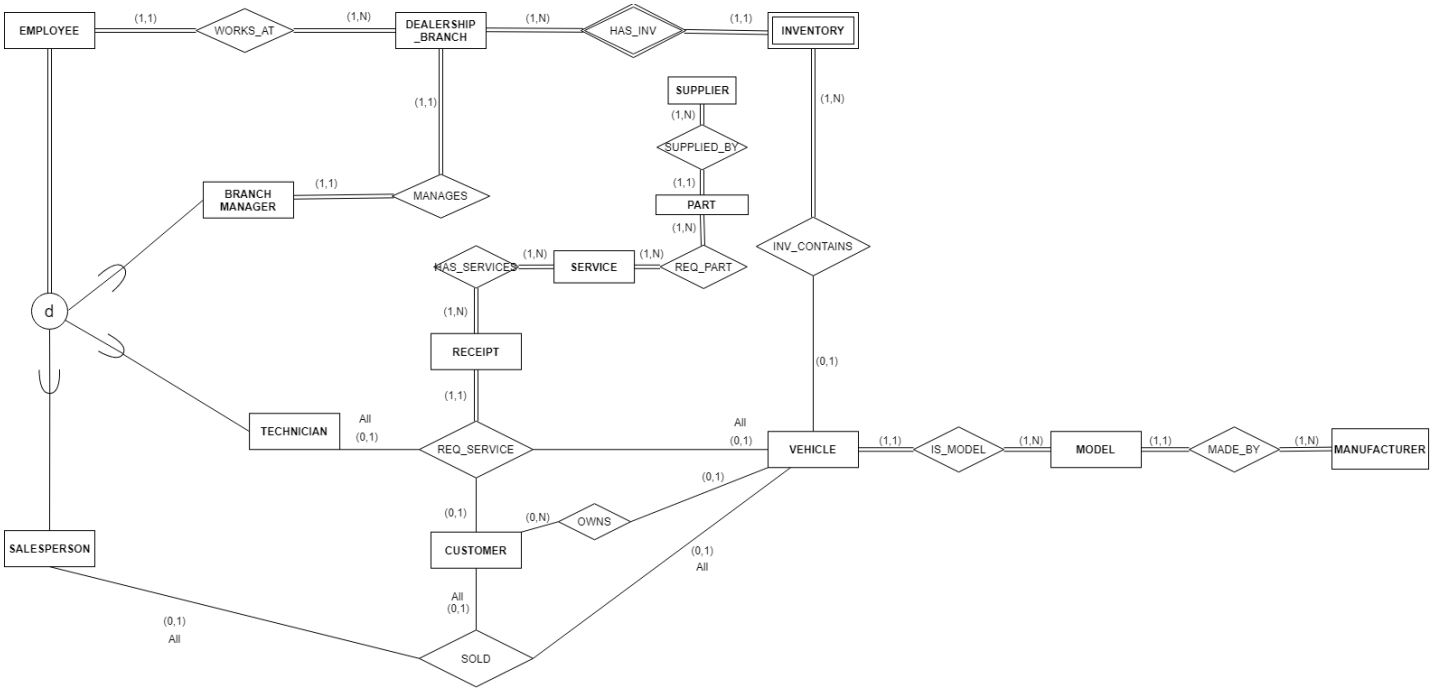
https://drive.google.com/open?id=1iHxFYYytElv_AGlgsjpUMIzdGNf9k7dW

Appendix

Initial ER Diagram



FINAL EER Diagram



CHANGE LOG

Functional Requirement changes
Removed

- A customer can be inserted into the database without a vehicle (such as through creating an online account), however, a vehicle cannot be inserted into the database unless it either has an associated inventory it is located in or an associated customer it belongs to.

This was removed because this problem went outside of the scope of our mini world. Since we are making a database for a dealership, it would not make sense for a dealership to keep track of vehicles that were not purchased from the dealership.

- A vehicle that has been sold must have its inventory ID set to NIL, indicating that it is no longer a part of the inventory, and its customer ID must be set to the ID of the customer who bought the vehicle.

We decided to change this and keep the inventory ID with the vehicle at all times, as this will allow us to keep track of which vehicles were sold from which inventory, allowing the dealership to assess each inventory and subsequently each branches' success.

- If a customer requires a new service to their car which is not currently in the database and requires parts that are not stored in the database, then the technician will have to manually insert the respective parts and supplier required for the completion of this service.

This was changed because it wasn't really showing how the database was used to obtain useful results, and this is more of a nonfunctional requirement.

New

- The manager can view the performance of each technician by looking at how many services each technician has completed.
- Salespeople will be responsible for adding in transactions for customers and altering them/ deleting them if an error has been made or some other problem occurred.
- Customers will be able to filter available cars online by any of the vehicle's or model's attributes.

Entities and Attributes:

Removed

- Removed Relationship ‘DEALERSHIP_PURCHASE’. Found to be outside scope of DBMS data requirements.
- Relationship ‘

New

- On EER Diagram: Converted ISA relationship on superclass Employee to a total participation Disjoint Specialization relationship, Where every employee must participate in one of its subclasses (Technician, Salesperson, or Branch Manager).
- Added Entity ‘MODEL’
- Added Entity ‘RECEIPT’
- Added Relationship ‘INV_CONTAINS’
- Added Relationship ‘IS_MODEL’
- Added Relationship ‘MADE_BY’
- Added Relationship ‘HAS_SERVICES’

RELATIONAL MODEL

SQL CODE

CREATE TABLES

```
create table EMPLOYEE
(
   ESSN VARCHAR(11) PRIMARY KEY,
   Fname VARCHAR(25),
   Lname VARCHAR(25),
   Phone_num VARCHAR(12),
   Birthdate DATE,
   Street VARCHAR(50),
   City VARCHAR(50),
   State VARCHAR(15),
   Zipcode VARCHAR(5),
   B_ID INT
);
create table BRANCH_MANAGER
(
   ESSN VARCHAR(11) PRIMARY KEY,
   Salary DECIMAL(9,2) NOT NULL Default 75000.00,
   Start_Date DATE
);

ALTER TABLE BRANCH_MANAGER
ADD FOREIGN KEY(ESSN)
REFERENCES EMPLOYEE(ESSN);
create table DEALERSHIP_BRANCH
(
   B_ID INT auto_increment PRIMARY KEY,
   Phone_Num VARCHAR(12),
   Street VARCHAR(50),
   City VARCHAR(50),
   State VARCHAR(15),
   Zipcode VARCHAR(5),
   Mgr_ESSN VARCHAR(11)
);

ALTER TABLE DEALERSHIP_BRANCH
ADD FOREIGN KEY (Mgr_ESSN)
REFERENCES BRANCH_MANAGER (ESSN);
ALTER TABLE EMPLOYEE
ADD FOREIGN KEY (B_ID)
```

```
REFERENCES DEALERSHIP_BRANCH (B_ID);
create table INVENTORY
(
   INV_ID INT auto_increment,
   B_ID INT,
   PRIMARY KEY(INV_ID, B_ID),
   FOREIGN KEY(B_ID) REFERENCES DEALERSHIP_BRANCH(B_ID)
);
create table SERVICE
(
   Service_ID INT auto_increment PRIMARY KEY,
   Type VARCHAR(100),
   Cost DECIMAL(7,2),
   Duration INT
);
create table RECEIPT
(
   Receipt_ID INT auto_increment PRIMARY KEY
);
create table PART
(
   Part_ID INT auto_increment PRIMARY KEY,
   P_name VARCHAR(100)
);
create table SUPPLIER
(
   SP_ID INT auto_increment PRIMARY KEY,
   S_name VARCHAR(50),
   Phone_Num VARCHAR(12),
   Street VARCHAR(50),
   City VARCHAR(50),
   State VARCHAR(15),
   Zipcode VARCHAR(5)
);
create table CUSTOMER
(
   C_ID VARCHAR(10) PRIMARY KEY,
   Fname VARCHAR(25),
   Lname VARCHAR(25),
   Birthdate DATE,
   Phone_Num VARCHAR(12),
   Street VARCHAR(50),
   City VARCHAR(50),
```

```
    State VARCHAR(15),
    Zipcode VARCHAR(5)
);
create table MANUFACTURER
(
    Man_name VARCHAR(30) PRIMARY KEY
);
create table MODEL
(
    M_ID INT auto_increment PRIMARY KEY,
    Make VARCHAR(15),
    Year INT,
    Type VARCHAR(15),
    HP INT,
    MPG INT,
    MSRP DECIMAL(8,2),
    Man_name VARCHAR(30),
    FOREIGN KEY(Man_name) REFERENCES MANUFACTURER(Man_name)
);
create table VEHICLE
(
    V_ID VARCHAR(7) PRIMARY KEY,
    Color VARCHAR(15),
    Curr_mileage INT,
    M_ID INT,
    FOREIGN KEY(M_ID) REFERENCES MODEL(M_ID)
);
create table OWNS
(
    C_ID VARCHAR(10),
    V_ID VARCHAR(7),
    PRIMARY KEY(C_ID, V_ID),
    FOREIGN KEY(C_ID) REFERENCES CUSTOMER(C_ID),
    FOREIGN KEY(V_ID) REFERENCES VEHICLE(V_ID)
);
create table TECHNICIAN
(
    ESSN VARCHAR(11) PRIMARY KEY,
    Hourly_wage Decimal(5,2) NOT NULL DEFAULT 18.75,
    FOREIGN KEY(ESSN) REFERENCES EMPLOYEE(ESSN)
);
create table REQ_SERVICE
(
```

```
    C_ID VARCHAR(10),
    V_ID VARCHAR(7),
    ESSN VARCHAR(11),
    Receipt_ID INT,
    Date DATE,
    PRIMARY KEY(C_ID, V_ID, ESSN, Receipt_ID),
    FOREIGN KEY(C_ID) REFERENCES CUSTOMER(C_ID),
    FOREIGN KEY(ESSN) REFERENCES TECHNICIAN(ESSN),
    FOREIGN KEY(RECEIPT_ID) REFERENCES RECEIPT(RECEIPT_ID),
    FOREIGN KEY(V_ID) REFERENCES VEHICLE(V_ID)
);
create table INV_CONTAINS
(
    B_ID INT ,
    INV_ID INT ,
    V_ID VARCHAR(7) ,
    PRIMARY KEY(B_ID, INV_ID, V_ID),
    FOREIGN KEY(B_ID) REFERENCES DEALERSHIP_BRANCH(B_ID),
    FOREIGN KEY(INV_ID) REFERENCES INVENTORY(INV_ID),
    FOREIGN KEY(V_ID) REFERENCES VEHICLE(V_ID)
);
create table HAS_SERVICES
(
    Receipt_ID INT,
    Service_ID INT,
    PRIMARY KEY(RECEIPT_ID, SERVICE_ID),
    FOREIGN KEY(RECEIPT_ID) REFERENCES RECEIPT(RECEIPT_ID),
    FOREIGN KEY(SERVICE_ID) REFERENCES SERVICE(SERVICE_ID)
);
create table SALESPERSON
(
    ESSN VARCHAR(11) PRIMARY KEY,
    Sales_Phone_Num VARCHAR(12),
    Salary DECIMAL(9,2) NOT NULL DEFAULT 35000,
    Commission DECIMAL(4,2) NOT NULL DEFAULT .25,
    FOREIGN KEY(ESSN) REFERENCES EMPLOYEE(ESSN)
);

create table REQ_PARTS
(
    Service_ID INT,
    Part_ID INT,
    Qty INT,
```

```
    PRIMARY KEY(Service_ID, Part_ID),
    FOREIGN KEY(Part_ID) REFERENCES PART(Part_ID),
    FOREIGN KEY(Service_ID) REFERENCES SERVICE(Service_ID)
);
create table SOLD
(
    ESSN VARCHAR(11),
    C_ID VARCHAR(10),
    V_ID VARCHAR(7),
    Price DECIMAL(9,2),
    Date DATE,
    PRIMARY KEY(ESSN, C_ID, V_ID),
    FOREIGN KEY(C_ID) REFERENCES CUSTOMER(C_ID),
    FOREIGN KEY(ESSN) REFERENCES SALESPERSON(ESSN),
    FOREIGN KEY(V_ID) REFERENCES VEHICLE(V_ID)
);
create table SUPPLIED_BY
(
    SP_ID INT,
    Part_ID INT,
    Price DECIMAL(8,2),
    PRIMARY KEY(SP_ID, Part_ID),
    FOREIGN KEY(Part_ID) REFERENCES PART(Part_ID),
    FOREIGN KEY(SP_ID) REFERENCES SUPPLIER(SP_ID)
);
```

INSERT

```
insert into CUSTOMER (C_ID, FName, Lname, Birthdate, Phone_Num, Street, City, State, Zipcode)
values
 ('007', 'Gisele', 'O"Siaghail', '1998-02-26', '716-138-1832', '036 Pearson Junction', 'Buffalo', 'NY',
'14276'),
 ('814', 'Loydie', 'Nobriga', '1984-06-29', '410-977-7851', '5808 Johnson Pass', 'Ridgely', 'MD', '21684'),
 ('901', 'Lori', 'Bibby', '1975-02-07', '952-870-1414', '41161 Vermont Parkway', 'Minneapolis', 'MN',
'55446'),
 ('441', 'Val', 'Goodenough', '1956-12-27', '214-722-1656', '10673 Fisk Alley', 'Dallas', 'TX', '75353'),
 ('143', 'Dorelia', 'Rickersey', '1936-10-18', '786-226-7158', '94980 Iowa Point', 'Miami', 'FL', '33147'),
 ('7901', 'Christan', 'Tew', '1951-07-29', '562-517-0478', '39133 Roxbury Terrace', 'Long Beach', 'CA',
'90847'),
 ('411', 'Quincey', 'Morriarty', '1985-10-05', '843-863-3562', '7 Almo Court', 'Beaufort', 'SC', '29905'),
 ('310', 'Roger', 'Titford', '1959-09-27', '319-334-5112', '5407 Kensington Terrace', 'Cedar Rapids', 'IA',
'52410'),
 ('776', 'Dex', 'Doswell', '1951-08-26', '316-450-5785', '9 Anthes Parkway', 'Wichita', 'KS', '67215'),
```

('127', 'Randie', 'Rowden', '1991-12-30', '803-822-5213', '15 Grasskamp Lane', 'Columbia', 'SC', '29220'),
('419', 'Larine', 'Doward', '1993-10-04', '214-621-1271', '20 Stoughton Trail', 'Dallas', 'TX', '75310'),
('088', 'Hanson', 'Finneran', '1951-04-06', '919-308-0321', '349 3rd Trail', 'Durham', 'NC', '27705'),
('916', 'Tamma', 'Lindsley', '2001-11-15', '808-532-8363', '25 Dakota Alley', 'Honolulu', 'HI', '96815'),
('482', 'Henrie', 'Hillin', '1984-06-20', '515-560-0738', '229 Westend Road', 'Des Moines', 'IA', '50981'),
('987', 'Carmencita', 'Bemment', '1988-03-28', '212-366-3536', '6 Oxford Court', 'New York City', 'NY', '10160'),
('265', 'Farand', 'Kordova', '1992-09-22', '212-177-3382', '4025 Jay Pass', 'New York City', 'NY', '10131'),
('161', 'Lincoln', 'Buckam', '1985-06-05', '229-183-4837', '63 La Follette Terrace', 'Valdosta', 'GA', '31605'),
('899', 'Shaylyn', 'McNalley', '1956-11-18', '267-541-7587', '65 Tomscot Center', 'Levittown', 'PA', '19058'),
('812', 'Vickie', 'Northey', '1960-10-06', '540-754-4695', '55466 Dixon Place', 'Fredericksburg', 'VA', '22405'),
('184', 'Cilka', 'Mauro', '1950-02-19', '951-525-1213', '7043 Canary Court', 'Riverside', 'CA', '92519'),
('278', 'Rani', 'Kellitt', '1957-04-29', '757-852-0847', '834 New Castle Way', 'Portsmouth', 'VA', '23705'),
('106', 'Malissa', 'Kiellor', '1953-05-04', '504-817-7928', '258 Northview Road', 'New Orleans', 'LA', '70154'),
('976', 'Trev', 'Kumar', '1981-09-20', '954-953-1852', '2588 Golf Alley', 'Fort Lauderdale', 'FL', '33305'),
('555', 'Peder', 'Wallenger', '1940-04-09', '305-126-0349', '14 Schmedeman Way', 'Miami', 'FL', '33142'),
('930', 'Gerladina', 'Priditt', '1987-04-25', '405-654-1669', '63436 Valley Edge Park', 'Oklahoma City', 'OK', '73179'),
('851', 'Fredra', 'Speechley', '1981-09-03', '714-479-4345', '642 Esch Circle', 'Fullerton', 'CA', '92835'),
('700', 'Gloriane', 'Yuranovev', '1989-06-02', '813-906-4530', '4615 Rowland Plaza', 'Tampa', 'FL', '33605'),
('912', 'Fredrika', 'Fayter', '1994-12-05', '917-727-0633', '277 Glacier Hill Crossing', 'New York City', 'NY', '10155'),
('421', 'Tadd', 'Hanshawe', '1947-05-20', '317-896-7368', '37 Claremont Trail', 'Indianapolis', 'IN', '46266'),
('166', 'Nat', 'Maty', '1958-02-22', '619-506-1154', '3 Randy Center', 'San Diego', 'CA', '92145');

insert into Dealership_Branch (Phone_num, Street, City, State, Zipcode) values
('404-149-6722', '4 Old Gate Court', 'Atlanta', 'GA', '31136'),
('517-610-1574', '255 Transport Point', 'Newport News', 'VA', '23612');

insert into EMPLOYEE (ESSN, Fname, Lname, Phone_num, Birthdate, Street, City, State, Zipcode, B_ID) values
('477-66-1385', 'Ara', 'Normant', '915-362-5917', '1975-10-02', '23346 Petterle Road', 'El Paso', 'TX', '88574', 1),
('102-55-0071', 'Trent', 'Ingle', '414-278-5402', '1992-09-17', '021 Riverside Lane', 'Milwaukee', 'WI', '53234', 2),

('334-77-8879', 'Rocky', 'Pattle', '804-381-5269', '1962-04-18', '857 Myrtle Lane', 'Richmond', 'VA', '23272', 1),
('791-56-1598', 'Lauraine', 'Craik', '719-832-0145', '1988-03-20', '74672 Talmadge Plaza', 'Colorado Springs', 'CO', '80930', 1),
('837-64-7739', 'Hilton', 'Folli', '202-187-5910', '1986-03-01', '2 Rockefeller Crossing', 'Washington', 'DC', '20260', 1),
('776-52-9735', 'Avram', 'Frazier', '816-498-7793', '1940-05-25', '3838 Bayside Hill', 'Kansas City', 'MO', '64149', 2),
('670-94-6117', 'Lilla', 'Bolter', '602-501-8295', '1992-03-09', '0598 Killdeer Court', 'Phoenix', 'AZ', '85083', 2),
('307-10-2238', 'Gelya', 'Griston', '915-640-4873', '1986-10-18', '35724 Butternut Junction', 'El Paso', 'TX', '79950', 2),
('468-74-6913', 'Star', 'Clinch', '850-984-8462', '1980-08-09', '609 Ramsey Plaza', 'Pensacola', 'FL', '32575', 1),
('259-31-7528', 'Allayne', 'Dark', '859-309-5735', '1956-12-20', '60916 Hollow Ridge Court', 'Lexington', 'KY', '40591', 1),
('427-30-5377', 'Hervey', 'Maidment', '619-257-0319', '1986-04-15', '4008 Messerschmidt Plaza', 'San Diego', 'CA', '92115', 1),
('421-63-0828', 'Hilliard', 'Fattori', '615-563-3637', '1971-11-22', '3759 Maple Plaza', 'Nashville', 'TN', '37205', 2),
('783-84-8346', 'Sibylle', 'Matveyev', '405-879-9414', '1967-07-16', '8 Clarendon Avenue', 'Oklahoma City', 'OK', '73157', 2),
('604-13-3834', 'Hamilton', 'Bennett', '502-232-8114', '1974-08-24', '12738 Darwin Lane', 'Louisville', 'KY', '40293', 2);

insert into Branch_Manager(ESSN, Salary, Start_date) values
('477-66-1385', 105000.00, '2005-10-08'),
('102-55-0071', 92000.00, '2007-11-20');

update Dealership_Branch
SET Mgr_ESSN = '477-66-1385'
WHERE B_ID = 1;

update Dealership_Branch
SET Mgr_ESSN = '102-55-0071'
WHERE B_ID = 2;

insert into technician(ESSN) Values
('334-77-8879'),
('791-56-1598'),
('837-64-7739'),
('776-52-9735'),
('670-94-6117'),

('307-10-2238');

insert into Salesperson(ESSN, Sales_Phone_Num) Values
('468-74-6913', '903-263-5596'),
('259-31-7528', '186-478-2607'),
('427-30-5377', '971-708-2992'),
('421-63-0828', '278-576-4232'),
('783-84-8346', '999-990-0247'),
('604-13-3834', '506-954-8989');

insert into INVENTORY(B_ID) VALUES
(1,1),
(1,2),
(2,1),
(2,2);

insert into SUPPLIER (S_name, Phone_Num, Street, City, State, Zipcode) values
('Skiles LLC', '786-224-2232', '3 Karstens Way', 'Miami', 'FL', '33111'),
('Raynor Inc', '510-220-6195', '8 Paget Park', 'Berkeley', 'CA', '94705'),
('Kautzer LLC', '210-849-3373', '1 Maple Road', 'San Antonio', 'TX', '78240');

insert into Manufacturer VALUES ('Honda'), ('Toyota'), ('Volkswagen'), ('Audi');

insert into MODEL(Make, Year, Type, HP, MPG, MSRP, Man_name) Values ('Accord', 2020, 'Sedan', 192, 38, 24000.00, 'Honda'), ('Civic', 2020, 'Sedan', 158, 38, 19850.00, 'Honda'),
('CR-V', 2020, 'SUV', 190, 28, 25050.00, 'Honda'), ('Tacoma', 2019, 'Truck', 159, 20, 26830.00, 'Toyota'),
('Supra', 2020, 'Coupe', 335, 24, 49990.00, 'Toyota'), ('Camry', 2020, 'Sedan', 203, 29, 24425.00, 'Toyota');

insert into MODEL(Make, Year, Type, HP, MPG, MSRP, Man_name) Values ('R8', 2018, 'Coupe', 532, 14, 166000.00, 'Audi'), ('A4', 2020, 'Sedan', 188, 27, 37400.00, 'Audi'),
('Q3', 2020, 'SUV', 228, 19, 34700.00, 'Audi'), ('Passat', 2020, 'Sedan', 174, 23, 22995.00, 'Volkswagen'),
('Jetta', 2020, 'Sedan', 147, 30, 18895.00, 'Volkswagen'), ('Beetle', 2019, 'Coupe', 174, 26, 20895.00, 'Volkswagen');

insert into SERVICE(Type, Cost, Duration) values
('Oil Change', 100.00, 60),
('Cabin air filter replacement', 45.00, 30),
('Trainsmission fluid flush', 100.00, 30),
('Brake pads & Rotor Replacement', 1200.00, 3600),

('Spark plug replacement', 200.00, 40);
insert into SERVICE(Type,Cost,Duration) values
('Windshield Replacement',300.00,60),
('Radiator Replacement',500.00,150),
('Water Pump Replacement',660.00,210),
('Installing New All-Season Tires',700.00,40),
('Serpentine Belt Replacement',900.00,180);

insert into PART(P_name) values
('Premium Synthetic Motor Oil, 5 quart'),
('Premium Oil Filter'),
('Premium Cabin Air Filter'),
('Super High Mileage Transmission Fluid'),
('Super Line Brake Pads'),
('Just-In-Time Brake Rotors'),
('Adamantium Spark Plugs');
insert into PART(P_Name) values ('Urethane'),
('Wind Resistant Windshield'),
('Preimum Air Compressor'),
('SuperHot Radiator'),
('Preimum Water Pump'),
('Dr. Freeze Coolant'),
('All Season Tires'),
('Medusa Serpentine Belt');

insert into supplied_by values
(1, 1, 20.00),
(1, 2, 8.00),
(1, 3, 19.00),
(1, 4, 19.00),
(1, 5, 120.00),
(2, 6, 225.00),
(2, 7, 18.00),
(2, 8, 100.00),
(2, 9, 190.00),
(2, 10, 300.00),
(3, 11, 350.00),
(3, 12, 400.00),
(3, 13, 100.00),
(3, 14, 75.00),
(3, 15, 200.00);

insert into VEHICLE (V_ID, Color, Curr_Mileage, M_ID) values

('2Q814', 'Violet', 9874, 7),
('41D088', 'Purple', 6193, 3),
('PN317T', 'Black', 1480, 6),
('MKM', 'Black', 2813, 9),
('MEQ4', 'Black', 1332, 1),
('815', 'Blue', 10137, 9),
('3CBA96', 'Blue', 3463, 6),
('IPL2L', 'Violet', 1795, 6),
('J8T', 'Purple', 8058, 5),
('3VVU', 'Silver', 5359, 7),
('XPN', 'Silver', 4667, 12),
('6SKW', 'Silver', 8304, 4),
('628', 'Gold', 7847, 4),
('S3Y', 'Pink', 4445, 8),
('7605', 'Black', 3377, 1),
('945S', 'Silver', 14053, 6),
('73949E', 'Silver', 8401, 11),
('OX3XC', 'Black', 10543, 2),
('GW8IK', 'Red', 14921, 5),
('OK03', 'Blue', 11285, 10),
('0MB6', 'Silver', 13537, 8),
('QTM3K4G', 'Pink', 3580, 5),
('M2C0', 'Blue', 10068, 3),
('8DR', 'Blue', 9915, 4),
('794K9', 'Silver', 7041, 6),
('XOK0', 'Pink', 10158, 2),
('2000GNS', 'Silver', 11637, 6),
('F567F7C', 'Yellow', 3798, 10),
('82AO', 'Red', 12709, 10),
('5797', 'Silver', 748, 2),
('P0L7', 'Blue', 1636, 2),
('MR6RA', 'Red', 11786, 7),
('40PQDZ', 'Black', 5794, 7),
('730', 'Red', 5597, 8),
('SI3E', 'Yellow', 12099, 5),
('J237460', 'White', 6084, 2),
('SBBS', 'White', 7888, 6),
('RVY4', 'White', 8023, 6),
('5293', 'Red', 5523, 4),
('S2IU650', 'Blue', 2956, 3),
('260XR', 'Blue', 315, 3),
('31123Q2', 'Green', 8742, 4),
('11Y', 'Red', 6443, 8),

('3501', 'Gold', 14828, 9),
('P5A86', 'Blue', 3585, 4),
('TL4H37X', 'Red', 12234, 10),
('V15L1', 'Violet', 7205, 9),
('B45BYS', 'Purple', 6878, 12),
('A98', 'Blue', 1030, 11),
('ZY9', 'Black', 6440, 12),
('Z44518', 'Silver', 5219, 4),
('6MMJ4', 'Purple', 11576, 2),
('9NV', 'Red', 8310, 12),
('HZ2U9', 'Blue', 10250, 11),
('UHDXT07', 'Black', 13268, 10),
('SP748E9', 'Pink', 14168, 7),
('JWZX', 'Blue', 5973, 8),
('49Z1Q4', 'Blue', 2186, 3),
('W7AY8', 'Blue', 3005, 9),
('X96Z8', 'Green', 11382, 9),
('B91', 'Yellow', 7528, 4),
('Y2OOP', 'White', 2320, 8),
('53JYH5', 'Yellow', 3484, 10),
('TBX8D', 'Red', 2848, 2),
('6G9N', 'Silver', 9542, 10),
('42699L', 'Blue', 116, 10),
('LA1L', 'Gold', 14673, 12),
('Y96LW', 'Blue', 5142, 2),
('UMC720', 'Gold', 8113, 5),
('VT9', 'Purple', 12821, 9),
('9P67C', 'Green', 3018, 11),
('YYO21', 'Red', 14589, 4),
('3JXTU', 'Blue', 6976, 5),
('11014', 'Red', 8600, 7),
('5H7UX16', 'Red', 14603, 5),
('079E689', 'Pink', 8815, 9),
('2F6XQAR', 'Violet', 6203, 4),
('Q2YSQ', 'Blue', 13270, 11),
('NGHP', 'White', 578, 11),
('8P8O84', 'Pink', 2265, 2),
('PA941', 'Silver', 9091, 8),
('0W2328I', 'Red', 13260, 4),
('489H', 'Blue', 4053, 1),
('926C9Q', 'Red', 13699, 8),
('00505', 'Green', 8984, 6),
('7H69', 'Red', 11262, 10),

('541', 'Blue', 12007, 6),
('T6ZW', 'Green', 3047, 4),
('226', 'Blue', 14934, 2),
('268', 'Black', 7766, 5),
('I2T9R77', 'Gold', 14123, 6),
('PC7F13M', 'Violet', 2902, 8),
('RNB17Z', 'Red', 1194, 6),
('637', 'Yellow', 4790, 4),
('4UR25D', 'Puce', 8701, 1),
('I150', 'Silver', 2216, 12),
('4MG', 'Purple', 5248, 1),
('9Y7N3L', 'Gold', 7753, 12),
('8MXT19', 'Red', 12496, 8),
('1E8640', 'Pink', 10034, 4),
('3VBO0', 'White', 5207, 9),
('6MEVWE', 'Pink', 3597, 9),
('KXYV', 'Black', 2648, 3),
('33577V9', 'Silver', 10999, 6),
('CLA', 'Green', 5163, 4),
('MTU4', 'Black', 5707, 3),
('15YG', 'Red', 1173, 1),
('G741', 'Pink', 3116, 6),
('8RC96C', 'Red', 1176, 1),
('W0SK120', 'Pink', 11262, 5),
('D9O6S', 'Red', 7485, 12),
('9I7232', 'Pink', 7625, 4),
('7QE7Q', 'White', 12771, 7),
('0C331', 'Pink', 4160, 12),
('L12K6', 'Black', 13209, 10),
('1OQE996', 'Silver', 12313, 12),
('H58', 'Puce', 13119, 8),
('1355', 'Violet', 7140, 8),
('UC9', 'Red', 3793, 11),
('0MS', 'Black', 3617, 6),
('JEN', 'Black', 2478, 1),
('DACICN', 'Gold', 5257, 12),
('755', 'Black', 10021, 9),
('7X2C71G', 'Pink', 9596, 7),
('1FA7M', 'Red', 2715, 10),
('BHG7', 'Black', 7575, 5),
('578', 'Pink', 9681, 10),
('RD1Y', 'Blue', 3666, 12),
('H0P4', 'Blue', 11902, 7),

('97CI', 'Black', 60, 5),
('1979773', 'Blue', 4451, 3),
('R8YAU0', 'Silver', 8167, 9),
('KFP5SO', 'Green', 4604, 12),
('VA964', 'Red', 4880, 2),
('938915', 'Violet', 9385, 11),
('Y1TX6D', 'Silver', 5045, 11),
('AOZ7L02', 'Black', 9438, 2),
('A7P07', 'Black', 3461, 4),
('6BWT', 'Pink', 3816, 12),
('H6EG56', 'Silver', 4564, 2),
('FGE', 'Pink', 4248, 7),
('3U59', 'Silver', 13261, 7),
('Z8TF', 'Silver', 562, 11),
('45XHSMH', 'Yellow', 11423, 11);

INSERT INTO INV_CONTAINS(B_ID,INV_ID,V_ID) values
(1,1,'2Q814'),
(1,1,'41D088'),
(1,2,'PN317T'),
(1,2,'MKM'),
(1,2,'MEQ4'),
(1,2,'815'),
(1,2,'3CBA96'),
(1,1,'IPL2L'),
(1,1,'J8T'),
(1,1,'3VVU'),
(1,1,'XPN'),
(1,2,'6SKW'),
(1,1,'628'),
(1,1,'S3Y'),
(1,1,'7605'),
(2,2,'945S'),
(2,1,'73949E'),
(2,1,'OX3XC'),
(2,1,'GW8IK'),
(2,1,'OK03'),
(2,2,'0MB6'),
(2,2,'QTM3K4G'),
(2,2,'M2C0'),
(2,2,'8DR'),
(2,1,'794K9'),

(2,1,'XOK0'),
(2,1,'2000GNS'),
(2,2,'F567F7C'),
(2,2,'82AO'),
(2,1,'5797'),
(1,1,'P0L7'),
(1,1,'MR6RA'),
(1,1,'40PQDZ'),
(2,1,'730'),
(2,2,'SI3E'),
(1,2,'J237460'),
(2,2,'SBBS'),
(2,1,'RVY4'),
(2,1,'5293'),
(2,1,'S2IU650'),
(2,2,'260XR'),
(1,1,'31123Q2'),
(2,2,'11Y'),
(2,2,'3501'),
(2,2,'P5A86'),
(1,2,'TL4H37X'),
(2,1,'V15L1'),
(1,1,'B45BYS'),
(1,2,'A98'),
(2,1,'ZY9'),
(2,1,'Z44518'),
(1,2,'6MMJ4'),
(1,1,'9NV'),
(2,1,'HZ2U9'),
(2,1,'UHDXT07'),
(1,2,'SP748E9'),
(2,1,'JWZX'),
(2,1,'49Z1Q4'),
(1,2,'W7AY8'),
(1,1,'X96Z8'),
(1,1,'B91'),
(1,2,'Y2OOP'),
(1,1,'53JYH5'),
(1,1,'TBX8D'),
(1,1,'6G9N'),
(2,2,'42699L'),
(2,2,'LA1L'),
(2,1,'Y96LW'),

(1,2,'UMC720'),
(1,2,'VT9'),
(1,1,'9P67C'),
(2,1,'YYO21'),
(2,2,'3JXTU'),
(1,2,'11014'),
(2,1,'5H7UX16'),
(1,2,'079E689'),
(2,1,'2F6XQAR'),
(2,2,'Q2YSQ'),
(2,2,'NGHP'),
(1,1,'8P8O84'),
(2,2,'PA941'),
(1,1,'0W2328I'),
(2,1,'489H'),
(2,1,'926C9Q'),
(2,2,'00505'),
(1,1,'7H69'),
(1,2,'541'),
(2,2,'T6ZW'),
(2,1,'226'),
(1,1,'268'),
(1,1,'I2T9R77'),
(2,2,'PC7F13M'),
(2,2,'RNB17Z'),
(1,2,'637'),
(2,1,'4UR25D'),
(2,1,'I150'),
(2,2,'4MG'),
(1,2,'9Y7N3L'),
(2,1,'8MXT19'),
(1,2,'1E8640'),
(1,1,'3VBO0'),
(1,2,'6MEVWE'),
(1,2,'KXYV'),
(2,1,'33577V9'),
(1,2,'CLA'),
(1,2,'MTU4'),
(1,2,'15YG'),
(1,1,'G741'),
(2,1,'8RC96C'),
(1,2,'W0SK120'),
(1,1,'D9O6S'),

(2,1,'9I7232'),
(2,1,'7QE7Q'),
(2,1,'0C331'),
(1,2,'L12K6'),
(2,1,'1OQE996'),
(1,2,'H58'),
(2,1,'1355'),
(2,1,'UC9'),
(2,1,'0MS'),
(1,2,'JEN'),
(1,2,'DACICN'),
(2,1,'755'),
(2,1,'7X2C71G'),
(2,1,'1FA7M'),
(1,1,'BHG7'),
(2,2,'578'),
(1,2,'RD1Y'),
(1,1,'H0P4'),
(1,2,'97CI'),
(1,1,'1979773'),
(2,1,'R8YAU0'),
(1,2,'KFP5SO'),
(1,1,'VA964'),
(2,2,'938915'),
(2,2,'Y1TX6D'),
(2,2,'AOZ7L02'),
(2,2,'A7P07'),
(2,1,'6BWT'),
(2,1,'H6EG56'),
(1,1,'FGE'),
(2,2,'3U59'),
(1,1,'Z8TF'),
(1,1,'45XHSMH');

insert INTO SOLD(ESSN,C_ID,V_ID,Price,Date) values
('468-74-6913','007','2Q814',27891.44,'2007-03-24'),
('468-74-6913','814','41D088',34070.72,'2001-05-13'),
('468-74-6913','901','PN317T',17715.81,'2016-11-25'),
('468-74-6913','441','MKM',15704.41,'2011-07-21'),
('468-74-6913','143','MEQ4',23012.05,'2011-07-26'),
('259-31-7528','7901','815',35742.12,'2001-10-15'),
('259-31-7528','411','3CBA96',42282.34,'2004-10-20'),

('259-31-7528','310','IPL2L',37178.58,'2014-01-29'),
('259-31-7528','776','J8T',21486.43,'2012-12-08'),
('259-31-7528','127','3VVU',38468.95,'2015-01-30'),
('427-30-5377','419','XPN',34553.4,'2014-04-17'),
('427-30-5377','088','6SKW',22442.42,'2019-12-26'),
('427-30-5377','916','628',16195.17,'2009-01-01'),
('427-30-5377','482','S3Y',16951.02,'2010-12-28'),
('427-30-5377','987','7605',42763.71,'2016-05-28'),
('421-63-0828','265','945S',16745.9,'2008-02-12'),
('421-63-0828','161','73949E',34394.36,'2003-01-13'),
('421-63-0828','899','OX3XC',33573.15,'2015-04-14'),
('421-63-0828','812','GW8IK',39177.27,'2016-04-26'),
('421-63-0828','184','OK03',28981.49,'2013-02-12'),
('783-84-8346','278','0MB6',17599.29,'2003-02-16'),
('783-84-8346','106','QTM3K4G',31833.86,'2012-12-21'),
('783-84-8346','976','M2C0',32998.4,'2008-12-29'),
('783-84-8346','555','8DR',30015.21,'2001-04-23'),
('783-84-8346','930','794K9',21272.89,'2019-02-24'),
('604-13-3834','851','XOK0',43328.68,'2004-05-15'),
('604-13-3834','700','2000GNS',30587.53,'2012-11-11'),
('604-13-3834','912','F567F7C',42072.26,'2019-12-04'),
('604-13-3834','421','82AO',29318.82,'2015-06-20'),
('604-13-3834','166','5797',43288.22,'2016-08-13');

insert INTO OWNS(C_ID,V_ID) values
('007','2Q814'),
('814','41D088'),
('901','PN317T'),
('441','MKM'),
('143','MEQ4'),
('7901','815'),
('411','3CBA96'),
('310','IPL2L'),
('776','J8T'),
('127','3VVU'),
('419','XPN'),
('088','6SKW'),
('916','628'),
('482','S3Y'),
('987','7605'),
('265','945S'),
('161','73949E'),

('899','OX3XC'),
('812','GW8IK'),
('184','OK03'),
('278','0MB6'),
('106','QTM3K4G'),
('976','M2C0'),
('555','8DR'),
('930','794K9'),
('851','XOK0'),
('700','2000GNS'),
('912','F567F7C'),
('421','82AO'),
('166','5797');

insert into REQ_PARTS(Service_ID, Part_ID, Qty) values
(1, 1, 1),
(1,2,1),
(2,3,1),
(3, 4, 1),
(4, 5, 2),
(4, 6, 2),
(5, 7, 2),
(6, 9, 1),
(6, 8, 1),
(7, 10, 1),
(7, 11, 1),
(8, 12, 1),
(8, 13, 1),
(9, 14, 1),
(10, 15, 1);

insert into Receipt values(),();

insert into has_services (Receipt_ID, Service_ID) values
(1, 1),
(2, 4),
(3, 7),
(4, 6),
(5, 7),
(6, 10),
(7, 4),
(8, 8),
(9, 7),

(10, 4),
(11, 9),
(12, 6),
(2, 5),
(14, 7),
(15, 1),
(16, 8),
(13, 1),
(1, 7),
(1, 9),
(17, 4),
(17,3),
(18,7),
(19,2),
(20, 3),
(21, 2),
(22, 4),
(23, 1),
(23, 8),
(24, 3);

insert into req_service(C_ID, V_ID, ESSN, Receipt_ID, Date) values
('007', '2Q814','334-77-8879', 1, '2020-01-10'),
('007', '2Q814','334-77-8879', 3, '2020-04-10'),
('814','41D088', '791-56-1598', 2, '2020-01-10'),
('901','PN317T', '837-64-7739', 4, '2019-02-10'),
('441','MKM','334-77-8879', 5, '2018-05-12'),
('143','MEQ4','791-56-1598', 6, '2019-09-09'),
('7901','815', '837-64-7739', 7, '2017-10-10'),
('411','3CBA96','791-56-1598', 8, '2019-12-12'),
('310','IPL2L','837-64-7739', 9, '2016-11-11'),
('127','3VVU', '334-77-8879', 10, '2017-04-02'),
('419','XPN', '837-64-7739', 22, '2019-05-09'),
('088','6SKW', '791-56-1598', 23, '2018-06-06');

insert into req_service(C_ID, V_ID, ESSN, Receipt_ID, Date) values
('265','945S','776-52-9735', 11, '2012-05-05'),
('161','73949E','776-52-9735', 12, '2018-09-09'),
('899','OX3XC', '776-52-9735', 13, '2017-07-07'),
('812','GW8IK', '670-94-6117', 14, '2020-01-04'),
('812','GW8IK', '670-94-6117', 15, '2019-03-03'),
('812','GW8IK', '670-94-6117', 16, '2018-05-01'),
('184','OK03', '307-10-2238', 17, '2017-02-02'),

```
('278','0MB6', '307-10-2238', 18, '2015-05-05'),
('106','QTM3K4G', '307-10-2238', 19, '2014-04-01'),
('976','M2C0', '776-52-9735', 20, '2018-08-08'),
('555','8DR','307-10-2238', 21, '2017-07-07'),
('930','794K9', '670-94-6117', 24, '2019-11-09');
```

Queries

```
/*Functional Requirement 1: Example of querying cost of all services received on a vehicle*/
SELECT SUM(SERVICE.cost) AS TotalServiceCost
FROM SERVICE
WHERE SERVICE.Service_ID IN (SELECT HAS_SERVICES.Service_ID
                    FROM HAS_SERVICES
                    WHERE HAS_SERVICES.Receipt_ID IN (SELECT
REQ_SERVICE.Receipt_ID
                                FROM REQ_SERVICE
                                WHERE REQ_SERVICE.C_ID = '007' AND
REQ_SERVICE.V_ID = '2Q814' ));

/*Functional Requirement 1: Example of querying names of all services received on a vehicle*/
SELECT SERVICE.Type,SERVICE.Cost
FROM SERVICE
WHERE SERVICE.Service_ID IN (SELECT HAS_SERVICES.Service_ID
                    FROM HAS_SERVICES
                    WHERE HAS_SERVICES.Receipt_ID IN (SELECT
REQ_SERVICE.Receipt_ID
                                FROM REQ_SERVICE
                                WHERE REQ_SERVICE.C_ID = '007' AND
REQ_SERVICE.V_ID = '2Q814'));

/*Functional Requirement 2: Querying Revenues of each salesperson*/
SELECT SUM(SOLD.Price) AS Revenue,SOLD.ESSN
FROM SOLD
GROUP BY SOLD.ESSN;

/*Functional Requirement 2: Querying Revenues of each salesperson, and getting the names of
the employees*/
SELECT SUM(SOLD.Price) AS
Revenue,EMPLOYEE.ESSN,EMPLOYEE.FName,EMPLOYEE.LName
```

```
FROM SOLD,EMPLOYEE,SALESPERSON
WHERE SOLD.ESSN = SALESPERSON.ESSN AND SALESPERSON.ESSN =
EMPLOYEE.ESSN
GROUP BY SOLD.ESSN;

/*Functional Requirement 3: Querying all services which technician may have to do*/
SELECT SERVICE.Type
FROM SERVICE;

/*Functional Requirement 3: Querying parts which can be ordered for each service*/
SELECT SERVICE.Type,Part.P_name,SUPPLIED_BY.Price
FROM PART,REQ_PARTS,SERVICE,SUPPLIED_BY
WHERE REQ_PARTS.Service_ID = SERVICE.Service_ID AND REQ_PARTS.Part_ID =
Part.Part_ID
AND PART.Part_ID = SUPPLIED_BY.Part_ID;

/*Functional Requirement 3: Querying duration of a specific service by ID and Type*/
SELECT SERVICE.duration
FROM SERVICE
WHERE SERVICE.Service_ID = 1;

SELECT SERVICE.duration
FROM SERVICE
WHERE SERVICE.Type = 'Serpentine Belt Replacement';

/*Functional Requirement 4: Querying number of vehicles in the inventory*/
SELECT COUNT(INV_CONTAINS.V_ID) AS numVehicles
FROM INV_CONTAINS
WHERE INV_CONTAINS.B_ID =2 AND INV_CONTAINS.INV_ID = 2
AND INV_CONTAINS.V_ID NOT IN (SELECT OWNS.V_ID FROM OWNS);

/*Functional Requirement 5: Querying how many services were completed by each technician at
branch 1*/
SELECT COUNT(REQ_SERVICE.ESSN) AS
numServices,TECHNICIAN.ESSN,EMPLOYEE.FName,EMPLOYEE.LName
FROM REQ_SERVICE,TECHNICIAN,EMPLOYEE
WHERE REQ_SERVICE.ESSN = TECHNICIAN.ESSN AND TECHNICIAN.ESSN =
EMPLOYEE.ESSN AND EMPLOYEE.B_ID = 1
GROUP BY REQ_SERVICE.ESSN;
```

```
/*Functional Requirement 6: Example of deleting transaction records*/
DELETE
FROM SOLD
WHERE SOLD.C_ID = '419' AND SOLD.ESSN = '427-30-5377' AND SOLD.V_ID = 'XPN';

/*Functional Requirement 6: Example of altering transaction records*/
UPDATE SOLD
SET SOLD.Price = 15000.00
WHERE SOLD.C_ID = '088' AND SOLD.ESSN = '427-30-5377' AND SOLD.V_ID = '6SKW';

/*Functional Requirement 6: Confirming deleting and update occurred*/
SELECT *
FROM SOLD
WHERE SOLD.ESSN = '427-30-5377';

/*Functional Requirement 7: Multiple examples of filtering available vehicles*/
/*Filtering by price*/
SELECT *
FROM VEHICLE, MODEL
WHERE VEHICLE.M_ID = MODEL.M_ID AND MODEL.MSRP<20000 AND
VEHICLE.V_ID NOT IN(SELECT SOLD.V_ID FROM SOLD);

/*Filtering by model make*/
SELECT *
FROM VEHICLE, MODEL
WHERE VEHICLE.M_ID = MODEL.M_ID AND MODEL.Make='R8' AND VEHICLE.V_ID
NOT IN(SELECT SOLD.V_ID FROM SOLD);

/*Filtering by color*/
SELECT *
FROM VEHICLE, MODEL
WHERE VEHICLE.M_ID = MODEL.M_ID AND VEHICLE.Color = 'Red' AND
VEHICLE.V_ID NOT IN(SELECT SOLD.V_ID FROM SOLD);

/*Filtering by MPG*/
SELECT *
FROM VEHICLE, MODEL
```

```
WHERE VEHICLE.M_ID = MODEL.M_ID AND MODEL.MPG>25 AND VEHICLE.V_ID
NOT IN(SELECT SOLD.V_ID FROM SOLD);


/*Functional Requirement 8: Calculating commission for a salesperson*/
SELECT SUM(SALESPERSON.Commission * SOLD.Price) AS CommissionEarned
FROM SOLD,SALESPERSON
WHERE SALESPERSON.ESSN = SOLD.ESSN AND SOLD.ESSN = '468-74-6913';

/*Functional Requirement 8: Calculating commission after a certain date*/
SELECT SUM(SALESPERSON.Commission * SOLD.Price) AS CommissionEarned
FROM SOLD,SALESPERSON
WHERE SALESPERSON.ESSN = SOLD.ESSN AND SOLD.ESSN = '468-74-6913' AND
SOLD.Date > '2010-01-01';

/*Functional Requirement 9: Querying number of each model sold*/
SELECT COUNT(MODEL.M_ID) AS numModels,MODEL.Make
FROM MODEL,VEHICLE,SOLD
WHERE VEHICLE.M_ID = MODEL.M_ID AND VEHICLE.V_ID = SOLD.V_ID
GROUP BY MODEL.Make;

/*Functional Requirement 9: Querying number of each model not sold*/
SELECT COUNT(MODEL.M_ID) numModelsNotSold,MODEL.Make
FROM MODEL,VEHICLE
WHERE VEHICLE.M_ID = MODEL.M_ID AND
VEHICLE.V_ID NOT IN(SELECT SOLD.V_ID
                    FROM SOLD)
GROUP BY MODEL.Make;

/*Functional Requirement 10: Comparing branches by sales*/
SELECT SUM(Sold.Price) AS Sales, EMPLOYEE.B_ID
FROM SOLD, SALESPERSON,EMPLOYEE
WHERE SOLD.ESSN = SALESPERSON.ESSN AND SALESPERSON.ESSN =
EMPLOYEE.ESSN
GROUP BY EMPLOYEE.B_ID;
```