Name: Kokil Dhakal

**Table of Contents**

[Project Direction Overview 1](#_Toc127564061)

[Use Cases and Fields 2](#_Toc127564062)

[Structural Database Rules 5](#_Toc127564063)

[Conceptual Entity-Relationship Diagram 7](#_Toc127564064)

[Full DBMS Physical ERD 7](#_Toc127564066)

[Stored Procedure Execution and Explanations 10](#_Toc127564067)

[Question Identification and Explanations 12](#_Toc127564068)

[Query Executions and Explanations 13](#_Toc127564069)

[Index Identification and Creations 14](#_Toc127564070)

[Query Driven index placement: 15](#_Toc127564071)

[Summary and Reflection 18](#_Toc127564072)

# Project Direction Overview

I would like to design a database for a car dealership company “CarMaxDeal” which will have their own website. This company uses an application that can access database to track all the cars they have for sell. This company also tracks inventory of the car like make, model, year, color etc. of the cars. Similarly, they must track of all sales information like sale date, sold car type, salesperson, customer to whom it was sold etc. Company also tracks the customers and employee who were involved in the transaction for a specific car.

A customer can check all the available cars and their features as well as their price using company web site. While an employee can check the status of car, price and initiate the transaction if a customer likes it and want to buy.

Here are some brief examples of how someone can use the database. Customer uses company’s website to check all the available cars, colors, make, model, and all other features etc. Customer can also check price of car. if they like it, they go nearby company’s dealer. If a customer decided to buy, employee check availability and initiate transaction and customer buy car.

Database stores information about cars’ inventory, sales information, customers information, Employee information, information about the car status, information about car features and information about transactions.

# Use Cases and Fields

Update, if necessary, the five or more use cases that enumerate steps of how the database will be typically used and identify significant database fields needed to support each use case.

**Vehicles receiving from vender use case:**

Managing the inventory of new and used vehicles would include.

1. Dealership order/receives vehicles from Venders.
2. This cause addition of new or used vehicles in the database inventory.

Significant fields for this use case are listed below.

|  |  |  |
| --- | --- | --- |
| Field | What it stores | Why it’s needed |
| VehicleID | Vehicle identification number generated by dealerships ‘application. | This is needed to track the vehicle. This uniquely identify the specific vehicle |
| Make | This field store Car company | This is needed to track the vehicle’s company |
| Model | This field stores Car model | This is needed to know what kind of model of car in the inventory |
| Color | This field Car color | This is needed to know what kind of color of car in the inventory |
| Year | This field stores Car makes year | This is needed to know what year of car in the inventory. |
| VIN | Vehicle Identification Number of the car. | This is needed to uniquely identify car |
| VenderID | Stores the type of service getting from venders | To track whether it is receiving used car, new car, and part for its inventory. |
| Condition | Condition of car received | Track whether it is new or used |
| ReceivedDate | Car received data | Track car received date |
| DealershipID | It stores which dealership sold a particular vehicle | Track the dealership and vehicle |
| Market\_price | It stores price decided by dealership | Needed to track MP of vehicle |

**Vender use case:**

1. When company receives car from vender its database updated with new car
2. When company sell a car, its database updated with one less car in inventory.

|  |  |  |
| --- | --- | --- |
| Field | What it stores | Why it is needed |
| VenderID | Vender Identification Number | Uniquely identify vender |
| VenderName | Vender name | To track Vender |
| VenderAddress | Vender Address | To track the Vender address |
| PhoneNumber | Vender Phone number | To track the vender phone |

**Tracking sales use case**

Tracking sell is important for the company to know what kind of car more frequently selling and which dealer is doing best and which need an improvement.

1. Employee long in their account and check for car of customer choice in the system.
2. Employees sell the car to customers.
3. Selling one car does delete one entity instance from the car inventory.

Significant fields for this use case are.

|  |  |  |
| --- | --- | --- |
| Field | What it stores | Why it’s needed |
| Date of sale | The date of care sale | Needed to track when car was sold for report analysis |
| CustomerID | Customer to whom car was sold | Need to track the customer to whom car was sold |
| VehicleID | Type of vehicle it sold | This tracks type of vehicle sold |
| EmployeeID | Employee who involved in selling car (Sale person) | This tracks Employee involved in transaction of the car |
| Transaction ID | Type of Trasaction | This field tracks the price at which vehicle was sold. It is needed for report generation. |
| SaleID | It stores sale Identification number | It is needed to track sales |
| Sale\_Price | It stores sale price of car | It is needed to track sold price of car. |

**Employee hire/leave use case**

This use case required ability of database to store employee information. When new employee is hire, their information are stores in the database. Significant field are given below.

|  |  |  |
| --- | --- | --- |
| Field | What it stores | Why it’s Needed |
| EmployeeID | It Stores the Employee identification number | Needed to track the Employee uniquely identify the Employee. |
| FirstName | This is First name of Employee | Need to display Employee who is involved in dealing with customer |
| LastName | This is last name of Employee | Need to display Employee who is involved in dealing with customer |
| Salary | This is salary of employee | This field tracks the salary of the employee. This is needed to determine total cost of the company. |
| Position | This position of employee | Needed for cost analysis |
| DOB | date of birth of employee | Needed to identify employee, also to know how long employee being employed in current company |
| Address | Address of employee | To identify employee, another means of communication via mail. |
| DepartmentID | It stores employee’s department | To track employee and department |

**Customer Tracking use case:**

Managing the relationship with the customers, including storing customer information, tracking customer interactions and purchase history, and providing customer service. This use case would require the ability to store customer information, track customer interactions, and provide customer service. Whenever customer come in to buy car, Salesman check whether customer is new to dealerships, if not new, they verify the customer information if customer is new, new record will be added to database. Significant fields are given below.

|  |  |  |
| --- | --- | --- |
| Field | What it stores | Why it is Needed |
| CustomerID | Customer identification number which is unique | Uniquely identify the customer |
| LastName | Customer Last name | This is needed to address a customer while communicating with them. |
| FirstName | Customer First name | This is needed to address a customer while communicating with them |
| PhoneNumber | Customer Phone number | This is needed to address a customer while communicating via text or call with them |
| Email Address | Customer email address | This is needed to address a customer while communicating via email with them |
| VehicleID | It stores type of services customer received | This is needed to track the type of services receiving by customer |

**Dealership use Case:**

- Dealership sells vehicle which directly interact with the company database.

|  |  |  |
| --- | --- | --- |
| Field | What it stores | Why it is needed |
| DealershipID | Uniquely identify dealership | To track dealership |
| Name\_dealership | Name of dealership | To track dealership |
| Address\_dealership | Address of dealership | To locate dealership |
| Phone\_dealership | Phone number of dealership | Dealership info |
| VenderID | IT stores the vender info | To track Vender |

**Payment uses case:**

1. When Employee initiates transaction, payment in involved
2. We need database to get some information about the transaction.

Significant field involved are:

|  |  |  |
| --- | --- | --- |
| Field | What it stores | Why it is needed |
| PaymentID | It stores unique payment identification number | To uniquely identify each payment |
| TransactionID | It stores the transaction information | To track each transaction that involved payment |
| CustomerID | It stores customer involved in payment | To track customer who invoved in payment |
| Date of payment | Data of payment | Track the date of payment |
| Payment\_amout | Total payment price | To track total amount of price in each payment. |

# Structural Database Rules

From the above use cases, following entities can be identified.

|  |  |  |
| --- | --- | --- |
| Vehicle | Vender | Customer |
| Employee | Department | Sale |
| Payment | Dealership |  |

1. Each **Dealership has many vehicles**; each **vehicle** has a dealership.
2. Each **vehicle** may sell to a customer; Each **customer** may buy many vehicles.
3. Each **vehicle** may sell by a dealership; each **dealership** may sell many vehicles.
4. Each **dealership** may have many departments; each **department** has a dealership.
5. Each **department** may provide many services to customers; and each **service is** received from a department.
6. Each **department** has many employees; each **employee** has its own department.
7. Each **employee** may initiate a transaction at a time; each **transaction** is initiated by an employee.
8. Each **customer** may involve many transactions: each **transaction** is from a customer.
9. Each **Vender** may have many dealerships; Each **dealership** may have many Venders (M: N)
10. Each sale has a vehicle; each vehicle may be sold many times in its lifetime.
11. Each sale can have many payments; each payment is for a sale.

For **specialization-generalization**, I have found an entity that need specialization. That is Employee. Hence, I have included structural rules based on this.

1. An Employee is Mechanic, Salesperson, or none of these.

After **normalization**, few more entities are included here.

1. Each vehicle has a type like SUV, sedan, or hatchback; in each type there are many vehicles.
2. Each vehicle has a make like Honda, Subaru, or Toyota; In each make there are many vehicles.
3. Each vehicle has a model like rav4, highlander, or XL; In each model There are many vehicles.
4. Each vehicle has a color like white, grey, or black; In each color there are many vehicles
5. Each vender has a address; and each address may have a vender.
6. Each dealership has a address; each address may have a dealership.
7. Each employee may have many addresses, each address may have an employee.
8. Each customer may have many addresses, each address may have a customer.

# Conceptual Entity-Relationship Diagram

# Diagram, engineering drawing Description automatically generated

# Full DBMS Physical ERD

I have included following attributes with their data types, constraints, and reasoning below in tabular form.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table** | **Attribute** | **Data types** | **Reasoning** | **Example** |
| Vehicle | Vehicle\_id | DECIMAL (12) | This uniquely identify the table, number with 12 digits is enough for a good number of vehicles. | 1 |
| Vehicle | VIN | VARCHAR (17) | Vehicle identification number usually has combination of characters and number up to 17. | 4Y1SL65848Z411439 |
| Vehicle | Mileage | DECIMAL (7) | 7 digits number of car mileage enough for a used car in the database. | 25000 |
| Vehicle | Price | DECIMAL (8,2) | 8 digits price of a car is enough for address all kind of car with up to two digits after decimal. | 50600.90 |
| Vehicle | Condition | VARCHAR (5) | Vehicle can be used, or new, 5-digit characters is enough for this attribute | New |
| VehicleI | Received\_date | DATE | It is the date when a specific vehicle is received in dealership | 12-Aug-2020 |
| Vehicle | year | DATE | It stores year car was made. Date in year | 2018 |
| vehicle | Vender\_price | Decimal(8,2) | This is the price of car from vender | 67444 |
| Type | Type\_id | DECIMAL (12) | It uniquely identifies each row in type table | 4 |
| Type | Type | VARCHAR (75) | This is type of vehicle, and 75 digits characters are enough to address all kind of vehicle type | E.g., SUV, Sedan, Hatchback |
| Make | Make\_id | DECIMAL (12) | It uniquely identifies each row in make table | 5 |
| Make | Make | VARCHAR (75) | It is vehicle make and I am allowing vehicle make named up to 75 characters which are enough for any make of car. They are usually short. | Honda, Toyota |
| Model | Model\_id | Decimal (12) | Uniquely identifies each row in table | 3 |
| Model | Model | VARCHAR (75) | It is model name of car; 75 character’s names are enough for a car model. | RAV4, Highlander |
| Color | Color\_id | DECIMAL (12) | It uniquely identifies each row of color table | 6 |
| Color | Color | VARCHAR (75) | It stores the name of color, and 75 characters are enough to name any color | Grey,black |
| Customer | Customer\_id | DECIMAL (12) | Uniquely identifies each row in customer table, 12 digits number is enough for many customers(billions) | 2 |
| Customer | Last\_name | Varchar (128) | Customer last name with up to 128 characters is allowed | Smith |
| Customer | First\_name | VARCHAR (128) | Customer first name with up to 128 characters is allowed and enough as well. | Briana |
| Customer | Phone | DECIMAL (10) | Phone numbers are made up of 10 digits. | 6174448912 |
| Customer | Address\_id | Decimal (12) | It links the customer table with address table to get customer address | Street city,state,country,zip |
| Customer | Email\_id | VARCHAR (128) | I am allowing up to 128 characters for the email id. Which is enough for the all kind of email id. | c@b.com |
| Payment | Payment\_id | DECIMAL (12) | Uniquely identifies each row in payment,12 digits numbers are enough to accommodate large number of rows in payment table. | 4 |
| Payment | Date\_of \_payment | DATE | This the date at which payment was issued. | 12-Aug-2019 |
| Payment | P\_amount | DECIMAL (8,2) | 8 digits price of car and 8 digits amount would be enough to accommodate the total amount of car payment.it also allow decimal values up to2 digits after decimal | 40000.99 |
| Payment\_type | P\_type\_id | DECIMAL (12) | Uniquely identifies each row in payment\_type table, 12 digits rows are large numbers of collection of information | 5 |
| Payment\_type | Method | VARCHAR (75) | 75characters are enough to name the method of payment | e.g., cash, credit card, check |
| Car\_sale | Sale\_id | DECIMAL (12) | Uniquely identifies each row in car\_sale table. 12 digits number can include billions of rows in the table. | 1000 |
| Car\_sale | Sale\_price | DECIMAL (8,2) | I am allowing up to 8 digits car price prices which is rare and with 2 digits after decimal | 26900.75 |
| Car\_sale | Sale\_date | DATE | It stores date at which car was stored | 04-Oct-2021 |
| Employee | Employee\_id | DECIMAL (12) | Uniquely identifies each row in the Employee table. 12 digits number can accommodate many rows in billions. | 9 |
| Employee | Last\_name | VARCHAR (128) | This table allow last name up to 128 characters and this is enough. | DHAKAL |
| Employee | First\_name | VARCHAR (128) | This table allow first name up to 128 characters and this is enough for most of the first name | KOKIL |
| Employee | Salary | DECIMAL (8,2) | It stores salary up to 8 digits amount which is huge with up to two digits after decimal | 75000.88 |
| Employee | Position | VARCHAR (128) | 35 digits name of a position is enough to name all kind of employee position. | Sale person |
| Employee | DOB | DATE | It stores of employee born date to verify an employee. | 12-Sep-1990 |
| Mechanic | Employee\_id | DECIMAL (12) | This from subtype table of supertype employee, it will have same primary key as employee have and same primary key also acts as foreign key | 8 |
| Mechanic | has\_mech\_certified | BOOLEAN | This is Boolean value; it will have either true or false | True, False |
| Sale \_Person | Employee\_id | Employee\_id | This from subtype table of supertype employee, it will have same primary key as employee have and same primary key also acts as foreign key | True or false |
| Sale\_person | Does\_sale\_car | BOOLEAN | This is Boolean value; it will have either true or false | TRUE OR FALSE |
| Address | Address\_id | DECIMAL (12) | Uniquely identifies each row in address table. Having row numbered 12 digits can accommodate billions of rows. | 140 |
| Address | Stree\_name | VARCHAR (128) | Allowing street name with 128 characters which is enough for a street name |  |
| Address | Street\_number | DECIMAL (7) | Street number with up to 7 digits is enough for street number | 55 |
| Address | Apartment\_num | VARCHAR (10) | Apartment number with up to 10 characters is allowed in this field which is enough for apartment number. This can be either number or characters or both that is why it is varchar data types | 56B,4,A |
| Address | City | VARCHAR (128) | City name with character number up to 128 is allowed in this field and is enough for the any city. | Boston |
| Address | State | VARCHAR (128) | State name with character number up to 128 is allowed in this field and is enough for the any state | Massachusetts |
| Address | Country | VARCHAR (128) | This field will allow a country name with up to 128-character name. usually, country name is not that long but I am allowing just in case. | United States |
| Address | Postal\_code | Varchar (128) | Postal code can have number and other character like dash, that is why it is included as varchar and giving 128 characters field | 02184-2345 |
| Dealership | Dealership\_id | DECIMAL (12) | Uniquely identifies each row of dealership table | 9 |
| Dealership | dealership\_name | VARCHAR (128) | I am allowing dealership name with up to 128 characters which is more enough for name. I am setting up upper limit higher so that in future in can accommodate a dealership with long name. | CarmaxDeal |
| Vender | Vender\_id | VARCHAR (12) | Uniquely identifies each row of vender table | 9 |
| Vender | Vender\_name | VARCHAR (128) | This field up to 128 characters for the name of vender who is supplying car to the dealer. | Honda |
| Vender | Location | VARCHAR (75) | This is the vender location either city or state where vender is headquartered. 75 characters are. enough | Boston |
| Department | Department\_id | DECIMAL (12) | Uniquely identifies each row of department table. Allowing a row numbered up to 12 digits which is huge number | 4 |
| Department | Department\_name | Varchar (128) | This field allow the name of the department, usually they are short, but I am allowing up to 128 character name just in case if needed. | HR,sales, sevice |
|  |  |  |  |  |

I feel I have included all necessary attributes which are needed for.

Carmaxdeal database. More information can be included but I think this is enough for this time.

For the **normalization** part, I will be doing normalization in places where data redundancy can occur one in vehicle entity where there are many purchases and attributes like color, make, model, and vehicle type can be repeated. Same things might happen to address as well. Similarly, payment method can be repeated. That is why I made separate entities. And those are update on my business rule as well. The normalized tables and full physical ERD with normalized tables are shown in figure below. There are few more attributes where we can further do nomination like in city and state part but I am not doing for this project because I think too many entity can slow down DBMS functions like while executing SQL scripts because of they need too many JOIN operations have to take place to run SQL queries. I do not find any partial, transitive dependencies. Also, all our determinants are candidate keys which means the tables are in BCNF.

Here is my Full DBMS Physical ERD with attributes, their association, specialization-generalization, and normalization.

Diagram

Description automatically generated

# Stored Procedure Execution and Explanations

1.

Populating address table using stored procedure.

When a new employee joins the company or new customer wanted to buy a vehicle, their address needs to be put in the system. For this I will be create transaction that create new address row related to an employee to address table. Here is the screen shot of the stored procedure and their execution.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

2. For customer table

When new customer wants to buy a vehicle, their information needs to put in the system. This causes interaction with database. Here the stored procedure and their execution while adding new customer in the system.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

3. Car sale

This is another use case. While selling a car, database with car\_sale table will be inserted with new row of vehicle info which is being sold. While selling a car table which tracks inventory also gets updated but here, I will be using stored procedure to update the Car\_sale table only. Here is the snapshot of the stored procedure and their execution.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

4. when dealership sells car, customer pays to dealership. While doing so, payment table need to be updated as well. Here is the snapshot of stored procedure and their execution.

Graphical user interface, text

Description automatically generated

Graphical user interface, text

Description automatically generated with medium confidence

# Question Identification and Explanations

1. What would be the gross margin of selling each car? this question is useful for the company to know what type of car has higher margin. This gives an idea to the company what type of vehicle must sell to get higher profits.
2. Which sale\_person doing better job as compared to their salary, which sale person need more motivation? This is another important question owner of the company might be looking for. If some employees need more hard work/incentive/motivation, it may be time to talk with them so that they could more motivated and increase revenue. Or may be company need alternative strategy to increase the revenue.
3. What is the Status of inventory, number of cars with make, model, type, and color: This is the another most important measure to check what are available to sell. This measure is required to know what need to be restock which prevent from running out of the model customer looking for.

# Query Executions and Explanations

1. **Checking gross margin on each car:**

Answer to the first question is to know the gross margin on each of the sold car. For this I just must join the vehicle table with sale table and compare the vender price versus sell price. Vender price of car from vender which is buy price. To know which car has high margin and which has low, I will be subtracting each car’s vender price from its sold price. The query I made for this question retrieves all the information about vehicles like make, type, model, color, and condition so that it is easy to figure it out that what make, mode, color and condition have higher margin and lower margin selling vehicles. Also, all the prices are in dollars amount hence dollars symbols are added to all price values.

Graphical user interface, application

Description automatically generated with medium confidence

2.**Check who is doing better among sale person:** Attributes for my subtypes are not very complex. That is why I could not make more complex queries. However, this is an important question every owner/manager concern about employee performance regarding generating revenue. For this question, I will be using just sales department and salesperson subtype. Though this is fictional data it simulates the real business data. Here, sale person Nyoun is comparatively doing better than other as compared to their salary. Rajan is not doing good as compared to other. It may me time to find the strategy to motivate Rajan so that he gets motivated and more involved. Also, Also, company might need to spend to advertisement so that they could sell more vehicles. This is simple statistics but useful to the business for informed decision making. The SQL queries and their results are given below.

Graphical user interface

Description automatically generated

**3.Checking inventory** is another important major to take by every car dealer company to know car’s stock situation. Which type of car sell fast, which is being in stock long time etc. For this purpose, I can just join the vehicle table with make, mode, vehicle\_type and color table and use group by method to group all vehicle based on their make, model, color, and count number of each vehicle present in the stock. Also, I want to order the available vehicle number increasing order so that I can check low number in stock on the top. I will be used synthetic data I have created which simulate the real data, but it is not actual data.

Table

Description automatically generated

# Index Identification and Creations

Identification of index:

1. **Primary keys:** Primary keys of each of the table are already indexed, here are the list of the primary keys.
2. V\_type. type\_id
3. Make.make\_id.
4. Model.model\_id
5. Color.color\_id
6. Address.address\_id
7. Vender.vender\_id
8. Dealership.dealerhip\_id
9. Department.department\_id
10. Vende\_dealership. vender\_dealership\_id
11. Employee.employee\_id
12. Customer.customer\_id
13. Vehicle.vehicle\_id
14. Payment\_type. payment\_type\_id
15. Car\_sale.car\_sale\_id
16. Payment.payment\_id
17. **Foreign keys:** As all foreign keys need indexing, I will be listing the foreign keys below in table from different entities with its unique or not unique and with some explanations.

|  |  |  |
| --- | --- | --- |
| Column | Unique? | Description |
| Department.dealership\_id | Not unique | Each dealership can have many departments. Hence it is not unique |
| Vender\_dealership.vender\_id | Not unique | Each vender can have many dealerships. Hence not unique |
| Vender\_dealership.dealerhip\_id | Not unique | Each dealership can be linked to many venders. Hence not unique. |
| Employee.deparment\_id | Not unique | Each department can have many employees. Hence not unique. |
| Employee.address\_id | Not unique | Each address can have from many employees. E.g. employees from a family living in a house |
| Mechanic.employee\_id | Not unique | Each employee can be a mechanic or not mechanic |
| Sale\_person.employee\_id | Not unique | Each employee can be a sale\_person or not sale person. |
| Customer.address\_id | Not unique | Each address can have from many customers. Hence not unique |
| Vehicle.vender\_dealership\_id | Not unique | Each vender-dealership link can have many vehicles. Hence not unique. |
| Vehicle.make\_id | Not unique | Each make can have many vehicles. Hence not unique |
| Vehicle.model\_id | Not unique | Each Model can have many vehicles. Hence not unique |
| Vehicle.color\_id | Not unique | Each color can have many vehicles. Hence not unique |
| Vehicle.type\_id | Not unique | Each type can have many vehicles. Hence not unique |
| Car\_sale.customer\_id | Not unique | Each customer can involve in many cars sale. Hence not unique |
| Car\_sale.employee\_id | Not unique | Each employee can involve in many cars sale. Hence not unique. |
| Car\_sale.vehicle\_id | Not unique | Each vehicle can be sold many times in a life of that vehicle. Not unique |
| Payment.sale\_id | Not unique | Each sale can have many payments. |
| Payment.payment\_type\_id | Not unique | Each payment\_type can have many payments. Hence not unique |

# Query Driven index placement:

1. It possible that searching a vehicle based on its price, so I will include **vehicle. price** to be indexed. It would be non-unique index as many vehicles can have same price.
2. It is possible that searching a vehicle based on received date of the vehicle. So, I will include **vehicle. received\_date** to be indexed. It would be non-unique index as there could be many vehicles received in the same day.
3. It is possible that searching a vehicle based on vender\_price, so I will include **vehicle. vender\_price** to be indexed. It would be non-unique index as many vehicles can have same vender\_price.
4. Another column that can be used to in query is sale\_price, so I will include **car\_sale. sale\_price** to be indexed. As there could be many vehicles whose sale\_price is same. Hence it will be a non-unique index.
5. It is possible that searching car sale based on the sale date (range of sale\_date), so I will include **car\_sale. sale\_date** to be indexed as well. It would be non-unique as there could be many vehicles sold in the same date.
6. Payment amount or range of payment amount from payment table can be used for payment related inquiry. So, I will be using **payment. payment\_amount** to be indexed. And it will be non-unique as there could be multiple payment with same amount.
7. Another column use for the query is payment\_date, so I will be using **payment. payment\_date** to be indexed. It will be non-unique index as there could be many payments in the same date.

Creating Indexes on all the column identified above which are given in a series of snapshot here.

1.

Graphical user interface, text, application, email

Description automatically generated

2.

Graphical user interface, application

Description automatically generated

3.

Graphical user interface, application

Description automatically generated

4.

Graphical user interface, application

Description automatically generated

# Summary and Reflection

My database for car dealership named CarMaxDeal” which stores information about cars inventory, Sales information, Customer information, Service/maintenance information, Employee information, information about the service type customer getting. Customer can use web site to check all available cars and its features to buy. Dealership has their own application to track all the activities involved in car purchase or servicing a car at the dealerships. Database must support a person using website and searching for their favorite car to buy. Also, Database must support generating report for analysis for example daily, weekly, monthly, and yearly report analysis.

**In this Iteration2,** to make more entities I have added some use cases for my database design. I came up with 14 business rules and I have mentioned 10 entities, their relationships and plurality. I did make ERD for all those entities and their relationships.

**In this Interation3,** goal of this iteration is to convert conceptual ERD to DBMS Physical. There are three entities which need specialization-generalization representation namely Vehicle, Department and Employee. Each of these entities act as supertype entities which is also known as abstract entity and each of these have their own subtypes which is also called as specialized entities. Once supertype entity is identified, I made a specialized-generalized ERD also known as EERD. Next step is to make DBMS physical ERD. This is based on one to one, one to many and many to many mapping properties. Appropriate mapping is done to avoid data inconsistencies, data redundancies and/or other type of data anomalies.

**In this iteration4**, I corrected my design according to the comment I get from my facilitator. Accordingly, I updated some business rules and conceptual ERD. After that, I have listed all the attributes that will be used in the tables, their data types. The reason behind the data types and their upper limit length and some examples what I am expecting as field data values. Then after I tried to normalize the tables so that it would prevent from data redundancies and other data anomalies. For this I found few attributes that can generate data redundancies that is why I made separate tables for them. For example, vehicle color, make, model, address, and payment methods are the attributes I took out from its old tables and made separate tables. Also, I have created the tables for all the entities based on their relations, association, constraints, and datatypes. Which are saved as .sql file.

**In this Iteration 5**, First I have corrected some of the contents after I get feedback from my facilitator. Next, based on the tables created on iteration4, I create stored procedures for the different tables and insert the data transactionally. I also use insert into command to insert the data for some of the tables. Also, I used stored procedures to insert data values just calling the procedures (not transactionally). After that some of the useful questions are made to get information from our database to check how our database working. My questions are, what would be the status of inventory? and what would be the gross margin selling each car? What is the sale person performance status on selling vehicle? I create the query to get those answers. In the next step is to create the index of the selected columns. I have identified primary keys, foreign keys and columns that are used to get the useful information about the database. As primary keys are indexed already, I created index on all foreign keys and some of the identified columns. The reason behind creating index on columns is to speed up the reading the tables.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Aspect** | **What is Measured** | **A+ Excellent** | **B Good** | **C Fair/Satisfactory** | **D Insufficient** | **F Failure** | **Letter Grade** |
| **Stored Procedures (20%)** | This is a measure of the quality of the construction and correctness of the syntax of the stored procedures, and the number of stored procedures created. Excellent stored procedures implement useful and complete transactions, are reusable through use of parameters, are robust with the inclusion of parameter validation, and compile and are executable. Excellent solutions define at least two stored procedures. | Excellent construction Entirely correct syntax At least two stored procedures | Good construction Mostly correct syntax At least two stored procedures | Satisfactory construction Somewhat correct syntax At least one stored procedure | Insufficient construction Mostly incorrect syntax At least one stored procedure | No stored procedures defined *or* Unacceptable construction Entirely incorrect syntax | A- |
| **Questions (10%)** | This is a measure of how essential the questions are to your database, as well as the number of questions provided. Excellent solutions have at least three questions defined that represent very important use of the data, where the answers to the questions are critical to those using the database. | Entirely essential At least three questions | Mostly essential At least three questions | Useful but secondary At least two questions | Mostly unnecessary At least one question | No questions given *or* Entirely unnecessary | A+ |
| **Queries (20%)** | This is a measure of how correctly the queries answer the questions and conform to the project requirements, the correctness of the syntax, and the number of queries defined. Excellent solutions exhibit all of the following properties. At least three queries produce results that completely answer the questions. Each query meets the requirements given in the project instructions. The columns selected give all needed information for those asking the question. No unneeded columns and rows are present. The syntax of the queries is entirely correct and could be executed in a modern relational database without modification. | Entirely correct answers and conformity Entirely correct syntax At least three queries | Mostly correct answers and conformity Mostly correct syntax At least three queries | Somewhat correct answers and conformity Somewhat correct syntax At least two queries | Mostly incorrect answers and conformity Mostly incorrect syntax At least one query | No queries defined *or* Entirely incorrect answers and conformity Entirely incorrect syntax | A |
| **Index Identifications (20%)** | This is a measure of how beneficial the indexes are to your database, how accurate the uniqueness choices are, and the coverage of the indexes. Excellent solutions exhibit all of the following properties. The query driven fields selected would benefit significantly with indexes. The unique/not unique choices are entirely accurate for all indexes. All primary and foreign key indexes have been identified, and at least three query driven indexes have been identified. | Entirely beneficial Entirely accurate Full coverage | Mostly beneficial Mostly accurate Good coverage | Somewhat beneficial Somewhat accurate Partial coverage | Mostly unhelpful Mostly inaccurate Insufficient coverage | No indexes identified *or* Entirely unhelpful Entirely inaccurate Unacceptable coverage | A+ |
| **SQL Script (10%)** | This is measure of how accurately the SQL script implements the designed database structure, and the correctness of the SQL syntax. Excellent scripts exhibit all of the following properties. The SQL script includes the table names, attribute names, attribute datatypes, and constraints exactly as defined in the DBMS physical ERD, as well as the identified sequences and indexes. Only designed elements are present in the script. The syntax of the SQL script is entirely correct and could be executed in a modern relational database without modification. | Entirely accurate Entirely correct | Mostly accurate Mostly correct | Somewhat accurate Somewhat correct | Mostly inaccurate Mostly incorrect | The SQL script is missing *or* The SQL script is generated by a tool without significant modification *or* Entirely inaccurate Entirely incorrect | A+ |
| **Overall Presentation (10%)** | This is a measure of how well your design and implementation choices are supported with explanations, as well as the quality of your documentation organization and presentation. | Excellent support Well organized and presented | Good support Organized and presentable | Partial support Somewhat organized and presented | Mostly unsupported Mostly disorganized presentation | No explanations Entirely disorganized presentation | A+ |
| **Prior Work (10%)** | This measures how well any issues from prior iterations have been improved in order to provide a frame of reference for this iteration. | Completely improved  *or* No improvement necessary | Mostly improved | Somewhat improved | Mostly not improved | No improvements | A+ |
| **Preliminary Grade:** | 97.6 | **Entities Deduction:** At least 10 required at the conceptual level At least two subtypes required 3 point deduction for each missing | 0 | **Lateness Deduction:** 5 points per day4 days maximumContact your facilitator for any exceptions | 0 | **Iteration Grade:** | 97.6 |