**Database Design Guide**

This guide will help the student to create a database on the Farmer’s Buddy System. It will help to manage the below functionalities.

* User Details
* Product Details
* Cart
* Order Details
* Payment
* Information about farming related products
* Schemes
* Feedback

We will use MySQL as the DBMS to create the database and its related operations.

**1. Introduction to MySQL**

MySQL is an open-source relational database management system (RDBMS) that uses structured query language (SQL) to manage and manipulate data in a database. It is widely used for various applications, from small web applications to large enterprise systems.

MySQL's key features include:

* Scalability: Capable of handling large amounts of data and concurrent connections.
* Flexibility: Supports various data types and storage engines.
* Performance: Optimized for speed and efficiency.
* Reliability: Known for its stability and robustness.

**2. Installation of MySQL**

MySQL can be installed on various operating systems, including Windows, macOS, and Linux. Here are the general steps to install MySQL:

**Windows:**

* Download the MySQL installer from the official website.

<https://dev.mysql.com/downloads/installer/>

* Run the installer and follow the on-screen instructions.
* Choose the installation type (Typical, Complete, or Custom). Recommended Custom.
* Set a root password for the MySQL server.

**3. E-R Diagram (ERD)**

An Entity-Relationship Diagram (ERD) is a visual representation of the data model that shows the entities, attributes, relationships between entities, and cardinality. ERDs are commonly used in database design to help developers and stakeholders understand the structure and relationships within a database.

**Identify Entities**

* Start by identifying the main entities in your system. These are the objects or concepts about which you want to store data.
* Each entity should correspond to a table in your database.

**Define Attributes**

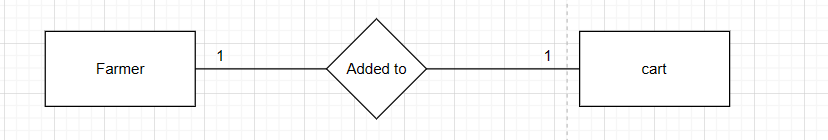
* For each entity, list the attributes (properties or fields) that describe it.
* These attributes will become columns in the corresponding database table.

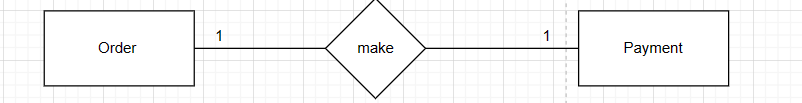
**Identify Relationships**

* Determine how entities are related to each other. There are three types of relationships: one-to-one (1:1), one-to-many (1:N), and many-to-many (N:M).
* Represent these relationships using lines connecting the entities.

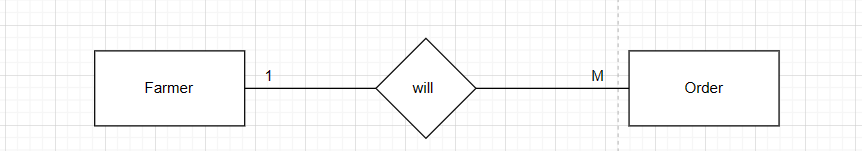
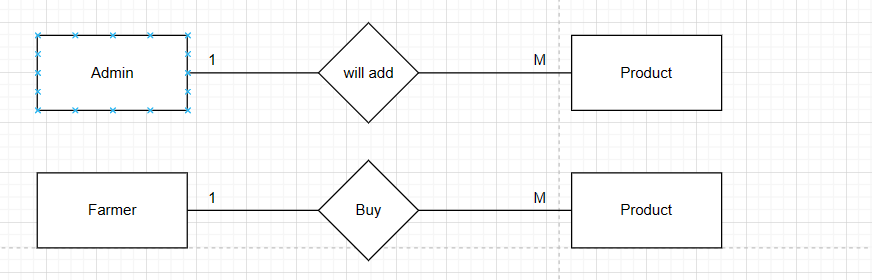
Let’s see a few examples of relationships:

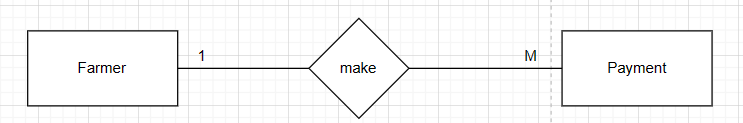
**One to One**

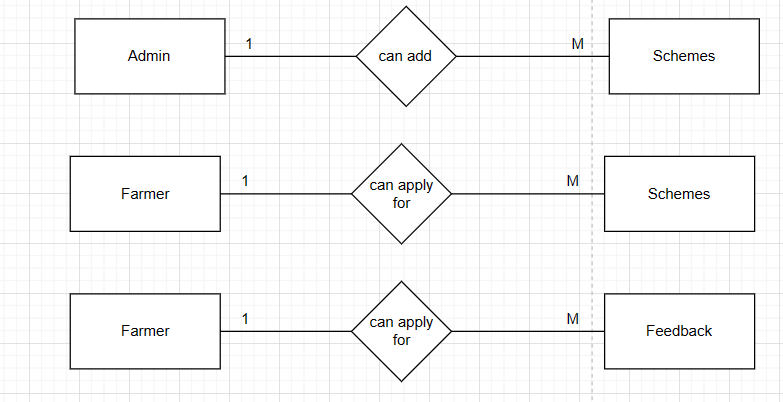




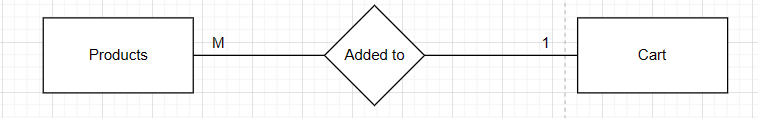
**One to Many**

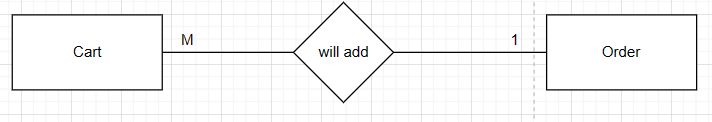






**Many to One**





**Many to Many**

**Cardinality Notation**

Cardinality represents the number of times an entity of an entity set participates in a relationship set. Or we can say that the cardinality of a relationship is the number of tuples (rows) in a relationship.

* Use notation (such as Crow's Foot Notation or Chen Notation) to indicate the cardinality of each relationship.
* Cardinality describes how many instances of one entity are related to how many instances of another entity.
* Common notations include:
* One (1)
* Zero or one (0..1)
* Many (N)
* Zero or many (0..N)

**Optional:**

**Add Attributes and Constraints**

* Include additional information in your ERD, such as primary keys, foreign keys, and constraints (e.g., unique constraints).

**Create the Diagram**

* Use specialized diagramming software or tools (e.g., Lucidchart, draw.io, or even pen and paper) to create your ERD.

**Refine and Review:**

* Review your ERD with stakeholders and team members to ensure it accurately represents the data model and relationships. Make any necessary refinements.

Let’s identify the entities of the Student management system

1. Student
2. Course
3. Instructor
4. Enrollment
5. Score
6. Feedback

\*\*\* Now let’s identify the attributes and relationships of each entity for the Student Management System.

**Admin**

* **Attributes:**

Admin\_id

Password

* **Relationships:**

One **Admin** can add many products and schemes.(One-to-Many)

**Farmers**

* **Attributes:**

Farmer\_ID (Primary Key)

Farmer\_Name

Password

Email

Phone\_No

* **Relationships:**

One **Farmer** can Register them self Only Ones. (**One-to-One**)

**Products**

* **Attributes:**

P\_Id (Primary Key)

P\_Name

Descpt

Price

* **Relationships:**

One **Farmer** can buy many **Products** (**One-to-Many**)

**Cart**

* **Attributes:**

Farmer\_Id (Foreign key)

P\_Id (Foreign Key)

P\_Name

Price

Quantity

* **Relationships:**

One **Farmer** can have **Cart** (**One-to-Many**)

**Orders**

* **Attributes**:

Order\_ID (Primary Key)

Farmer\_ID (Foreign key)

Name

Address

Amount

* **Relationships:**

One **Farmer** can **Order many Products  (One-to-Many)**

**Payment**

* **Attributes**:

Payment\_ID (Primary Key)

Farmer\_ID (Foreign key)

Order\_ID (Foreign key)

Payment\_Type

Cvv

Amount

Cardholder\_Name

Expiry\_Date

* **Relationships:**

One **Farmer** can do Payment many times.**(One-to-Many)**

**Schemes**

* **Attributes**:

Schemes\_ID (Primary Key)

Accountholder\_Name

Name

Bank\_Name

Account\_No

Farmer\_ID (Foreign key)

* **Relationships:**

One **Farmer** can apply for many Schemes.**(One-to-Many)**

**Feedback**

* **Attributes**:

Feedback\_No (Primary Key)

Farmer\_ID (Foreign key)

Phone\_No

Quality

Suggestion

* **Relationships**:

  One **Farmer** will maps to Many **Feedbacks** (**One-to-Many**)

**Table Structure**

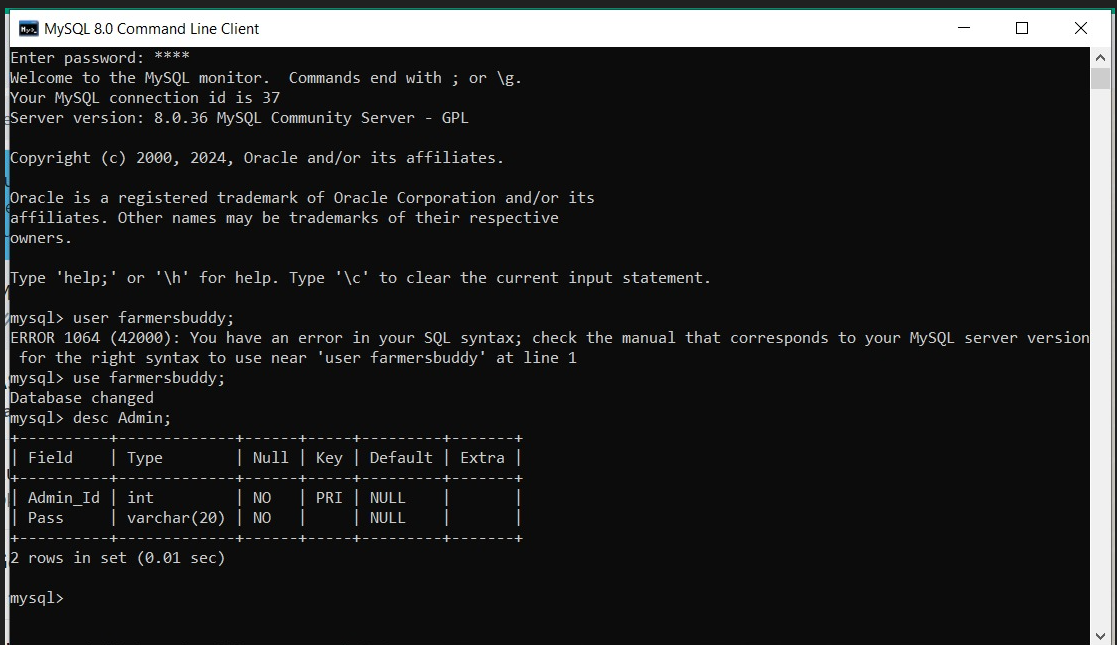
1. **Admin**

create table Admin(

Admin\_Id int(10) primary key,

Pass varchar(20) NOT NULL

);



1. **Farmer**

create table Farmer(

Farmer\_Id int(10) primary key,

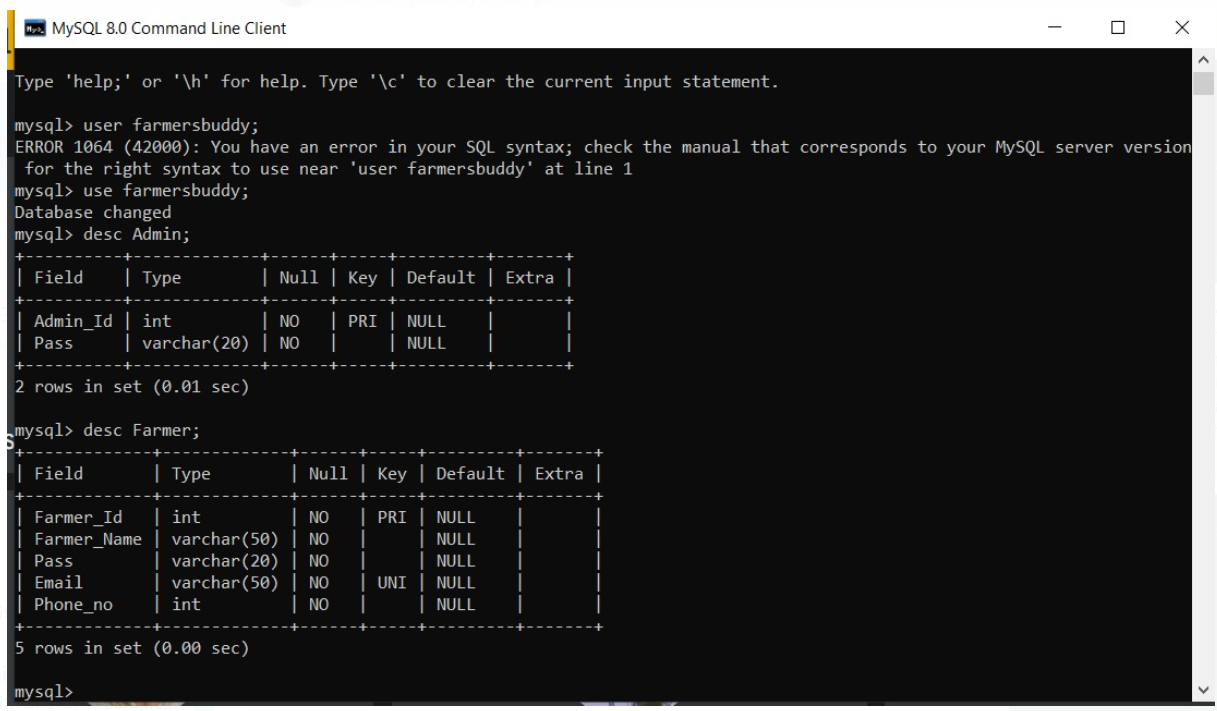
Farmer\_Name varchar(50) NOT NULL,

Pass varchar(20) NOT NULL,

Email varchar(50) NOT NULL unique,

Phone\_no int (10) NOT NULL

);



1. **Product**

create table Product(

P\_Id int(10) primary key,

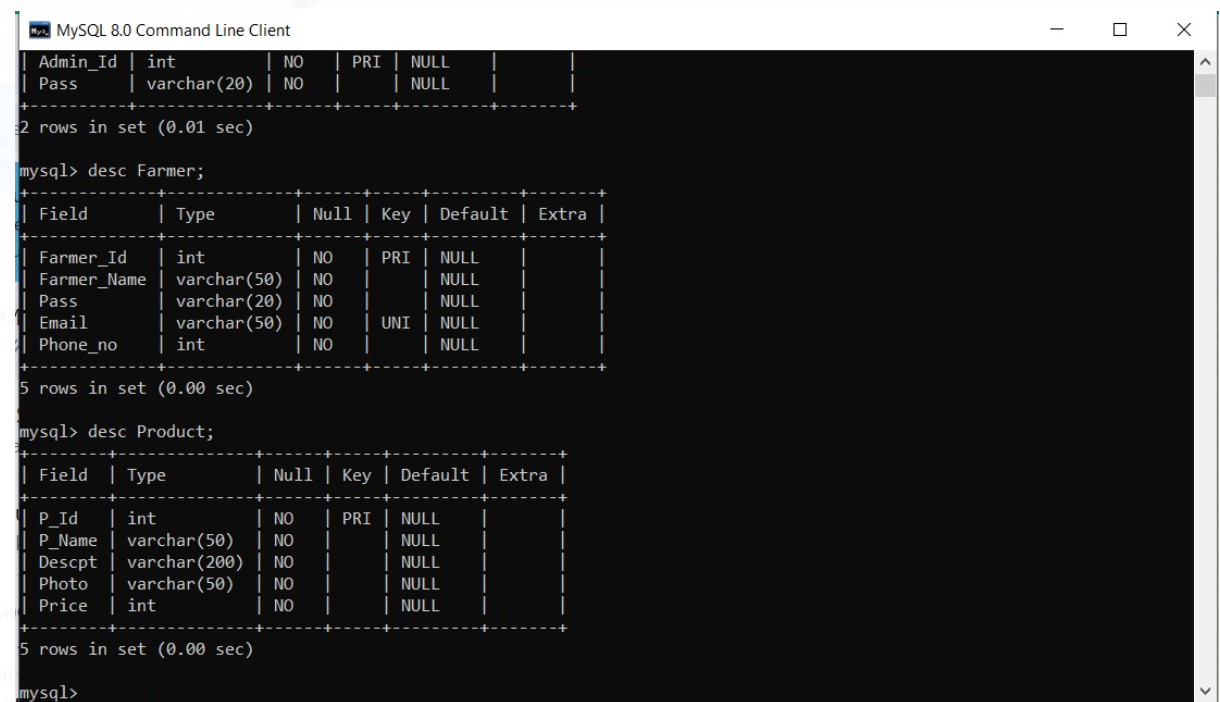
P\_Name varchar(50) NOT NULL,

Descpt varchar(200) NOT NULL,

Photo varchar(50) NOT NULL ,

Price int NOT NULL

);



1. **Cart**

create table Cart(

Farmer\_Id int(10) NOT NULL,

P\_Id int(10) NOT NULL,

P\_Name varchar(50) NOT NULL,

Descpt varchar(200) NOT NULL,

Price int NOT NULL,

Quantity int NOT NULL,

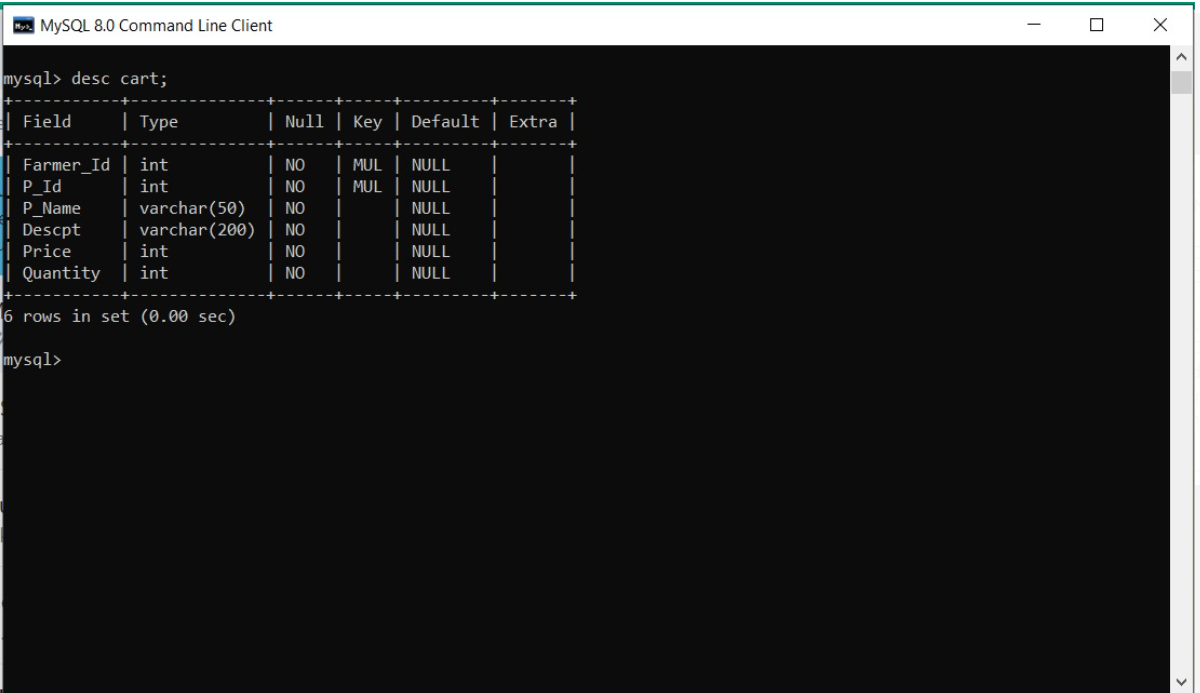
constraint FK\_ProductID foreign key (P\_Id)

references Product(P\_Id),

constraint FK\_CustomerID foreign key (Farmer\_Id)

references Farmer(Farmer\_Id)

);



1. **Order**

create table Orders(

Order\_Id int(10) primary key,

Phone\_no int(10) NOT NULL,

Name varchar(50) NOT NULL,

Address varchar(200) NOT NULL,

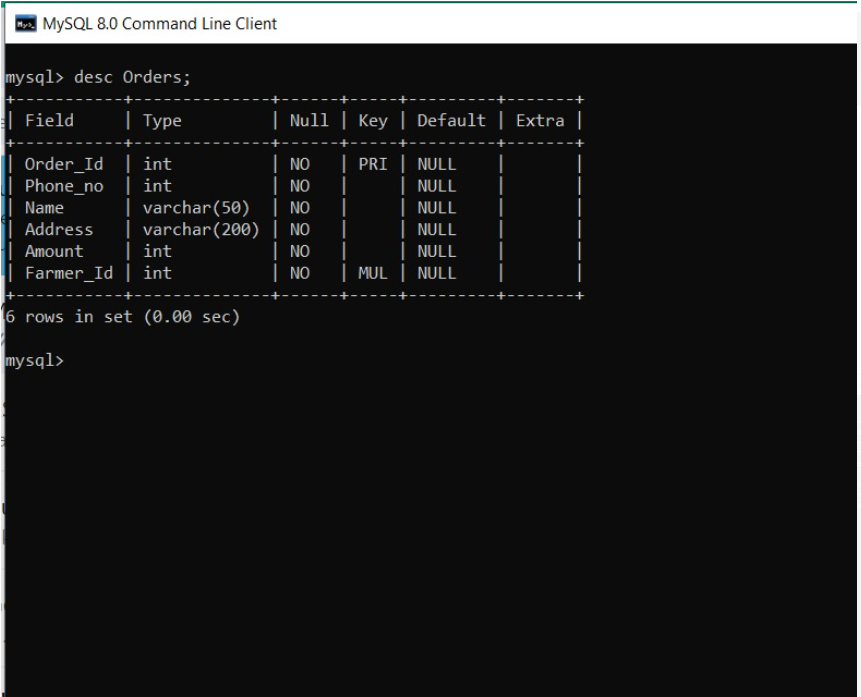
Amount int NOT NULL,

Farmer\_Id INT NOT NULL,

constraint FK\_UserID foreign key (Farmer\_Id)

references Farmer(Farmer\_Id)

);



1. **Payment**

create table Payment(

Farmer\_Id int(10) NOT NULL,

Cardholder\_Name varchar(50) NOT NULL,

Amount int NOT NULL,

Cvv int(3) NOT NULL,

Expiry\_Date varchar(20) NOT NULL,

Order\_Id int(10) NOT NULL,

Payment\_Id int(20) NOT NULL,

Payment\_Type varchar(20) NOT NULL,

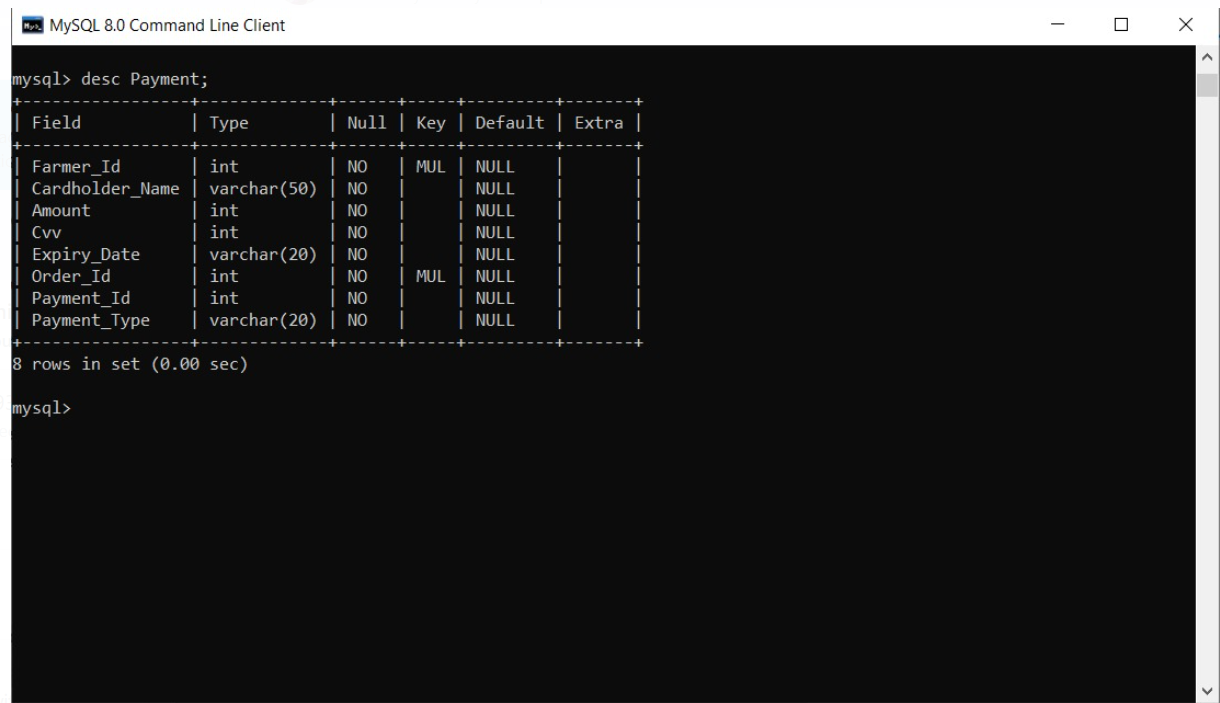
constraint FK\_OrderID foreign key (Order\_Id)

references Orders(Order\_Id),

constraint FK\_UsersID foreign key (Farmer\_Id)

references Farmer(Farmer\_Id)

);



**7.Schemes**

create table Schemes(

Farmer\_Id int(10) NOT NULL,

Accountholder\_Name varchar(50) NOT NULL,

Name varchar(50) NOT NULL,

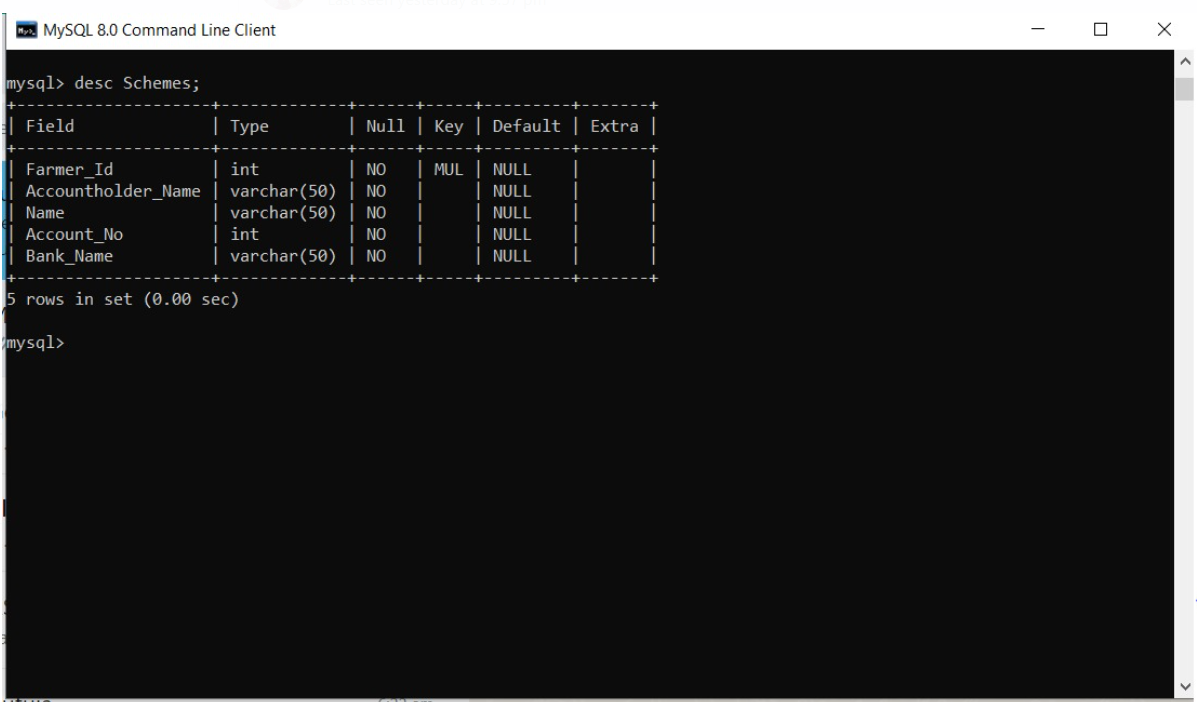
Account\_No int(14) NOT NULL,

Bank\_Name varchar(50) NOT NULL,

constraint FK\_SchemesID foreign key (Farmer\_Id)

references Farmer(Farmer\_Id)

);



**8.Feedback**

create table Feedback(

Feedback\_No int(10) primary key,

Farmer\_Id int(10) NOT NULL,

Phone\_no int(10) NOT NULL,

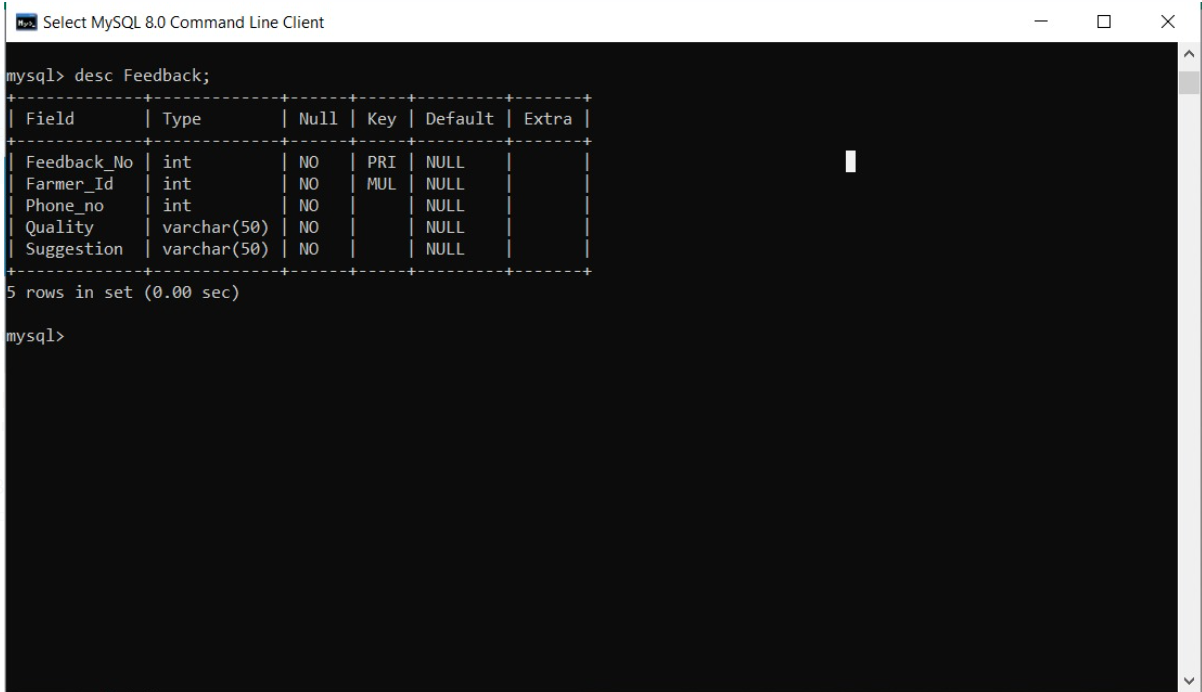
Quality varchar(50) NOT NULL,

Suggestion varchar(50) NOT NULL,

constraint FK\_FeedbackID foreign key (Farmer\_Id)

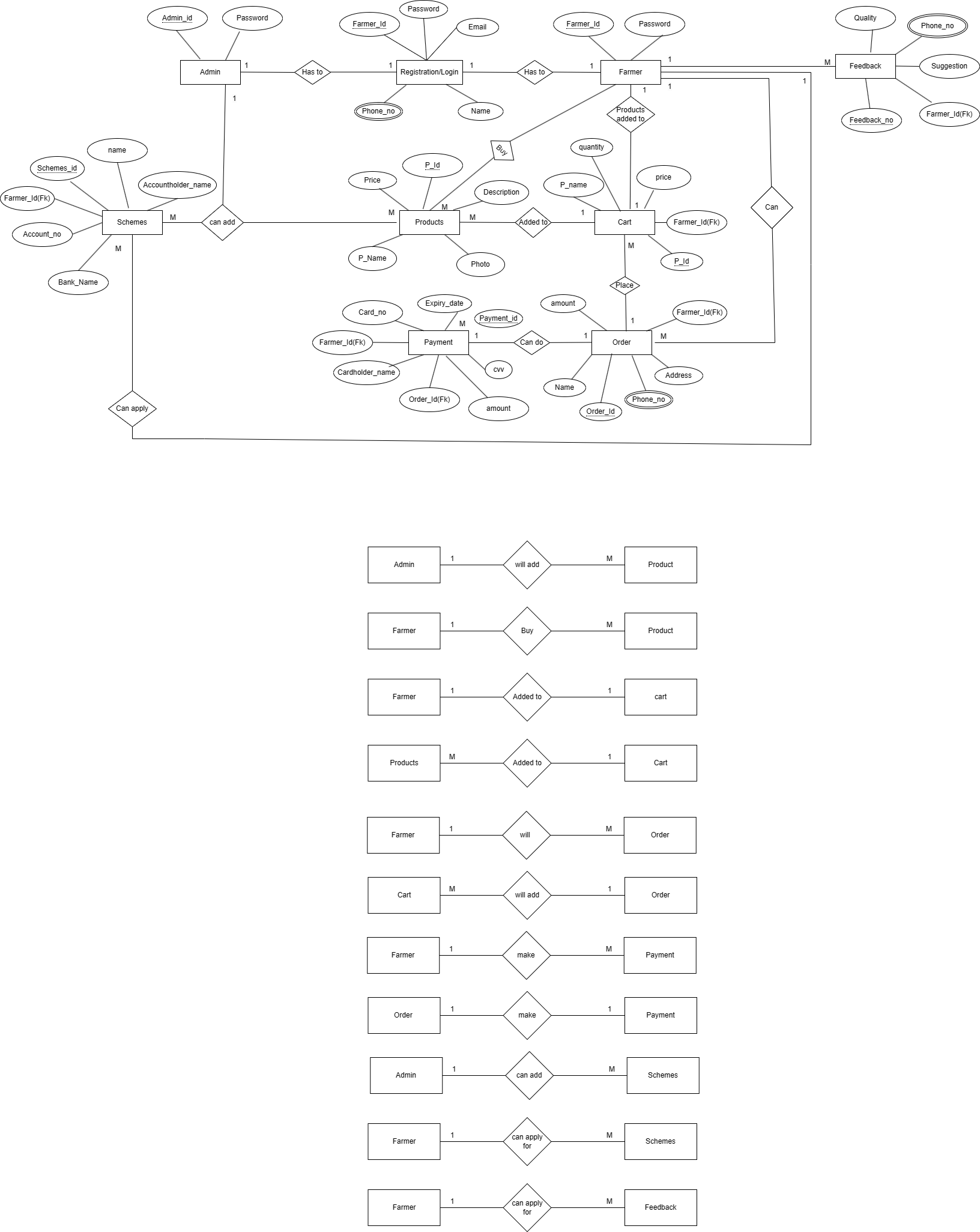
references Farmer(Farmer\_Id)

);



Now, let’s create the ER diagram to visually represent the entities and relationships.

**ERD Diagram**



**In this ERD:**

* Students can enroll, and each course can have multiple students, creating a many-to-many relationship.
* The Enrollment entity serves as a bridge table between Student and Course entities to represent this relationship.
* Multiple courses can be taught by one Instructor (many-to-one relationship).
* Each Instructor can teach multiple courses (one-to-many relationship).
* A student can give multiple feedbacks
* Student may have scores of multiple courses

**4. Creating a Database**

Using MySQL server, create a new database for your student management system. You can do this with SQL commands or through the graphical interface.

*CREATE DATABASE StudentManagementSystem;*

**5. Using a Database**

Before performing any operations on a database, you need to select it using the USE statement:

*USE StudentManagementSystem;*

**6. Creating the tables for each entity**

**PN:** Ideally no data should be deleted from any tables. You can use an additional column to set the status of that record to ‘Active/Inactive’, etc. Or you can use an Archive table to move the unnecessary records out of the main table.