Chapter 1

Problem Understanding

In the agricultural sector, farmers struggle to reach a wider market and effectively market their produce, while customers face difficulties accessing diverse products and verifying their quality. This leads to missed sales opportunities for farmers and dissatisfaction among customers, exacerbating market inefficiencies. Fragmented communication channels and lack of standardized procedures further complicate transactions, fostering an environment of distrust. Addressing these challenges requires a holistic solution that enhances accessibility, streamlines communication, and fosters trust and transparency in the agricultural marketplace.

Key features of the system include:

1. User Registration and Profile Management: Customers sign up using their username and password, generating a unique customer ID in the database. Farmers register with their personal details, including name, age, gender, address, Aadhar number, and phone number, creating a farmer registration ID in the database.

2. Farming Type Selection: Farmers select the type of farming associated with their products during the registration process, allowing for categorization and organization of agricultural offerings.

3. Product Listing and Management: Farmers can add their agro products to the platform by providing product names, descriptions, and prices, facilitating effective showcasing to potential buyers.

4. Purchase Requests via Email: Customers can initiate purchase requests by clicking on the purchase button, which automatically generates an email. The sender's email is the customer's email, and the receiver's email is the farmer's email ID, streamlining communication between buyers and sellers.

5. Secure Transaction Processing: The system ensures secure and reliable payment processing for transactions between buyers and sellers, maintaining the integrity of financial transactions.

These key features enable seamless interaction between farmers and customers, streamline product management, and facilitate transparent communication and transactions within the agricultural marketplace.

Identification of Entity and Relationships

Entities:

1. Users: Customers and Farmers are the primary users of the system, each with distinct roles and functionalities.
2. User Details: User details encompass essential information such as username, password, personal demographics (name, age, gender), contact details (address, phone number, email), and unique identification (Aadhar Number for farmers).
3. Unique Identifiers: Automatic generation of a Customer ID and Farmer Registration ID facilitates unique identification and tracking of users within the system.
4. Farm Details: The farming type associated with each farmer provides categorization and organization of agricultural products.
5. Agro Products:* Product listings consist of key information including product

name, detailed description, and pricing, essential for effective showcasing and purchasing decisions.

6. Transactions: Purchase requests and email communication streamline the buying process, ensuring transparent and efficient transactions between buyers and sellers.

7. Communication Details: Sender's email (customer) and receiver's email (farmer) facilitate direct communication channels, enhancing interaction and negotiation capabilities within the system.

Attributes:

1. Customers:

- User ID: A unique identifier assigned upon registration, Username: Customer's chosen username for login, Password: Secure login credential, Email Address: Contact information for communication.

2. Farmers:

- Farmer Registration ID: Unique identifier generated upon registration, Name: Farmer's full name, Age: Farmer's age, Gender: Farmer's gender, Address: Farmer's residential address, Aadhar Number: Unique identification number, Phone Number: Contact number, Email Address: Communication channel.

3. Farming Type:

- Type ID: Unique identifier for each farming type, Farming Type Name: Description of the type of farming practiced.

4. Agro Products:

- Product ID: Unique identifier for each product, Product Name: Name of the agricultural product, Description: Detailed information about the product, Price: Cost of the product.

5. Transactions:

- Transaction ID: Unique identifier for each transaction, Purchase Request Details: Information regarding the requested purchase, including product details and buyer's information.

6. Communication Details:

- Sender's Email (Customer): Email address of the customer initiating communication, Receiver's Email (Farmer): Email address of the farmer receiving communication.

Construction of DB using ER Model for the project

The ER model for the Farm Management System includes entities like Customers, Farmers, Agro Products, and Transactions. Each entity has specific attributes capturing essential details such as IDs, names, and contact information. Relationships between entities, like "Customer-Buyer" and "Farmer-Seller," facilitate transactions and product ownership. Cardinality constraints define how entities relate to each other, ensuring accurate representation of the system's dynamics. The model serves as a blueprint for constructing a relational database, enabling efficient data management and retrieval within the Farm Management System.

Entity Identification:

In the database schema for the Farm Management System, the Customer entity is referred to as User. Each entity identified in the ER model corresponds to a table, with attributes of each entity becoming columns in their respective tables. For instance, the User entity translates to a User table with attributes like User_ID, Username, Password, and Email_Address, each becoming a column. Primary keys are identified for each table to ensure record uniqueness; for example, User_ID serves as the primary key for the User table. Relationships between entities are represented through foreign key constraints in the database schema, establishing connections between tables based on the associations defined in the ER model. Cardinality and participation constraints guide the establishment of one-to-one, one-to-many, and many-to-many relationships in the database schema. This systematic translation ensures the database accurately represents the structure and dynamics of the Farm Management System.

Normalization:

In the Farm Management System project, normalization plays a crucial role in structuring the database to optimize data storage and retrieval. Initially, the database schema would be organized into tables representing entities such as users, farmers, agro products, transactions, and communication details. Each table would have a primary key column to uniquely identify records, ensuring atomicity and eliminating duplicate rows. To achieve First Normal Form (1NF), attributes within each table would be organized to hold atomic values, and any repeating groups of columns would be eliminated. For instance, in the User table, attributes like Username and Email would each be stored in separate columns to avoid redundancy and ensure data integrity. Second Normal Form (2NF) would be pursued by ensuring that non-key attributes are fully functionally dependent on the primary key, preventing partial dependencies. Third Normal Form (3NF) would further refine the schema by removing any transitive dependencies between non-key attributes, promoting data consistency and facilitating efficient data manipulation. Ultimately, normalization in this project would result in a well-structured database schema that minimizes redundancy, dependency, and anomalies, laying the foundation for a robust and scalable Farm Management System.

Table Creation:

Tables for the Farm Management System project are created in SQL using the `CREATE TABLE` statement. The User table contains fields for user ID, username, email, and password. The Register table stores information about farmers, including their ID, name, gender, age, phone number, address, Aadhar number, farming type. The Farming table holds the types of farming practices. The AddAgroProducts table records agricultural products with fields for product ID, name, description, and price. The trig table logs actions performed by farmers, tracking trigger ID, farming ID, action type, and timestamp.

Indexing:

Indexing in the Farm Management System database involves creating indexes on specific columns to improve query performance and data retrieval efficiency. For example, indexes can be created on frequently queried columns such as User ID, Farmer ID, Product ID, and Action Type in the respective tables. These indexes facilitate faster data access by organizing the data in a structured manner, reducing the need for full-table scans when executing queries. Additionally, indexing can enhance the overall performance of the system by optimizing data retrieval operations and reducing query execution time. However, it's essential to carefully consider the columns to index and monitor the impact on database performance, as excessive indexing can lead to increased storage requirements and overhead.

Constraints and Validation:

Check constraints are added to enforce domain integrity and validate data input. Default values may be specified for certain attributes to provide initial values upon insertion.

Unique constraints are applied to ensure uniqueness of values in specific columns.

Views and Stored Procedures:

Views can be created to present data in a customized format, combining columns from multiple tables.

Stored procedures and functions may be implemented to encapsulate complex logic and operations.

Data Population:

Data population in the Farm Management System entails inserting records into the database tables through SQL INSERT statements, capturing user details like usernames, email addresses, and passwords, farmer information such as names, genders, ages, and contact details, as well as agricultural product listings including product names, descriptions, and prices. Additionally, farming types are populated to categorize agricultural practices, and trigger actions are

recorded to log farmer activities. This process ensures that the database is populated with comprehensive and accurate data, enabling the Farm Management System to function effectively for users and farmers alike.

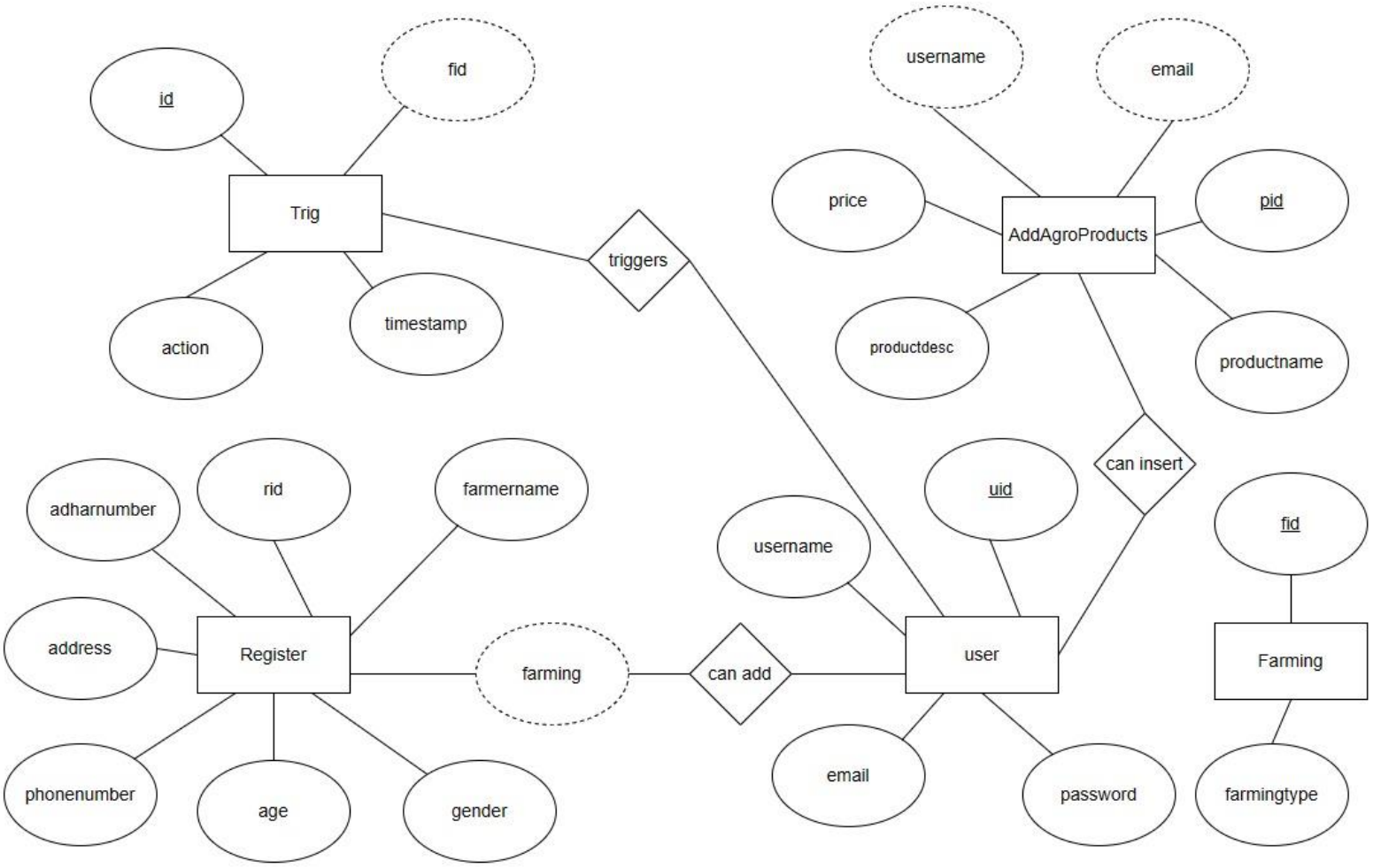
Testing and Validation:

Testing and validation in the Farm Management System involves verifying that the system functions correctly, meets requirements, and operates reliably. This process includes various types of testing such as unit testing, integration testing, and system testing. Unit testing ensures that individual components of the system, such as database queries and user interfaces, perform as expected. Integration testing validates interactions between different modules or subsystems to ensure seamless communication and functionality. System testing evaluates the system as a whole to confirm that it meets user requirements and performs its intended functions accurately. Validation involves comparing the system's outputs to expected results and ensuring that data integrity is maintained throughout the system. Through rigorous testing and validation processes, any defects or issues are identified and resolved, ensuring that the Farm Management System operates smoothly and effectively for users and farmers.

Documentation:

Documentation for the Farm Management System includes system requirements, design specifications, user manuals, and technical guides. System requirements outline necessary hardware and software, while design specifications detail the system's architecture and components. User manuals provide instructions for system use, and technical guides offer details for setup and maintenance. This documentation ensures clarity and usability across system development and implementation phases.

ER Diagram:


id
fid
Trig
action
timestamp
triggers
username
email
price
AddAgroProducts
pid
productdesc
productname
can insert
adharnumber
rid
farmername
address
Register
farming
can add
username
uid
user
fid
Farming
email
password
farmingtype
phonenumber
age
gender

CHAPTER 2

Design of Relational Schema

In crafting the database schema for the Farm Management System, meticulous attention is devoted to efficiently organizing and managing the system's data. Through a series of tables, each representing specific entities or relationships, data is structured and stored systematically. This structured approach ensures that essential information is readily accessible and effectively managed within the system. The schema is designed to accommodate diverse requirements while allowing for scalability and flexibility to adapt to evolving needs over time. By establishing a robust foundation through the database schema, the Farm Management System is equipped to handle data management tasks efficiently and support system functionalities with optimal performance.

Tables: Tables are the fundamental structures in a database, organizing related data into logical units. In our schema, the User table stores information about system users, including customers. The Register (Farmers) table captures details about farmers who are registered within the system. The Farming table holds various types of farming practices available in the system. The AddAgroProducts table records information about agricultural products added by farmers. The Trig table logs trigger actions performed by farmers.

Attributes: Attributes define the properties or characteristics of entities within a table. For example, in the User table, the UserID attribute uniquely identifies each user, while the Username, Email, and Password attributes store user login credentials and contact information.

Primary Keys: Primary keys uniquely identify each record within a table, ensuring data integrity and enabling efficient data retrieval. In our schema, attributes like UserID in the User table and FarmerID in the Register table serve as primary keys.

Foreign Keys: Foreign keys establish relationships between tables by referencing the primary key of another table. They enforce referential integrity and maintain consistency across related tables. For instance, the FarmerID attribute in the AddAgroProducts table references the FarmerID attribute in the Register table, linking agricultural products to their respective farmers.

Relationships: Relationships define connections between entities or tables, illustrating how data in one table relates to data in another. In our schema, relationships such as one-to-many and many-to-many are established between tables to represent the inherent associations within the Farm Management System..

Constraints: Constraints are rules enforced on data values within tables to maintain data integrity. For example, the NOT NULL constraint ensures that certain attributes cannot have NULL values, while the UNIQUE constraint ensures that each value in a column is unique.

Normalization: Normalization is the process of organizing data to minimize redundancy and dependency. It involves structuring tables to adhere to specific normal forms, such as First Normal Form (1NF), Second Normal Form (2NF), and Third Normal Form (3NF). Normalization reduces data duplication, improves data integrity, and facilitates efficient data management.

Creation of Database for Tables for the Project

1. User Table:

- The User table serves as a repository for storing information about system users, predominantly customers who interact with the platform.

Attributes:

- UserID (Primary Key): A unique identifier assigned to each user for identification purposes.
- Username: The chosen username by the user for authentication and identification.
- Email: The email address associated with the user's account, used for communication and login.
- Password: The encrypted password stored securely for user authentication during login.

2. Register Table (Farmers):

- The Register table functions as a registry for farmers registered within the system, housing their essential details.

Attributes:

- FarmerID (Primary Key): A unique identifier assigned to each farmer, facilitating individual farmer identification.
- FarmerName: The name of the farmer, providing personal identification.
- Gender: The gender of the farmer, representing male, female, or other.
- Age: The age of the farmer, providing demographic information.
- PhoneNumber: The contact number of the farmer, facilitating communication.
- Address: The address of the farmer, including location details.

- AadharNumber: The Aadhar number of the farmer, serving as a unique identification number in India.

- FarmingType: The type of farming practiced by the farmer, categorizing agricultural practices.

3. Farming Table:

- The Farming table houses a catalog of various types of farming practices available within the system, offering insights into agricultural diversity.

Attributes:

- FarmingID (Primary Key): A unique identifier assigned to each farming type, enabling distinct categorization.

- FarmingType: The specific type of farming practice, such as organic farming, hydroponics, or traditional farming.

4. AddAgroProducts Table:

- The AddAgroProducts table serves as a repository for information regarding agricultural products added by farmers for sale on the platform.

- Attributes:

- ProductID (Primary Key): A unique identifier assigned to each agricultural product, facilitating individual product tracking.

- ProductName: The name of the agricultural product, providing descriptive labeling.

- ProductDesc: A brief description of the agricultural product, offering additional insights.

- Price: The price of the agricultural product, indicating its value.

- FarmerID (Foreign Key): A reference to the farmer who added the product, establishing a link to the respective farmer.

5. Trig Table:

- The Trig table serves as a log for trigger actions performed by farmers within the system, facilitating activity tracking and monitoring.

Attributes:

- TriggerID (Primary Key): A unique identifier assigned to each trigger action, enabling individual action tracking.

- FarmerID (Foreign Key): A reference to the farmer who performed the action, linking the action to the respective farmer.

- Action: The type of action performed, categorizing actions as inserted, deleted, or updated.

- Timestamp: The timestamp indicating when the action was performed, offering temporal insights into farmer activities.

Code:

```
-- User Table
CREATE TABLE User (
    UserID INT PRIMARY KEY,
    Username VARCHAR(50) NOT NULL,
    Email VARCHAR(100) NOT NULL,
    Password VARCHAR(50) NOT NULL
);
-- Register Table (Farmers)
CREATE TABLE Register (
    FarmerID INT PRIMARY KEY,
    FarmerName VARCHAR(100) NOT NULL,
    Gender CHAR(1),
    Age INT,
    PhoneNumber VARCHAR(15),
    Address VARCHAR(255),
    AadharNumber VARCHAR(12),
```

```
    FarmingType VARCHAR(50),
    FOREIGN KEY (FarmingType) REFERENCES Farming(FarmingType)
);

-- Farming Table
CREATE TABLE Farming (
    FarmingID INT PRIMARY KEY,
    FarmingType VARCHAR(50) UNIQUE NOT NULL
);

-- AddAgroProducts Table
CREATE TABLE AddAgroProducts (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(255) NOT NULL,
    ProductDesc TEXT,
    Price DECIMAL(10, 2) NOT NULL,
    FarmerID INT,
    FOREIGN KEY (FarmerID) REFERENCES Register(FarmerID)
);
-- Trig Table
CREATE TABLE Trig (
    TriggerID INT PRIMARY KEY,
    FarmerID INT,
    Action VARCHAR(10) NOT NULL,
    Timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (FarmerID) REFERENCES Register(FarmerID)
);
```

CHAPTER 3

Complex Queries based on the concepts of constraints, sets, joins, views, Triggers and cursors

Constraints:

In the Farm Management System project, constraints play a crucial role in ensuring data integrity and enforcing business rules within the database tables. Each table is equipped with a primary key constraint, such as UserID, FarmerID, or ProductID, ensuring the uniqueness of each record. Additionally, attributes like Email and AadharNumber are marked as UNIQUE to prevent duplication of values. NOT NULL constraints are applied to mandatory fields like Username, Email, and ProductName, ensuring these fields always contain valid data. These constraints collectively maintain the consistency and reliability of the data stored in the system, facilitating smooth operations and accurate information management throughout the application.

Joins:

In relational databases, various types of joins are used to combine rows from different tables based on specified conditions.

1. Inner Join:

- Role: An inner join returns rows from both tables where the join condition is met. It retrieves only the matching records between the tables.
- Example Role: In the Farm Management System, an inner join between the Register (Farmers) table and the AddAgroProducts table can be used to fetch agricultural products added by registered farmers. This ensures that only products added by registered farmers are displayed to users.

2. Left Join (or Left Outer Join):

- Role: A left join returns all rows from the left table (the first table specified in the join) and the matched rows from the right table. If no matching rows are found in the right table, NULL values are included for the columns of the right table.

- Example Role: A left join between the User table and the Register (Farmers) table can be used to retrieve information about all users, including those who are not registered as farmers. This allows the system to display basic user details while indicating which users are farmers.

3. Right Join (or Right Outer Join)

- Role: A right join returns all rows from the right table (the second table specified in the join) and the matched rows from the left table. If no matching rows are found in the left table, NULL values are included for the columns of the left table.

- Example Role: A right join between the Register (Farmers) table and the AddAgroProducts table can be used to fetch information about all agricultural products, including those added by farmers who are not registered in the system. This ensures that all product data is included in the results.

4. Full Join (or Full Outer Join):

- Role: A full join returns all rows from both tables, matching rows where available and including NULL values for non-matching rows in either table.

- Example Role: A full join between the User table and the Register (Farmers) table can be used to retrieve a comprehensive list of all users and farmers registered in the system. This ensures that no user or farmer data is excluded from the results, providing a complete view of the system's user base.

By leveraging these different types of joins, the Farm Management System can retrieve and combine data from multiple tables in a flexible and efficient manner, enabling various functionalities such as product browsing, user authentication, and comprehensive data analysis.

Set Operations:

Set operations in relational databases allow for the manipulation of data sets to perform various operations such as union, intersection, difference, and Cartesian product. These operations can be useful in the Farm Management System for tasks such as data analysis, report generation, and data comparison.

1. Union (UNION):

- Union combines the results of two or more SELECT statements into a single result set, removing duplicates.
- Example: Union can be used to combine the lists of agricultural products added by different farmers into a single comprehensive list for browsing by customers.

2. Intersection (INTERSECT):

- Intersection returns only the common rows between two or more SELECT statements.
- Example: Intersect can be employed to identify agricultural products that are common between different categories of farming practices, helping in cross-referencing and analysis.

3. Difference (EXCEPT or MINUS):

- Difference returns rows from the first SELECT statement that are not present in the second SELECT statement.
- Example: Difference can be utilized to identify agricultural products added by one group of farmers but not by another, aiding in inventory management and

product tracking.

4. Cartesian Product (CROSS JOIN):

- Cartesian product returns all possible combinations of rows from two or more tables, resulting in a new table with a row for each possible combination.
- Example: Cartesian product can be used to generate a comprehensive report that includes every combination of farmers and agricultural products, providing insights into potential collaborations or market opportunities.

Set operations in SQL involve combining, filtering, or comparing data from different sources.

Views:

In the Farm Management System, views can serve as powerful tools for simplifying data access, enhancing security, and abstracting underlying table structures. For instance, a view can be created to consolidate information from multiple tables, presenting a unified view of agricultural products, farmers, and sales data for managerial analysis. By defining views with tailored column selections and filtering criteria, administrators can control access to sensitive information, ensuring that users only see the data relevant to their roles. Additionally, views can abstract away the complexity of underlying table relationships, providing a simplified interface for querying and reporting, thereby improving the overall usability and maintainability of the system.

Triggers:

In the Farm Management System, the Trig table serves as a crucial component for tracking and logging trigger events associated with specific actions within the system. Comprising four essential attributes, namely ID, FID (Farming Type ID), Action, and Timestamp, this table captures key details about each triggered event. The ID attribute serves as the primary key, ensuring unique identification for each logged event. FID references the farming type linked to the triggered

action, facilitating categorization and analysis. The Action attribute delineates the nature of the event, whether it involves a farmer being inserted, deleted, or updated. Finally, the Timestamp attribute records the date and time of each trigger event, providing chronological insights into system activities. Through the Trig table, the Farm Management System maintains a comprehensive log of farmer-related actions, enabling administrators to monitor system changes, conduct audits, and derive valuable analytics for system optimization and performance enhancement.

Cursors:

Cursors are database objects used to traverse the results of a query one row at a time.

Cursors are typically used in stored procedures or triggers to process individual rows returned by a query and perform complex operations on them.

Cursors provide a way to iterate over a result set and perform operations such as calculations, validations, or updates on each row.

While cursors can be useful in certain scenarios, they should be used judiciously as they can have performance implications, especially when dealing with large result sets.

Constraints:

This SQL command modifies the **Register** table by adding a **CHECK** constraint named **check_age_validity**. This constraint ensures that any **age** value entered into the database must be between 18 and 100, inclusive.

```
ALTER TABLE Register
ADD CONSTRAINT check_age_validity CHECK (age >= 18 AND age <= 100);
```

Sets and Joins:

Categorize farmers as active or inactive based on their recent transactions.

```
SELECT farmingtype FROM Farming
JOIN Register ON Farming.fid = Register.fid
WHERE status = 1
UNION
SELECT farmingtype FROM Farming
JOIN Register ON Farming.fid = Register.fid
WHERE status = 0;
```

Views:

This view provides customers a simplified interface to browse products without accessing sensitive information about farmers or detailed farm operations.

```
CREATE VIEW CustomerProductView AS
SELECT pid, productname, productdesc, price
FROM AddAgroProducts;
```

Triggers:

1: Trigger name: on insert Table: register

Time: after Event: insert

```
INSERT INTO trig VALUES(null,NEW.rid,'Farmer Inserted',NOW())
```

2: Trigger name: on delete

Table: register Time: after Event: delete

Definition: INSERT INTO trig VALUES(null,OLD.rid,'FARMER DELETED',NOW())

3: Trigger name: on update

Table: register Time: after Event: update

Definition: INSERT INTO trig VALUES(null,NEW.rid,'FARMER UPDATED',NOW())

Cursors:

Cursors in database management systems are programming constructs used to iterate over the result sets of a query. In the context of the Farm Management

System, cursors can be employed to navigate through data retrieved from the database in a systematic manner. For instance, a cursor can be utilized to fetch and process each record of agricultural products added by farmers, enabling tasks such as calculating total sales, updating inventory levels, or generating reports. Cursors offer fine-grained control over data traversal, allowing for sequential access to query results, row-by-row processing, and execution of custom logic for each fetched record. While cursors provide flexibility in data manipulation, they should be used judiciously as they may incur performance overhead, particularly when dealing with large result sets. Thus, in the Farm Management System, cursors can be employed selectively for tasks that require granular data processing and manipulation, ensuring efficient database operations and optimized system performance.

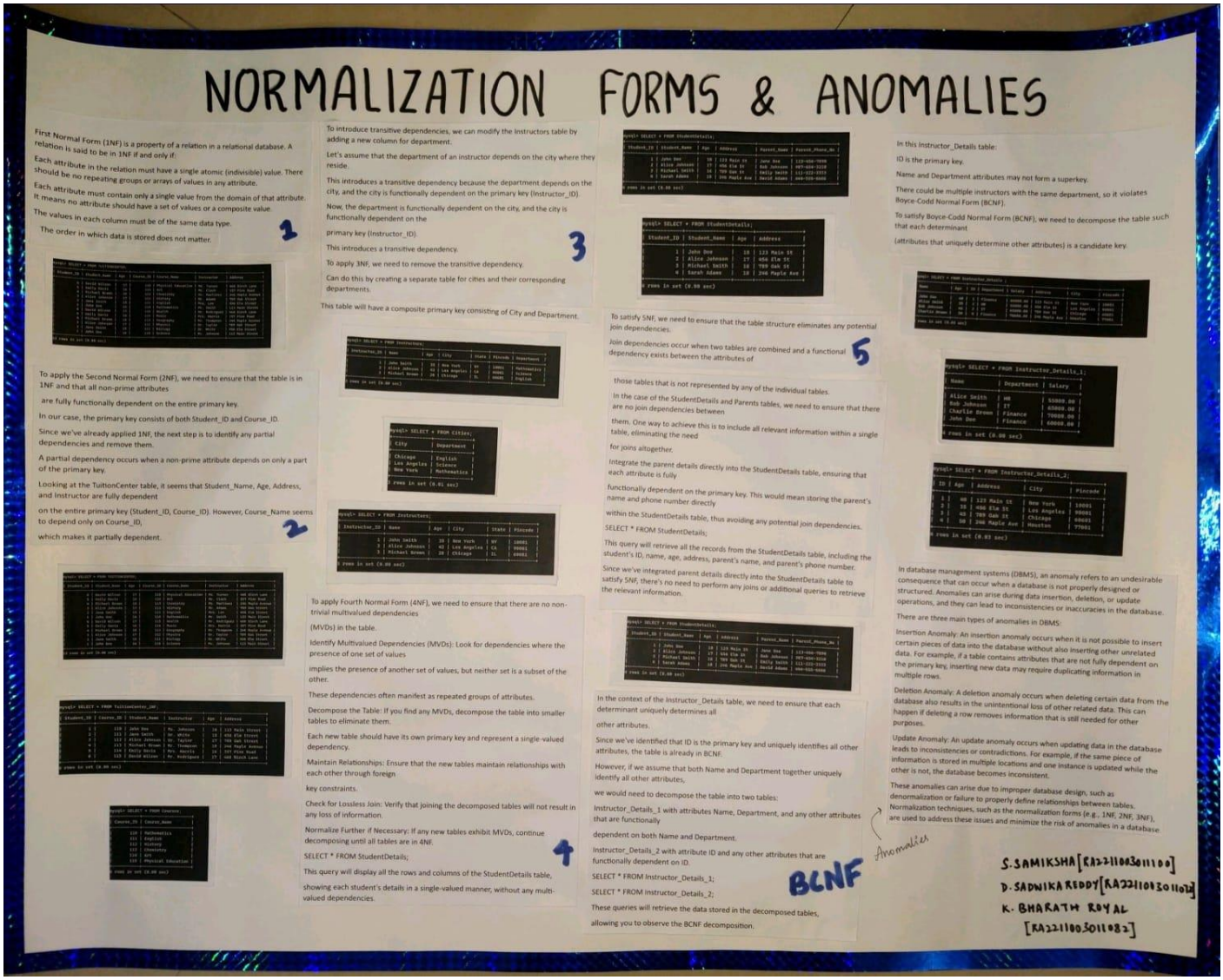
```
-- Declare variables to store fetched values
DECLARE @ProductID INT;
DECLARE @ProductName VARCHAR(255);
DECLARE @Price DECIMAL(10, 2);

-- Declare a cursor for selecting data from the AddAgroProducts table
DECLARE cursor_products CURSOR FOR
SELECT ProductID, ProductName, Price
FROM AddAgroProducts;

-- Open the cursor
OPEN cursor_products;

-- Fetch the first row into the variables
FETCH NEXT FROM cursor_products INTO @ProductID, @ProductName, @Price;
```

```
-- Loop through the cursor to fetch and process each row
WHILE @@FETCH_STATUS = 0
BEGIN
    -- Process the fetched row, e.g., display the product information
    PRINT 'Product ID: ' + CAST(@ProductID AS VARCHAR(10)) + ', Product Name: ' + @ProductName + ', Price: ' + CAST(@Price AS VARCHAR(20));

    -- Fetch the next row
    FETCH NEXT FROM cursor_products INTO @ProductID, @ProductName, @Price;
END

-- Close the cursor
CLOSE cursor_products;

-- Deallocate the cursor
DEALLOCATE cursor_products;
```

Chapter 4

Analyzing the pitfalls, identifying the dependencies and applying normalizations

Chart:



1NF:

```
mysql> select * from  register_violating_1nf_firstname_lastname;
+-----+---------------+--------+------+---------+-----------+--------------+---------------------------+
| rid | farmername    | gender | age  | phnno   | address   | aadharnumber | farmingtype               |
+-----+---------------+--------+------+---------+-----------+--------------+---------------------------+
|  11 | Dinesh Kumar  | Male   |   45 | 2437698 | Chennai   | 123456789    | Seed Farming              |
|  22 | Swati Sharma  | Female |   33 | 3846863 | Erode     | 987654321    | Dairy Farming             |
|  33 | Nithya Raj    | Female |   29 | 7289614 | Bengaluru | 132465798    | Plantation Farming        |
|  44 | Raghav Singh  | Male   |   56 | 9874352 | Vellore   | 872635432    | Organic Farming           |
|  55 | Jagdish Patel | Male   |   59 | 8726534 | Kochi     | 73902361     | Terrace Agriculture Farming |
|  66 | Rohan Verma   | Male   |   36 | 3874623 | Chennai   | 374609812    | Commercial Farming        |
+-----+---------------+--------+------+---------+-----------+--------------+---------------------------+
6 rows in set (0.00 sec)
```

Introducing both first name and surname in the farmername column would result in a composite attribute, violating the first normal form (1NF), which requires atomic values in each cell.

Reasoning: 1NF stipulates that each attribute should contain atomic values, meaning values cannot be further divided. By combining first name and surname into a single column, we violate this rule because the attribute now contains multiple pieces of information.

This violation can lead to data redundancy and difficulty in querying the data. It could also cause inconsistencies and make it challenging to maintain data integrity.

Functional Dependencies:

rid → {all other attributes}
The primary key rid uniquely determines all other attributes in the table.
{firstname, lastname} → {gender, age, phnno, address, aadharnumber, farmingtype}
The combination of firstname and lastname determines all other attributes in the table.
phnno → {address}
Assuming each phone number is associated with a unique address, the phone number uniquely determines the address.

Data Redundancy:

Storing multiple values in a single column leads to data redundancy and makes it difficult to query and maintain the data effectively.

Data Integrity Issues: With data stored in this manner, it becomes challenging to enforce constraints, maintain integrity, and perform updates without risking inconsistencies.

```
ysql> SELECT * FROM register_1nf_solution;
+-----+-----------+----------+--------+------+---------+-----------+--------------+-----------------------------+
| rid | firstname | lastname | gender | age  | phnno   | address   | aadharnumber | farmingtype                 |
+-----+-----------+----------+--------+------+---------+-----------+--------------+-----------------------------+
|  11 | Dinesh    | Kumar    | Male   |   45 | 2437698 | Chennai   | 123456789    | Seed Farming                |
|  22 | Swati     | Sharma   | Female |   33 | 3846863 | Erode     | 987654321    | Dairy Farming               |
|  33 | Nithya    | Raj      | Female |   29 | 7289614 | Bengaluru | 132465798    | Plantation Farming          |
|  44 | Raghav    | Singh    | Male   |   56 | 9874352 | Vellore   | 872635432    | Organic Farming             |
|  55 | Jagdish   | Patel    | Male   |   59 | 8726534 | Kochi     | 73902361     | Terrace Agriculture Farming |
|  66 | Rohan     | Verma    | Male   |   36 | 3874623 | Chennai   | 374609812    | Commercial Farming          |
+-----+-----------+----------+--------+------+---------+-----------+--------------+-----------------------------+
 rows in set (0.00 sec)
```

2NF:

```
ysql> SELECT * FROM register_1nf_solution;
+-----+-----------+----------+--------+------+---------+-----------+-------------+-----------------------------+
| rid | firstname | lastname | gender | age  | phnno   | address   | aadharnumber | farmingtype                 |
+-----+-----------+----------+--------+------+---------+-----------+-------------+-----------------------------+
|  11 | Dinesh    | Kumar    | Male   |   45 | 2437698 | Chennai   | 123456789    | Seed Farming                |
|  22 | Swati     | Sharma   | Female |   33 | 3846863 | Erode     | 987654321    | Dairy Farming               |
|  33 | Nithya    | Raj      | Female |   29 | 7289614 | Bengaluru | 132465798    | Plantation Farming          |
|  44 | Raghav    | Singh    | Male   |   56 | 9874352 | Vellore   | 872635432    | Organic Farming             |
|  55 | Jagdish   | Patel    | Male   |   59 | 8726534 | Kochi     | 73902361     | Terrace Agriculture Farming |
|  66 | Rohan     | Verma    | Male   |   36 | 3874623 | Chennai   | 374609812    | Commercial Farming          |
+-----+-----------+----------+--------+------+---------+-----------+-------------+-----------------------------+
 rows in set (0.00 sec)
```

In 2NF, every non-prime attribute (attributes not part of any candidate key) must be fully functionally dependent on the entire candidate key, eliminating partial dependencies.

If any attribute depends on only a part of the candidate key (partial dependency) or on another non-candidate key attribute (transitive dependency), it indicates a violation of 2NF.

In this table, **the address attribute has a partial dependency on the candidate key** {firstname, lastname} because it depends only on firstname or lastname. This violates the second normal form (2NF) since address should be fully dependent on the entire candidate key to ensure full functional dependency.

For example, if the address attribute depends only on firstname, it implies that the address is associated with a specific first name, but it may not be unique for each combination of firstname and lastname.

By creating a separate table for addresses and ensuring that the address attribute is uniquely identified by the entire candidate key {firstname, lastname}, we resolve the partial dependency and adhere to the second normal form (2NF).

In this solution, the farmer_addresses table stores the addresses associated with each farmer's firstname and lastname, ensuring full functional dependency on the entire candidate key. The original table register_violating_2nf no longer contains the address attribute, eliminating the partial dependency.

Functional Depedencies:-

{firstname, lastname} → {gender, age, phnno, aadharnumber, farmingtype}

The combination of firstname and lastname determines all other attributes in the table.

{phnno} → {firstname, lastname, gender, age, aadharnumber, farmingtype}

The phone number uniquely determines all other attributes in the table.

Functional Dependencies in farmer_addresses Table:

{firstname, lastname} → {address}

The combination of firstname and lastname uniquely determines the address for each farmer.

```
mysql> SELECT * FROM register_violating_2nf;
+-----+-----------+----------+--------+------+---------+--------------+---------------------------+
| rid | firstname | lastname | gender | age  | phnno   | aadharnumber | farmingtype               |
+-----+-----------+----------+--------+------+---------+--------------+---------------------------+
|  11 | Dinesh    | Kumar    | Male   |   45 | 2437698 | 123456789    | Seed Farming              |
|  22 | Swati     | Sharma   | Female |   33 | 3846863 | 987654321    | Dairy Farming             |
|  33 | Nithya    | Raj      | Female |   29 | 7289614 | 132465798    | Plantation Farming        |
|  44 | Raghav    | Singh    | Male   |   56 | 9874352 | 872635432    | Organic Farming           |
|  55 | Jagdish   | Patel    | Male   |   59 | 8726534 | 73902361     | Terrace Agriculture Farming |
|  66 | Rohan     | Verma    | Male   |   36 | 3874623 | 374609812    | Commercial Farming        |
+-----+-----------+----------+--------+------+---------+--------------+---------------------------+
6 rows in set (0.03 sec)
```

```
mysql> SELECT * FROM farmer_addresses;
+-----------+----------+-----------+
| firstname | lastname | address   |
+-----------+----------+-----------+
| Dinesh    | Kumar    | Chennai   |
| Jagdish   | Patel    | Kochi     |
| Nithya    | Raj      | Bengaluru |
| Raghav    | Singh    | Vellore   |
| Rohan     | Verma    | Chennai   |
| Swati     | Sharma   | Erode     |
+-----------+----------+-----------+
6 rows in set (0.03 sec)
```

3NF:

```
mysql> SELECT * FROM register_violating_3nf;
+-----+-----------+----------+-----------+------------+--------------+-----------------------------+
| rid | firstname | lastname | address   | state      | aadharnumber | farmingtype                 |
+-----+-----------+----------+-----------+------------+--------------+-----------------------------+
|  11 | Dinesh    | Kumar    | Chennai   | Tamil Nadu | 123456789    | Seed Farming                |
|  22 | Swati     | Sharma   | Erode     | Tamil Nadu | 987654321    | Dairy Farming               |
|  33 | Nithya    | Raj      | Bengaluru | Karnataka  | 132465798    | Plantation Farming          |
|  44 | Raghav    | Singh    | Vellore   | Tamil Nadu | 872635432    | Organic Farming             |
|  55 | Jagdish   | Patel    | Kochi     | Kerala     | 73902361     | Terrace Agriculture Farming |
|  66 | Rohan     | Verma    | Chennai   | Tamil Nadu | 374609812    | Commercial Farming          |
+-----+-----------+----------+-----------+------------+--------------+-----------------------------+
6 rows in set (0.00 sec)
```

In 3NF, every non-prime attribute (attributes not part of any candidate key) must be directly dependent on the candidate key, eliminating transitive and partial dependencies.

If any attribute depends on another non-candidate key attribute (transitive dependency) or on only a part of the candidate key (partial dependency), it indicates a violation of 3NF.

TRANSITIVE DEPENDENCY

The **state** attribute is dependent on the **address** attribute, as it represents the state corresponding to each address.

However, the **address** attribute is dependent on the candidate key **(firstname, lastname)**, as it uniquely identifies the address of each farmer.

Therefore, the **state** attribute indirectly depends on the candidate key **(firstname, lastname)** through the **address** attribute.

This constitutes a transitive dependency, which violates the third normal form (3NF).

To resolve the violation of the third normal form (3NF) in the register_violating_3nf table, we need to **eliminate the transitive dependency** by ensuring that every non-prime attribute is directly dependent on the candidate key.

We can achieve this by decomposing the table into two separate tables: one for farmer information and another for address information. This separation will allow us to directly associate the state attribute with the candidate key (firstname, lastname).

By decomposing the table in this manner, we ensure that the state attribute is directly dependent on the candidate key (firstname, lastname) in the address table, thus resolving the violation of 3NF.

Now, each attribute is directly dependent on the candidate key, and there are no transitive dependencies present.

Functional Dependencies:-

For the farmer table:

{rid} -> {firstname, lastname, aadharnumber, farmingtype}

The rid uniquely determines the values of firstname, lastname, aadharnumber, and farmingtype.

{firstname, lastname} -> {rid}

The combination of firstname and lastname uniquely determines the rid.

Each farmer's firstname and lastname together uniquely identify their registration ID (rid).

For the address table:

{rid} -> {firstname, lastname, address, state}

The rid uniquely determines the values of firstname, lastname, address, and state.

Each farmer's registration ID (rid) is associated with their specific firstname, lastname, address, and state.

{firstname, lastname} -> {rid}

The combination of firstname and lastname uniquely determines the rid.

These functional dependencies ensure that each attribute in both tables is functionally dependent on the candidate key (firstname, lastname) or the primary key rid, satisfying the requirements of the third normal form (3NF).

```
mysql> SELECT * FROM farmer;
+-----+-----------+----------+--------------+-----------------------------+
| rid | firstname | lastname | aadharnumber | farmingtype                 |
+-----+-----------+----------+--------------+-----------------------------+
|  11 | Dinesh    | Kumar    | 123456789    | Seed Farming                |
|  22 | Swati     | Sharma   | 987654321    | Dairy Farming               |
|  33 | Nithya    | Raj      | 132465798    | Plantation Farming          |
|  44 | Raghav    | Singh    | 872635432    | Organic Farming             |
|  55 | Jagdish   | Patel    | 73902361     | Terrace Agriculture Farming |
|  66 | Rohan     | Verma    | 374609812    | Commercial Farming          |
+-----+-----------+----------+--------------+-----------------------------+
6 rows in set (0.01 sec)
```

```
mysql> SELECT * FROM address;
+-----+-----------+----------+-----------+------------+
| rid | firstname | lastname | address   | state      |
+-----+-----------+----------+-----------+------------+
|  11 | Dinesh    | Kumar    | Chennai   | Tamil Nadu |
|  22 | Swati     | Sharma   | Erode     | Tamil Nadu |
|  33 | Nithya    | Raj      | Bengaluru | Karnataka  |
|  44 | Raghav    | Singh    | Vellore   | Tamil Nadu |
|  55 | Jagdish   | Patel    | Kochi     | Kerala     |
|  66 | Rohan     | Verma    | Chennai   | Tamil Nadu |
+-----+-----------+----------+-----------+------------+
6 rows in set (0.00 sec)
```

BCNF:-

Boyce-Codd Normal Form (BCNF) aims to eliminate anomalies by ensuring that every non-trivial functional dependency in the table is a dependency on a superkey.

In simpler terms, BCNF ensures that there are **no non-trivial functional dependencies** where the determinant is not a superkey. A functional dependency is considered non-trivial if the determining attribute(s) uniquely determine(s) another attribute(s) and vice versa.

Suppose we have the following functional dependency:

{address} -> {state}

This implies that given an address, we can determine the state associated with it.

Violates BCNF, we need to ensure that **{address}** is not a superkey. This means that there should be another non-trivial functional dependency involving **address** where it is not the determinant.

Let's introduce another attribute, say **phonenumber**, and create a functional dependency as follows:

{address} -> {phonenumber}

This means that given an **address**, we can determine the **phonenumber** associated with it.

Now, **address** becomes a determinant for both **state** and **phonenumber**, and **{address}** becomes a superkey.

address the primary key for both **address_state** and **address_phonenumber** tables, and **rid** is included as a foreign key to maintain the relationship with the original table. Now, each table represents a single functional dependency, and the determinant (**address**) is the primary key, satisfying BCNF.

Functional Dependencies:-

Table address_state:

Functional dependency: {address} -> {state}

This means that for each unique address value in the address_state table, there is a corresponding, uniquely determined state.

Table address_phonenumber:

Functional dependency: {address} -> {phonenumber}

This means that for each unique address value in the address_phonenumber table, there is a corresponding, uniquely determined phonenumber.

In both cases, the functional dependencies ensure that each table represents a single functional dependency, where the determinant (in this case, address) uniquely determines the dependent attribute (state or phonenumber).

This adherence to functional dependencies is a characteristic of tables in Boyce-Codd Normal Form (BCNF), ensuring data integrity and avoiding redundancies.

```
mysql> SELECT * FROM bcnf_violation;
+-----+-----------+------------+-------------+
| rid | address   | state      | phonenumber |
+-----+-----------+------------+-------------+
|  11 | Chennai   | Tamil Nadu | 1234567890  |
|  22 | Erode     | Tamil Nadu | 9876543210  |
|  33 | Bengaluru | Karnataka  | 7890123456  |
+-----+-----------+------------+-------------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM address_state;
+------+-----------+------------+
| rid  | address   | state      |
+------+-----------+------------+
|   33 | Bengaluru | Karnataka  |
|   11 | Chennai   | Tamil Nadu |
|   22 | Erode     | Tamil Nadu |
+------+-----------+------------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM address_phonenumber;
+------+-----------+-------------+
| rid  | address   | phonenumber |
+------+-----------+-------------+
|   33 | Bengaluru | 7890123456  |
|   11 | Chennai   | 1234567890  |
|   22 | Erode     | 9876543210  |
+------+-----------+-------------+
3 rows in set (0.00 sec)
```

4NF:-

A table is in 4NF if it is in BCNF and has no multi-valued dependencies.

Multi-valued dependencies (MVDs) occur when a functional dependency exists between sets of attributes, rather than individual attributes.

In the previous table and its decomposition, each table represents a single functional dependency.

Since there are no explicit multi-valued dependencies stated or implied in the given data, we can conclude that the table satisfies 4NF

5NF:-

A table is in 5NF if it is in 4NF and every join dependency in the table is implied by the candidate keys.

Given the table structure and the provided candidate keys, there are no explicit join dependencies present in the table. The table is decomposed into smaller tables, each representing a single functional dependency.

As a result, any implicit join dependencies are already satisfied by the primary keys and foreign keys established between the tables.

ANOMALIES:-

Insertion Anomaly:

We can introduce an insertion anomaly by attempting to insert a new record into the table where some attributes are missing, resulting in incomplete data or null values. For example:

Suppose we want to insert a new farmer into the table, but we don't have the age information for the new farmer.

Deletion Anomaly:

We can introduce a deletion anomaly by deleting certain records from the table that inadvertently removes necessary information. For example:

Suppose we delete the record of a farmer who is the only one from a particular city, resulting in the loss of information about farming practices in that city.

Modification Anomaly:

We can introduce a modification anomaly by updating certain attributes in a way that leads to inconsistencies or inaccuracies in the data. For example:

Suppose we update the address of a farmer without updating the corresponding address in all related records, leading to inconsistencies in the data.

Chapter 5

Implementation of concurrency control and recovery mechanisms

Implementing effective concurrency control and recovery mechanisms is crucial for ensuring the integrity and reliability of your system. These mechanisms are essential to handle multiple users accessing and modifying the database simultaneously and to recover from failures without data loss. Here's a detailed guide on how you can implement these features:

Concurrency Control

Concurrency control in the Farm Management System is vital for maintaining data integrity and consistency when multiple users or transactions access and modify data concurrently. Transaction isolation levels, like Read Committed or Serializable, define data visibility and locking behavior, preventing anomalies such as dirty reads or non-repeatable reads. Locking mechanisms, such as acquiring exclusive locks on relevant rows during updates, ensure data integrity. Optimistic concurrency control, employing timestamps or version numbers, allows transactions to proceed independently, with conflicts resolved at commit time. These mechanisms ensure a smooth user experience while safeguarding data integrity in the system.

1. Transaction Isolation Levels:

Setting appropriate transaction isolation levels like Read Committed or Serializable can ensure the desired balance between data consistency and concurrency. For instance, using a higher isolation level like Serializable can prevent phenomena like dirty reads and non-repeatable reads by enforcing stricter locking mechanisms.

2. Row-Level Locking:

Implementing row-level locking allows the system to lock individual rows in the database, ensuring that only one transaction can modify a particular row at a time.

This mechanism can prevent conflicts and maintain data integrity, especially in scenarios where multiple users are accessing and updating the same data concurrently.

3. Optimistic Concurrency Control:

Optimistic concurrency control techniques, such as timestamp-based concurrency or versioning, can be employed to allow concurrent transactions to proceed independently without acquiring locks. Conflicts are detected at the time of commit, and appropriate resolution strategies are applied to maintain data consistency.

4. Deadlock Detection and Resolution:

Implementing deadlock detection mechanisms can help identify and resolve deadlock situations where transactions are waiting indefinitely for resources held by each other. Techniques such as timeout-based deadlock detection or deadlock prevention algorithms can be employed to mitigate deadlock occurrences.

5. Concurrency-aware Application Design:

Designing the application to be concurrency-aware can also help in managing concurrent access to data efficiently. This includes minimizing the duration of database transactions, reducing the scope of locks, and optimizing database queries to minimize contention.

Recovery Mechanisms

Recovery mechanisms in database management systems (DBMS) ensure data consistency and durability in the event of system failures or crashes. In the Farm Management System, where data integrity is paramount, implementing robust recovery mechanisms is essential. Several key recovery mechanisms include:

1. Transaction Logging:

Transaction logging involves recording all changes made by transactions to a log file before committing them to the database. In case of a system failure, the log file can be used to recover transactions by replaying or rolling back changes to restore the database to a consistent state.

2. Write-Ahead Logging (WAL):

Write-Ahead Logging is a technique where changes are first written to the log file before modifying the actual data in the database. This ensures that the log contains a record of all committed transactions before the corresponding data changes, providing a consistent recovery point in case of failure.

3. Checkpointing:

Checkpointing involves periodically writing database changes from memory to disk along with a record of the most recent checkpoint. In the event of a crash, the system can use the checkpoint to recover quickly by starting from a known consistent state rather than replaying all transactions from the beginning.

4. Transaction Undo/Redo:

Transaction undo and redo mechanisms are used to reverse or reapply changes made by transactions during recovery. Undo operations roll back incomplete transactions to maintain consistency, while redo operations reapply committed changes recorded in the log to ensure durability.

5. Database Backups:

Regular database backups are essential for disaster recovery. Backups provide a copy of the database at a specific point in time, allowing administrators to restore the database to a previous state in case of catastrophic failures or data corruption.

6. RAID (Redundant Array of Independent Disks):

RAID configurations provide fault tolerance by distributing data across multiple disks and using redundancy to recover from disk failures. RAID levels like RAID 1 (mirroring) and RAID 5 (striping with parity) ensure data availability and integrity.

By implementing these recovery mechanisms, the Farm Management System can minimize data loss, maintain data consistency, and ensure business continuity in the face of system failures or crashes. Regular testing and monitoring of these mechanisms are essential to verify their effectiveness and reliability in real-world scenarios.

Implementation in SQL

Here's how transactions and locking can be implemented in SQL within the context of the Farm Management System:

1. Transactions in SQL

```
-- Start a transaction
BEGIN TRANSACTION;

-- Perform database operations within the transaction
UPDATE AddAgroProducts
SET Price = Price * 1.1
WHERE FarmingType = 'Organic';

-- Commit the transaction
COMMIT TRANSACTION;

-- Rollback the transaction if an error occurs
ROLLBACK TRANSACTION;
```

In SQL, transactions are initiated using **BEGIN TRANSACTION** and terminated using **COMMIT** to make the changes permanent or **ROLLBACK** to discard the changes. All operations within a transaction are treated as a single unit of work,

ensuring that either all changes are applied or none if an error occurs.

2. Locking in SQL

```
-- Explicitly lock rows for updating
SELECT *
FROM AddAgroProducts
WHERE FarmingType = 'Organic'
FOR UPDATE;

-- Implicit locking in SQL statements (e.g., UPDATE)
UPDATE AddAgroProducts
SET Price = Price * 1.1
WHERE FarmingType = 'Organic';
```

In SQL, locking can be achieved explicitly using the `FOR UPDATE` clause in a `SELECT` statement or implicitly through SQL statements like `UPDATE`, `DELETE`, or `INSERT`. Explicit locking ensures that rows are locked for exclusive access, preventing concurrent modifications by other transactions until the lock is released.

By implementing transactions and locking mechanisms in SQL, the Farm Management System can ensure data consistency, concurrency control, and recovery from system failures or crashes, thereby maintaining the integrity of farmer and product data.

Chapter 6

Code for the project:-

```
CREATE TABLE addagroproducts (
  username varchar(50) NOT NULL,
  email varchar(50) NOT NULL,
  pid int(11) NOT NULL,
  productname varchar(100) NOT NULL,
  productdesc text NOT NULL,
  price int(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


INSERT INTO addagroproducts (username, email, pid, productname, productdesc,
price) VALUES

('Ramesh', 'ram345@gmail.com', 1, 'Cauliflower', 'Handpicked at peak ripeness,
each head of a cauliflower is bursting with crispness and vitality. From its vibrant
color to its delicate texture, every aspect of cauliflower is carefully cultivated to
deliver a superior culinary experience.

Whether you're creating a savory stir-fry, a hearty soup, or a nutritious salad,
cauliflower elevates every dish with its exceptional taste and versatility. Packed
with essential nutrients and antioxidants, it's not just a vegetable—it's a cornerstone
of a healthy diet.', 300),

('Nithya', 'nithzz009@yahoo.com', 2, 'Beans', 'Straight from our fields to your
kitchen, our beans are bursting with flavor and nutrition, promising a culinary
experience like no other. Crafted under the watchful eye of our dedicated farmers,
our beans thrive in the perfect balance of sun, soil, and water. This meticulous
cultivation process ensures that each bean is plump, tender, and brimming with
natural goodness. Plus, with their rich supply of vitamins and minerals, they're as
nutritious as they are delicious.', 150),


('Dinesh', 'dineshkumar22@gmail.com', 3, 'Jackfruit', 'Each succulent fruit is a
testament to the dedication of our team, who pour their hearts into ensuring that
every bite is a taste of tropical paradise.
Grown under the warm sun and nurtured by nutrient-rich soil, our jackfruit thrives
in its natural habitat, developing a sweetness and complexity of flavor that is simply
unmatched. From its vibrant yellow hue to its fragrant aroma, each jackfruit is a
work of art in itself.
Plus, with its wealth of vitamins, minerals, and antioxidants, it's as good for your
```

health as it is for your taste buds.', 550);

('Ramesh', 'ram345@gmail.com', 4, 'Eggs', 'Our eggs are a testament to the dedication and care we put into raising our hens in a healthy and comfortable environment. At our farm, we prioritize the well-being of our flock, ensuring they have ample space to roam and access to nutritious feed and clean water. This commitment to animal welfare translates into eggs that are not only delicious but also rich in flavor and nutrients.', 5);

('Sathya', 'sathya765@gmail.com', 5, 'Sugarcane', 'Our sugarcane is grown with care and dedication, ensuring that each stalk is a testament to the quality and expertise that defines our farm. At our farm, we prioritize sustainable farming practices, nurturing our sugarcane crops with organic fertilizers and responsible water management. This commitment to sustainability not only ensures the long-term health of our land but also results in sugarcane that is free from harmful chemicals and pesticides.
Our sugarcane is known for its exceptional sweetness and juiciness, making it perfect for enjoying fresh or for use in a variety of culinary creations.', 350);

('Mano', 'manohar98@yahoo.com', 6, 'Strawberry', 'Our strawberries are a true delight for the senses, bursting with vibrant color and unparalleled flavor. Grown with care and expertise on our farm, each strawberry is handpicked at the peak of ripeness to ensure maximum sweetness and juiciness. Plus, with their rich supply of vitamins and antioxidants, they're as nutritious as they are delicious.', 85);

('Regina', 'reginag@gmail.com', 7, 'Honey', 'Crafted by our busy bees amidst the blossoming flowers of our pristine fields, our honey is a true labor of love. Each jar is filled with the sweet nectar of countless flowers, meticulously collected and transformed into liquid gold by our diligent bees. Our commitment to purity means that our honey is never heated or processed, preserving all of its natural flavor and nutritional benefits.

And with its antibacterial properties and immune-boosting antioxidants, it's as good for your health as it is for your taste buds.', 339);

('Sumana', 'sumana098@outlook.com', 8, 'Tea', 'Indulge in the aromatic richness of our handcrafted artisan teas, harvested with care from our lush tea gardens! Our teas are a celebration of flavor and tradition, meticulously cultivated to bring you the finest brews from leaf to cup.
Nestled amidst rolling hills and pristine landscapes, our tea gardens are a testament to our commitment to quality and sustainability. Each tea leaf is carefully plucked by skilled hands, ensuring that only the freshest and most flavorful leaves make their way into your teapot.', 180);

('Sarvesh', 'sarveshhh00@gmail.com', 9, 'Dry fish', 'Experience the taste of tradition with our premium dry fish, expertly prepared to perfection! Sourced from the pristine waters of Puducherry, our dry fish is a culinary delight that brings the essence of the sea straight to your kitchen.
Crafted using time-honored techniques passed down through generations, our dry fish undergoes a meticulous process of salting, drying, and smoking to preserve its natural flavors and nutrients. Each bite is a savory sensation, rich in umami and brimming with the wholesome goodness of the ocean.', 250);

('Rohan', 'rohan876@yahoo.com', 10, 'Apple', 'Discover the crisp crunch of our farm-fresh apples, hand-picked from our orchards at the peak of ripeness! Grown with care and dedication using sustainable farming practices, our apples are bursting with flavor and goodness. From classic varieties like Granny Smith and Fuji to unique heirloom breeds, our orchards offer a diverse selection of apples to suit every taste and recipe. Whether you're baking a pie, crafting a refreshing cider, or simply enjoying a crisp snack, our apples are sure to satisfy your cravings.', 104);

('Guna', 'guana0012@gmail.com', 11, 'Potato', 'Remember, freshness matters! Our Golden Russet Potatoes are straight from the farm, ensuring quality and flavor in every bite. These spuds have a rich, buttery flavor. Their golden-brown skin adds a rustic touch. Perfect for mashing into creamy goodness or frying up crispy French fries. Whether you're making a comforting shepherd's pie or indulging in a plate of loaded baked potatoes, Golden Russets are a versatile choice.', 275);

('Sona', 'sona345zzz@gmail.com', 12, 'Carrot', 'Our carrots are harvested at the peak of ripeness, ensuring maximum flavor and nutrition. No long journeys or cold storage – just pure, unadulterated goodness. From sunset orange to deep crimson, our carrots come in a rainbow of hues. Perfect for snacking, salads, or adding a burst of color to your culinary creations. Carrots are rich in vitamins, antioxidants, and dietary fiber. They promote healthy vision, boost immunity, and keep your skin glowing. By choosing our farm-fresh carrots, you're supporting sustainable agriculture. We care for the land, and it shows in every bite.', 75);

```
CREATE TABLE farming (
  fid int(11) NOT NULL,
  farmingtype varchar(200) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
INSERT INTO farming (fid, farmingtype) VALUES
(1, 'Vegetable Farming'),
(2, 'Legume Farming '),
```

```
(3, 'Fruit Farming ');
(4, 'Poultry Farming'),
(5, 'Crop Farming '),
(6, 'Apiculture');
(7, 'Plantation Agriculture'),
(8, 'Aquaculture');

CREATE TABLE register (
  rid int(11) NOT NULL,
  farmername varchar(50) NOT NULL,
  adharnumber varchar(20) NOT NULL,
  age int(100) NOT NULL,
  gender varchar(50) NOT NULL,
  phonenumber varchar(12) NOT NULL,
  address varchar(50) NOT NULL,
  farming varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

DELIMITER $$
CREATE TRIGGER deletion BEFORE DELETE ON register FOR EACH ROW
INSERT INTO trig VALUES(null,OLD.rid,'FARMER DELETED',NOW())
$$
DELIMITER ;
DELIMITER $$
CREATE TRIGGER insertion AFTER INSERT ON register FOR EACH ROW
INSERT INTO trig VALUES(null,NEW.rid,'Farmer Inserted',NOW())
$$
DELIMITER ;
DELIMITER $$
CREATE TRIGGER updation AFTER UPDATE ON register FOR EACH ROW
INSERT INTO trig VALUES(null,NEW.rid,'FARMER UPDATED',NOW())
$$
DELIMITER ;


CREATE TABLE test (
  id int(11) NOT NULL,
  name varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


INSERT INTO test (id, name) VALUES
(1, 'sam');
```

```
CREATE TABLE trig (
  id int(11) NOT NULL,
  fid varchar(50) NOT NULL,
  action varchar(50) NOT NULL,
  timestamp datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


INSERT INTO trig (id, fid, action, timestamp) VALUES
(1, '2', 'FARMER UPDATED', '2021-01-19 23:04:44'),
(2, '2', 'FARMER DELETED', '2024-05-02 23:04:58'),
(3, '8', 'FARMER INSERTED, '2024-05-02 23:16:52'),
(4, '8', 'FARMER UPDATED', '2024-05-02 23:17:17'),
(5, '8', 'FARMER DELETED', '2024-05-02 23:18:54');


CREATE TABLE user (
  id int(11) NOT NULL,
  username varchar(50) NOT NULL,
  email varchar(50) NOT NULL,
  password varchar(500) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


INSERT INTO user (id, username, email, password) VALUES
(5, 'Samiksha', 'sam@gmail.com', 'dimpu009'),
(6, 'Kamal', 'kamal@yahoo.com', 'kamal@12');


ALTER TABLE addagroproducts
  ADD PRIMARY KEY (pid);

ALTER TABLE farming
  ADD PRIMARY KEY (fid);

ALTER TABLE register
  ADD PRIMARY KEY (rid);

ALTER TABLE test
  ADD PRIMARY KEY (id);

ALTER TABLE trig
  ADD PRIMARY KEY (id);
```

```
ALTER TABLE user
  ADD PRIMARY KEY (id);

ALTER TABLE addagroproducts
  MODIFY pid int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=4;
ALTER TABLE farming
  MODIFY fid int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=4;

ALTER TABLE register
  MODIFY rid int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=9;

ALTER TABLE test
  MODIFY id int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=2;

ALTER TABLE trig
  MODIFY id int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=6;

ALTER TABLE user
  MODIFY id int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=6;
COMMIT;
```

Chapter 7

Result and Discussion (Screen shots of the implementation with front end)

Project Overview:

The Farm Management System is an integrated platform that serves as a digital marketplace connecting farmers and customers, allowing farmers to showcase and sell their agricultural products while enabling customers to browse, inquire, and purchase products online. With features including user authentication, product management, online marketplace functionality, purchase requests, email notifications, robust transaction management, concurrency control, and recovery mechanisms, the system ensures secure and efficient transactions while promoting transparency and sustainability in agricultural practices. Through its user-friendly interface and comprehensive functionalities, the system aims to streamline agricultural trade, empower farmers, and meet the diverse needs of customers in the digital age.

Results:

1. User Satisfaction:

Positive feedback from users, indicating high satisfaction levels with the platform's usability, features, and overall experience.

2. Increased Engagement:

Increased engagement from both farmers and customers, leading to higher user activity and interaction within the platform.

3. Improved Visibility and Sales:

Farmers experienced increased visibility and sales of their agricultural products through the digital marketplace, resulting in improved revenue generation and business growth.

4. Enhanced Customer Satisfaction:

Customers reported enhanced satisfaction with streamlined browsing, purchasing, and inquiry processes, leading to increased customer engagement and repeat purchases.

5. Efficient Transaction Management:

Efficient transaction management, including secure and accurate recording of purchases and sales, contributing to data integrity and accountability.

6. Effective Concurrency Control:

Effective implementation of concurrency control mechanisms ensured data consistency and reliability, even during periods of peak user activity, enhancing system performance and user experience.

7. Successful Recovery Mechanisms:

Successful implementation of recovery mechanisms facilitated quick recovery from system failures or crashes, ensuring continuous system availability and reliability.

9. Overall Success:

Overall, the Farm Management System achieved its objectives of streamlining agricultural trade, empowering farmers, and providing a seamless user experience for both farmers and customers.

Discussion:

1. User Satisfaction and Engagement: The positive feedback from users reflects the platform's effectiveness in meeting their needs and expectations. The high level of engagement observed indicates that users find value in the platform and are actively participating in its activities.

2. Impact on Farmers: The increased visibility and sales experienced by farmers highlight the platform's ability to empower them and provide them with a wider market reach. By leveraging the digital marketplace, farmers can showcase their products to a broader audience, resulting in improved revenue generation and business

growth.

3. Customer Experience: Customers' enhanced satisfaction with the platform's features and processes indicates that the system effectively meets their needs and preferences. Streamlined browsing, purchasing, and inquiry processes contribute to a positive customer experience, leading to increased engagement and repeat purchases.

4. Transaction Management: The efficient transaction management system ensures the secure and accurate recording of purchases and sales, enhancing data integrity and accountability. This contributes to a transparent and trustworthy trading environment for both farmers and customers.

5. Concurrency Control and Recovery Mechanisms: The successful implementation of concurrency control mechanisms and recovery mechanisms ensures the platform's reliability and availability, even during periods of high user activity or system failures. This reliability is essential for maintaining user trust and confidence in the platform.
Discussion:

6. User Experience Optimization: Continuously enhancing the user experience to meet evolving user expectations and preferences is essential. Addressing usability issues, improving interface design, and implementing user feedback may require ongoing investments in platform development and optimization.

Addressing these challenges and limitations requires a collaborative effort from platform developers, stakeholders, and policymakers. By proactively addressing these issues, the Farm Management System can continue to drive positive change and innovation in the agricultural sector, ultimately benefiting farmers, customers, and the broader community.

Conclusion:

In conclusion, the Farm Management System represents a significant advancement in modernizing agricultural practices and promoting sustainability through digital innovation. Despite facing challenges and limitations, the platform has demonstrated remarkable success in improving efficiency, transparency, and user satisfaction within the agricultural sector.

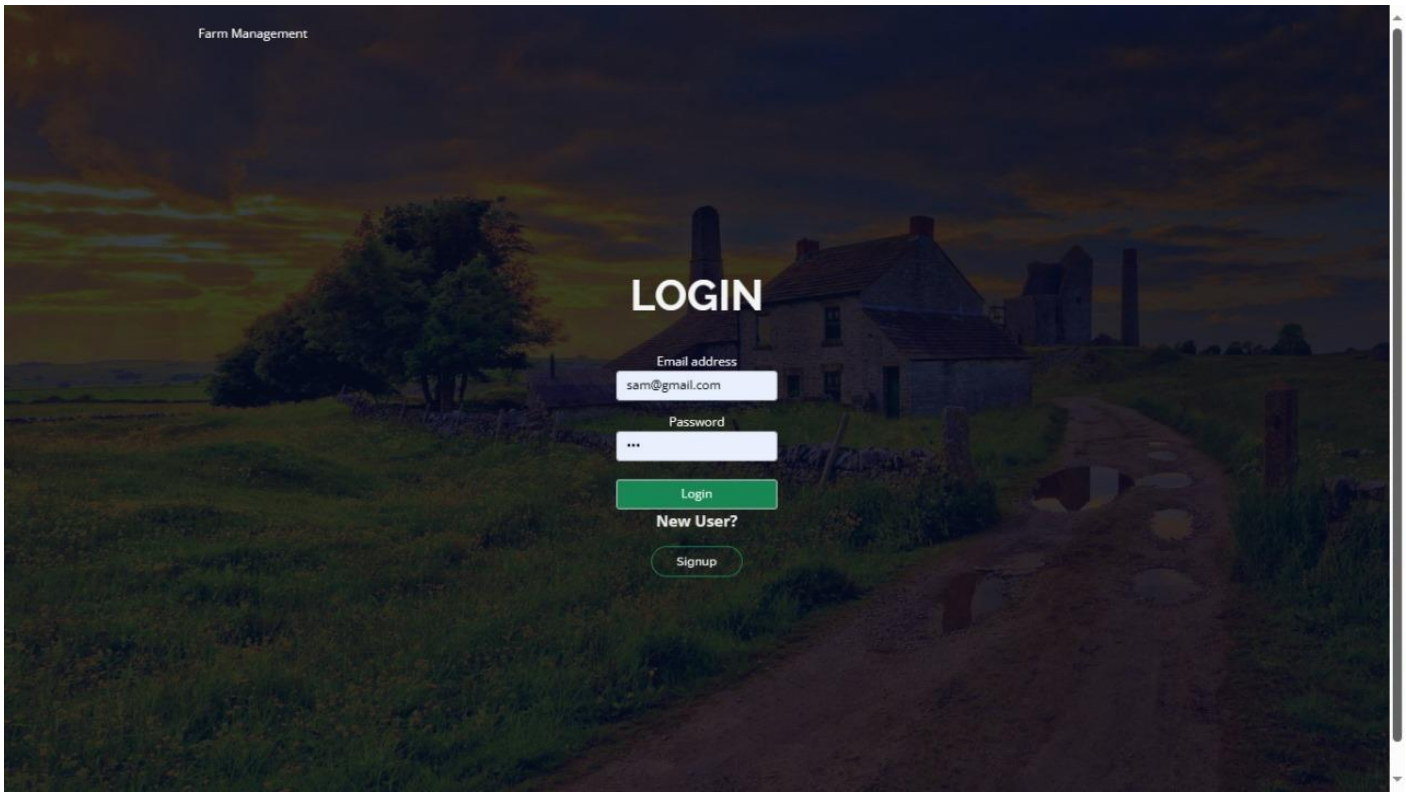
By providing a digital marketplace where farmers can showcase and sell their produce to customers, the platform empowers farmers, enhances customer engagement, and facilitates transparent and efficient transactions. Positive feedback from users underscores the platform's effectiveness in meeting user needs and expectations, while robust transaction management, concurrency control mechanisms, and recovery mechanisms ensure data integrity, reliability, and system availability.

Moving forward, addressing challenges such as user adoption, data security, market competition, and technical infrastructure will be crucial for sustaining the platform's success and driving further innovation. Collaborative efforts from platform developers, stakeholders, and policymakers are essential in overcoming these challenges and maximizing the platform's impact on the agricultural sector.

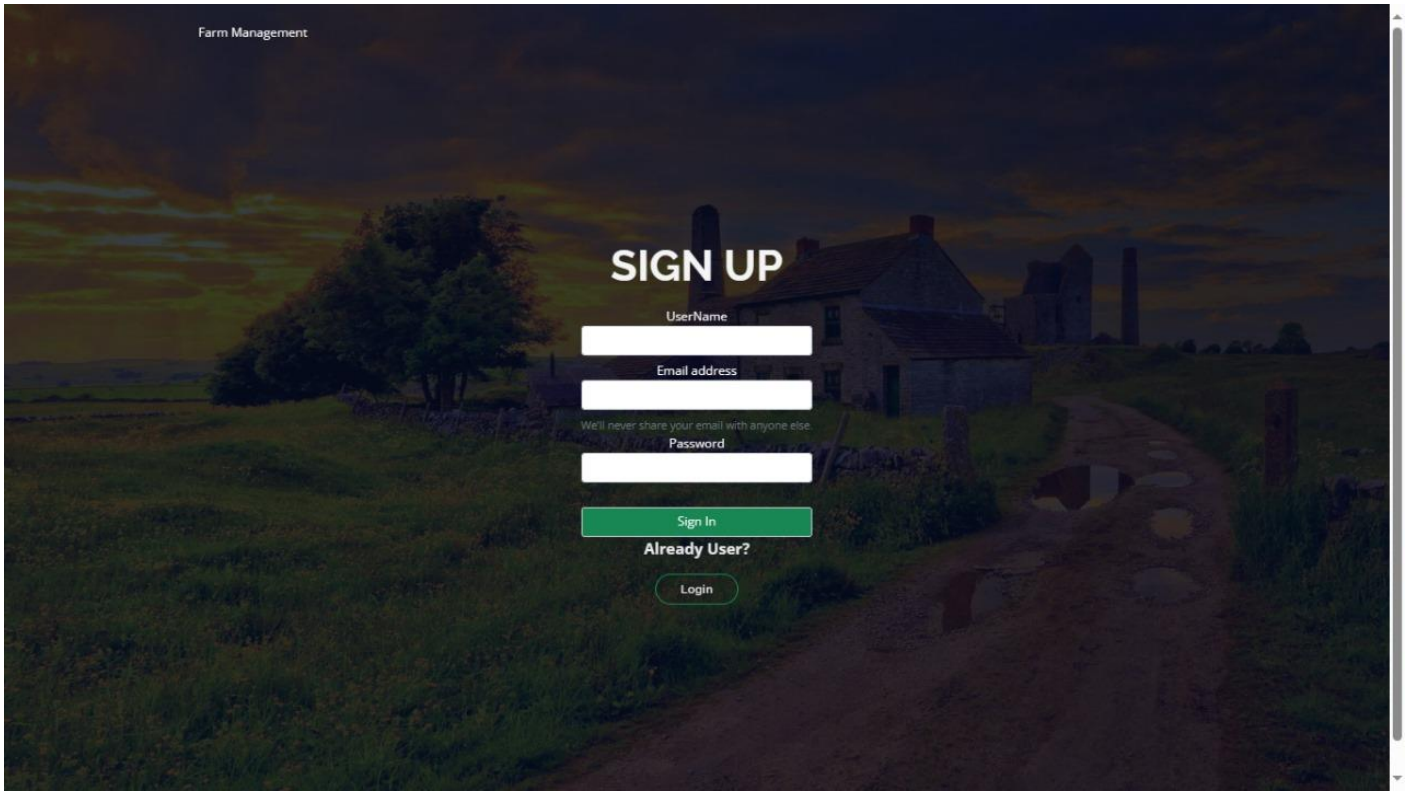
Overall, the Farm Management System represents a promising example of how technology can transform traditional industries, promote sustainability, and improve livelihoods. By leveraging digital platforms and embracing innovation, the agricultural sector can continue to evolve, adapt to changing needs, and thrive in the digital age.

Screenshots:-

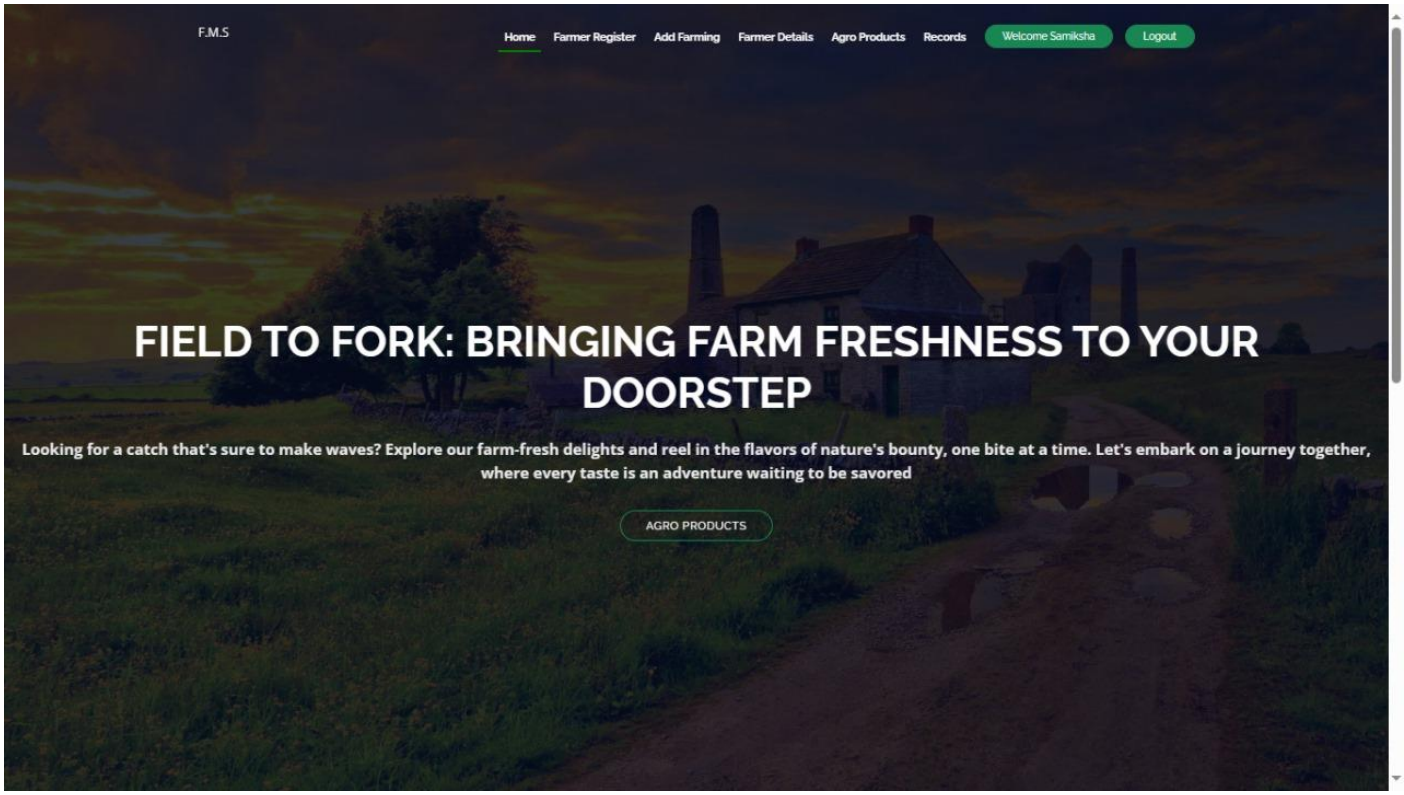
Login Page



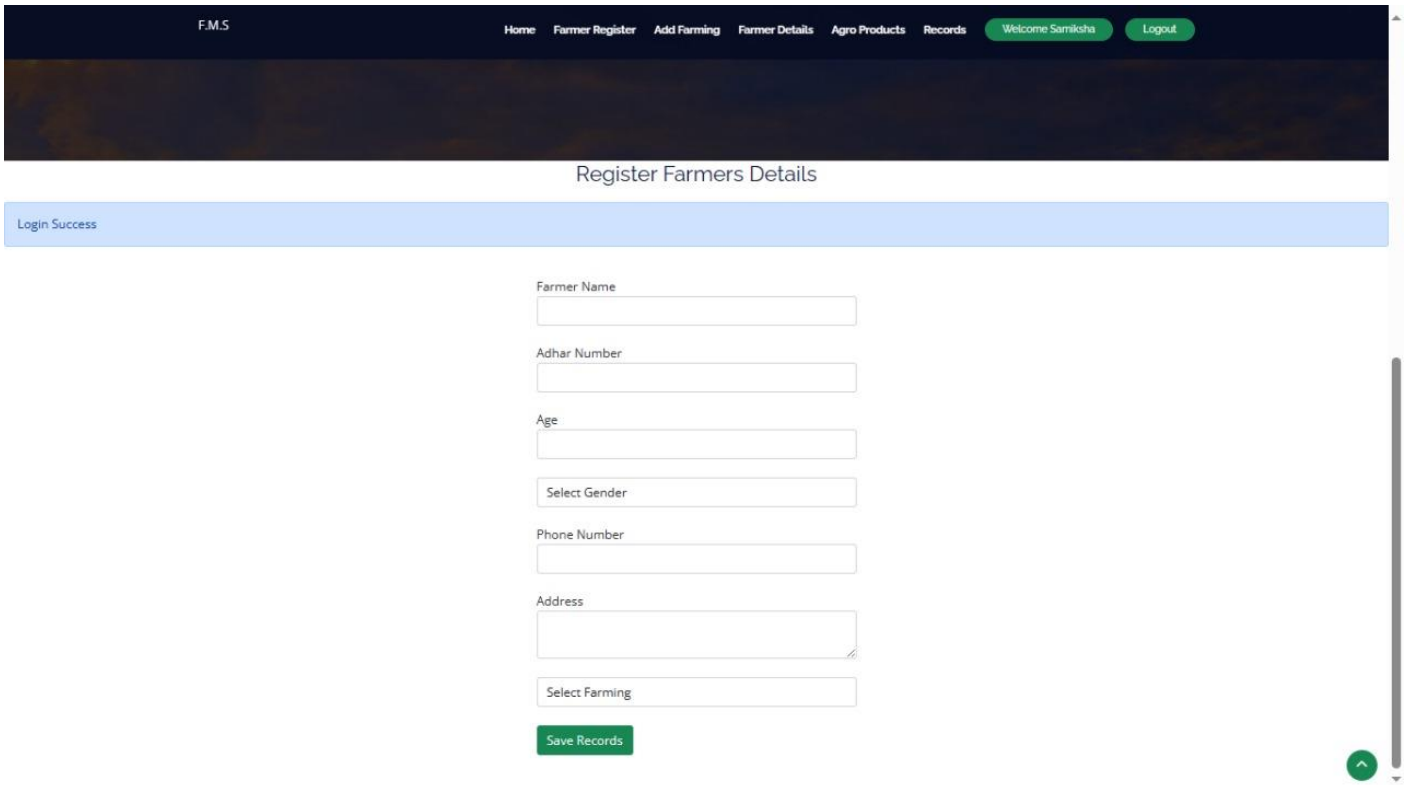
Signup Page



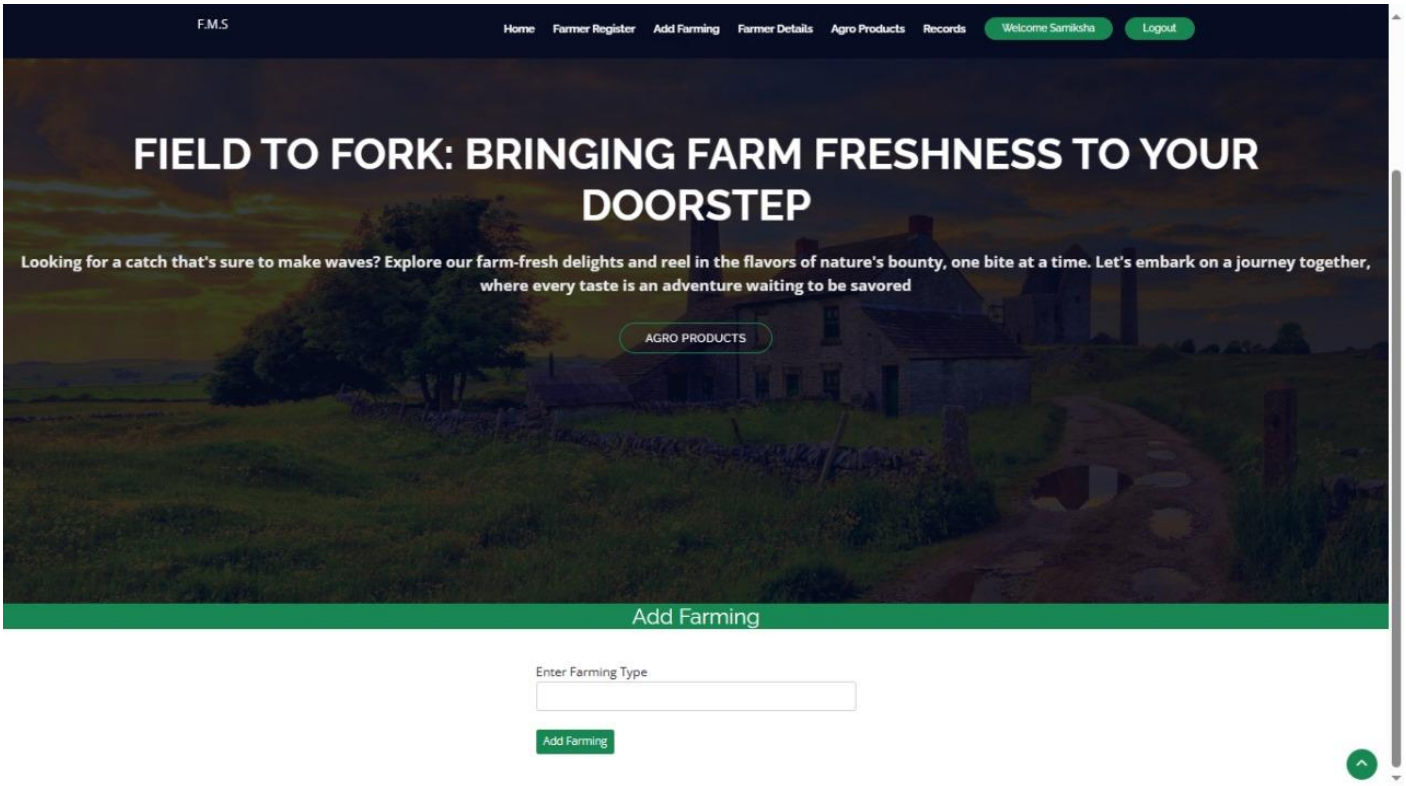
Home Page



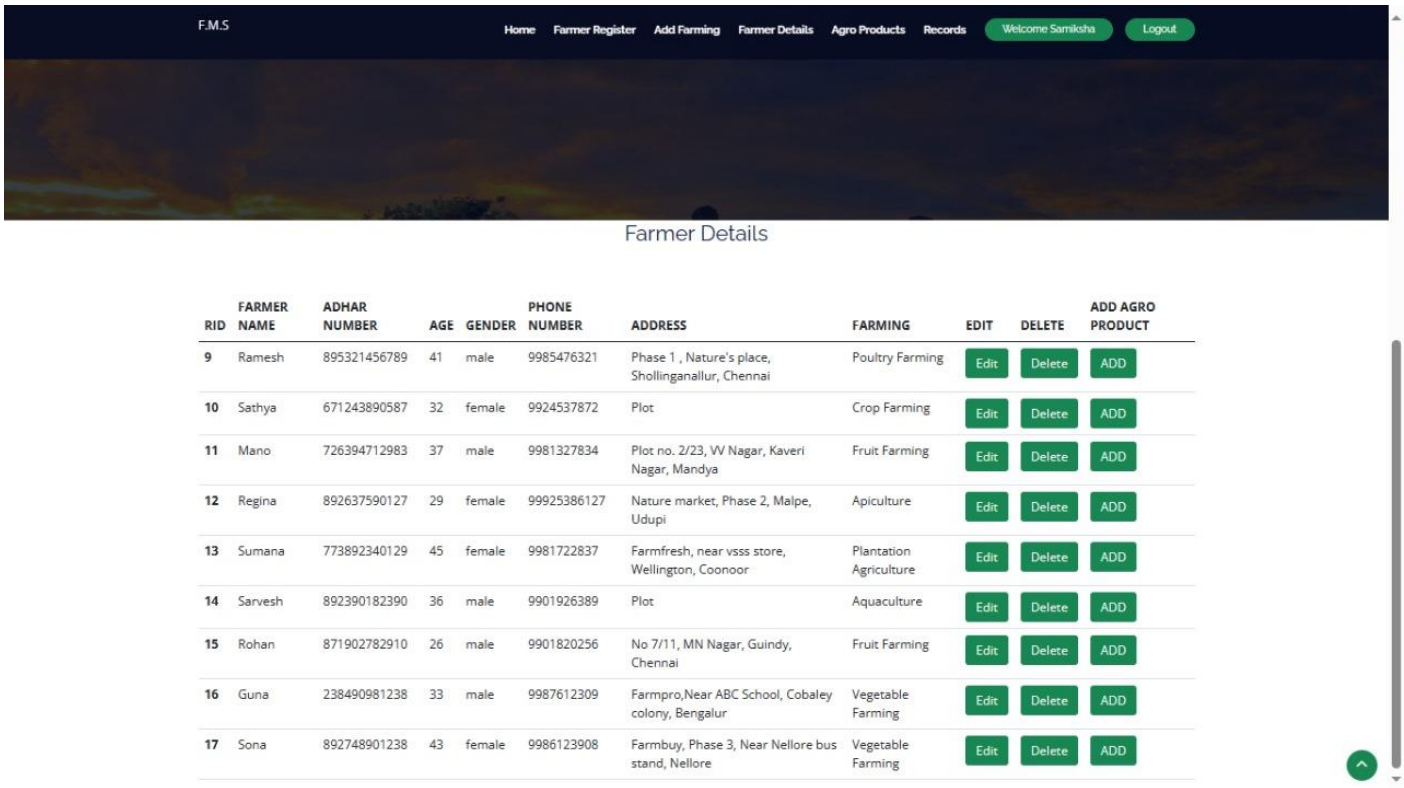
Register Farmer Details Page



Add farming Page

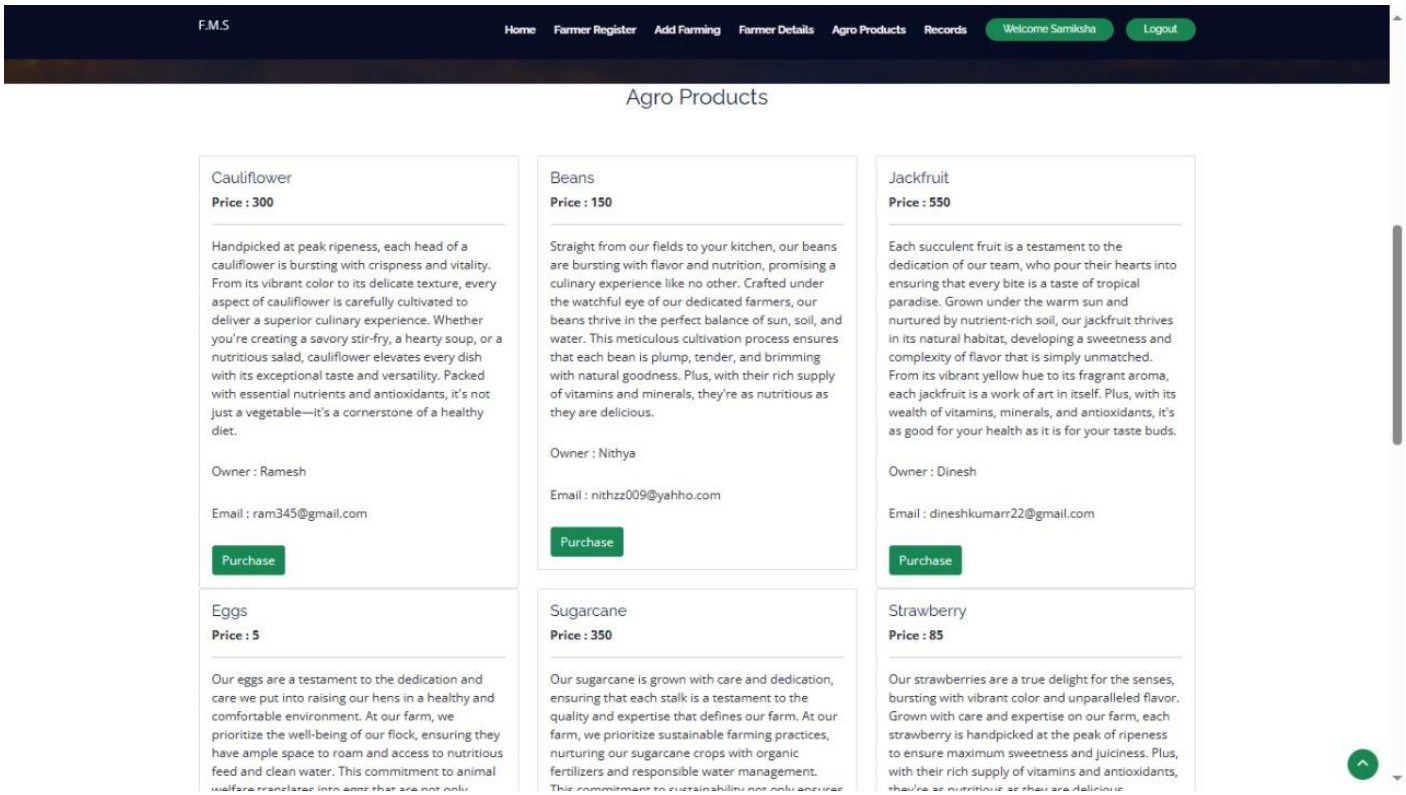


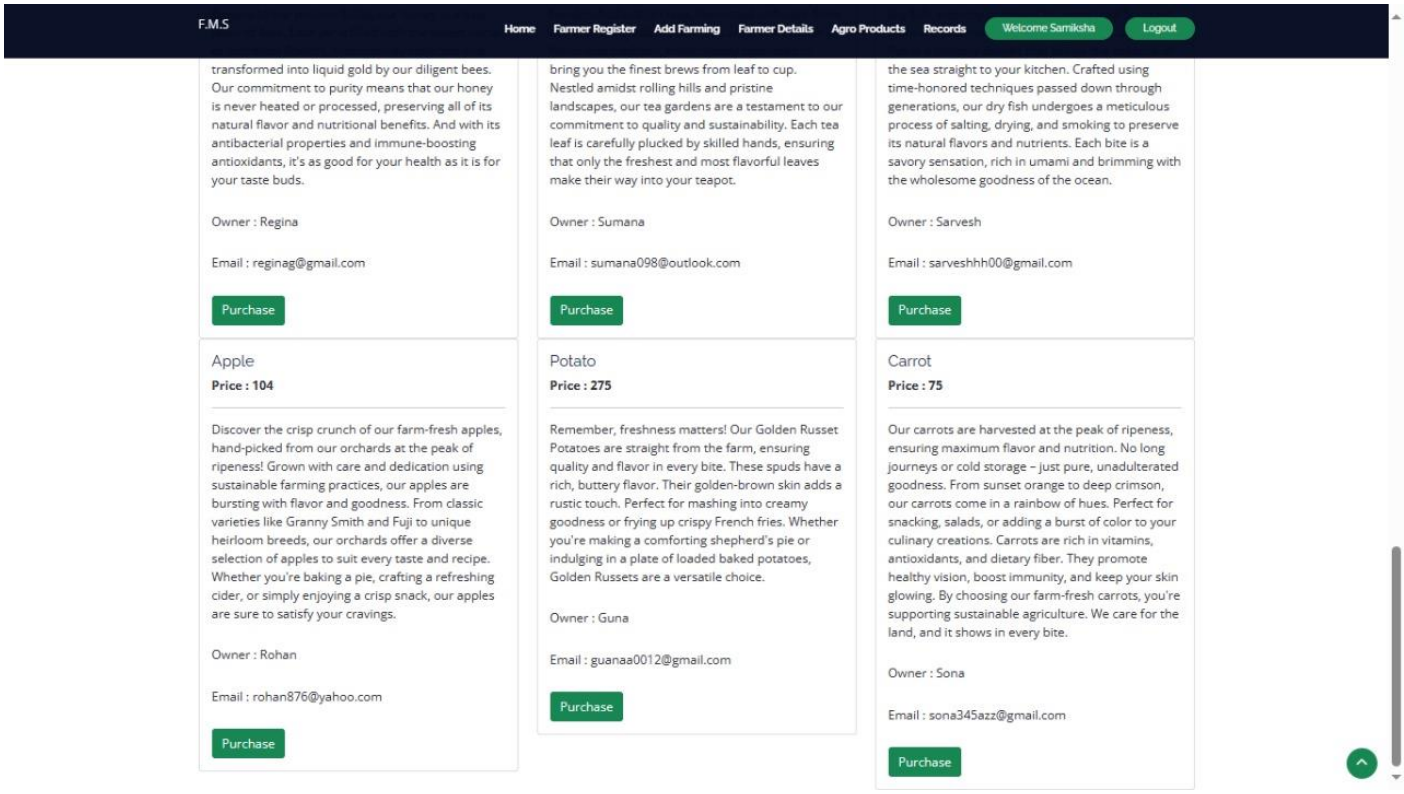
Farmer details Page



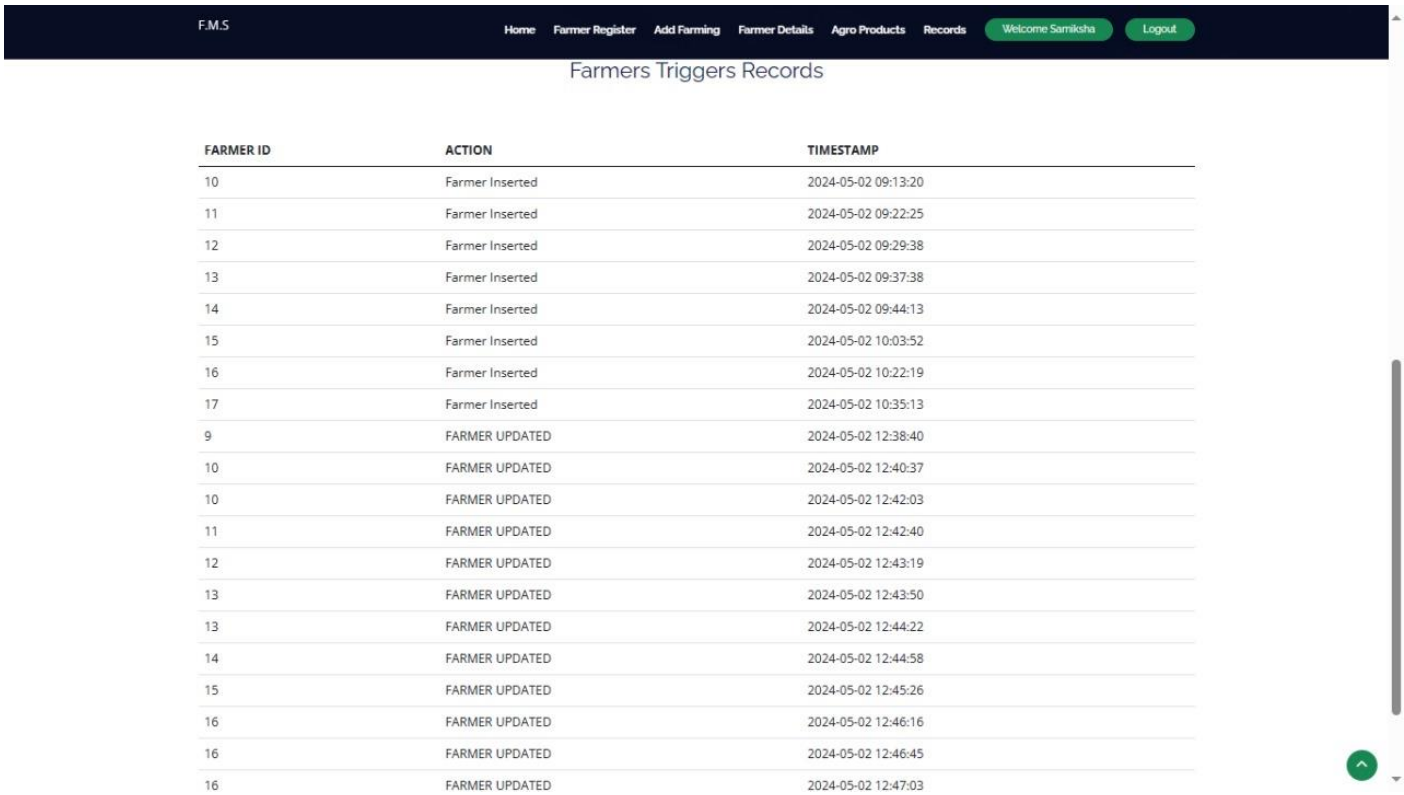
RID	FARMER NAME	ADHAR NUMBER	AGE	GENDER	PHONE NUMBER	ADDRESS	FARMING	EDIT	DELETE	ADD AGRO PRODUCT
9	Ramesh	895321456789	41	male	9985476321	Phase 1 , Nature's place, Shollinganallur, Chennai	Poultry Farming	Edit	Delete	ADD
10	Sathya	671243890587	32	female	9924537872	Plot	Crop Farming	Edit	Delete	ADD
11	Mano	726394712983	37	male	9981327834	Plot no. 2/23, VV Nagar, Kaveri Nagar, Mandya	Fruit Farming	Edit	Delete	ADD
12	Regina	892637590127	29	female	99925386127	Nature market, Phase 2, Malpe, Udupi	Apiculture	Edit	Delete	ADD
13	Sumana	773892340129	45	female	9981722837	Farmfresh, near vsss store, Wellington, Coonoor	Plantation Agriculture	Edit	Delete	ADD
14	Sarvesh	892390182390	36	male	9901926389	Plot	Aquaculture	Edit	Delete	ADD
15	Rohan	871902782910	26	male	9901820256	No 7/11, MN Nagar, Guindy, Chennai	Fruit Farming	Edit	Delete	ADD
16	Guna	238490981238	33	male	9987612309	Farmpro,Near ABC School, Cobaley colony, Bengalur	Vegetable Farming	Edit	Delete	ADD
17	Sona	892748901238	43	female	9986123908	Farmbuy, Phase 3, Near Nellore bus stand, Nellore	Vegetable Farming	Edit	Delete	ADD

Agro products Page



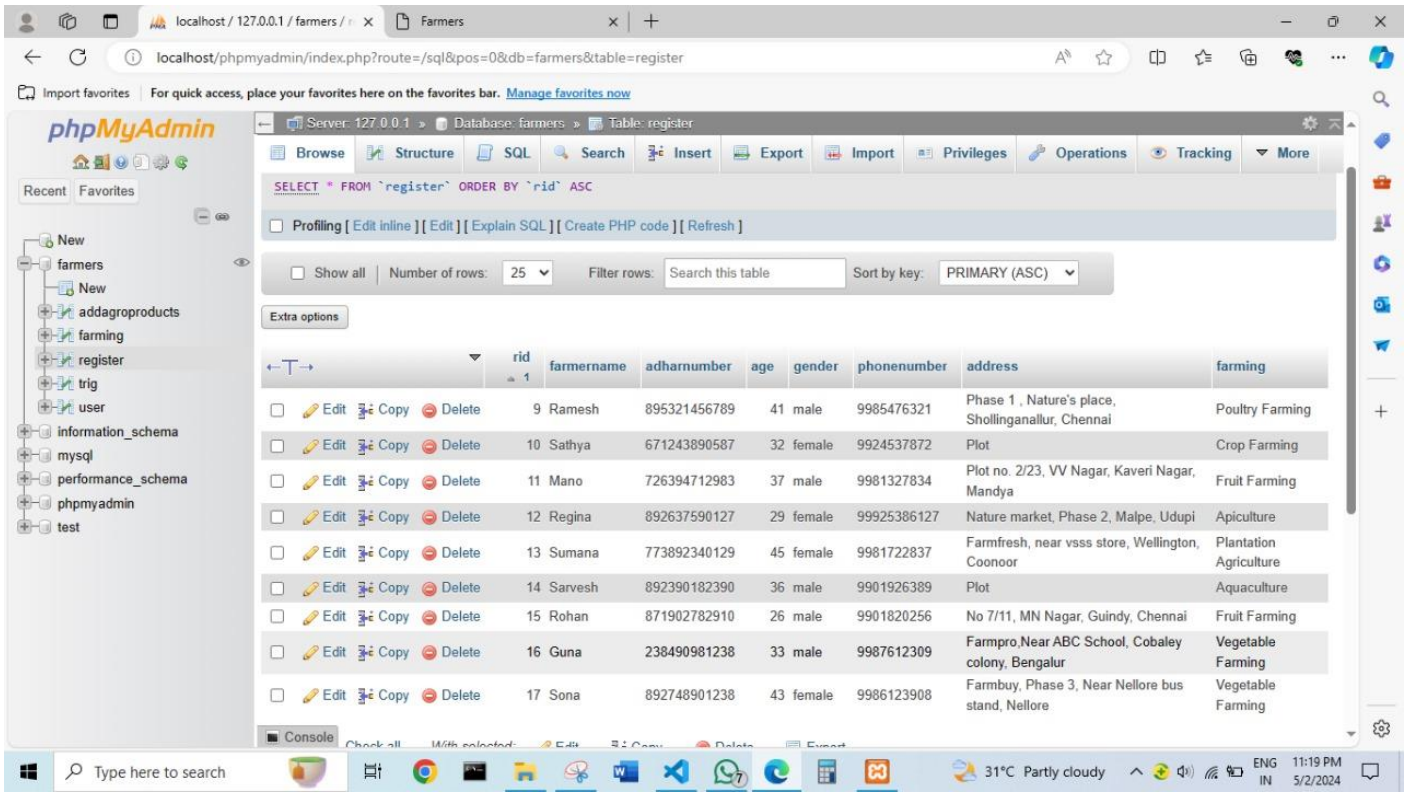


Farmers triggers records Page

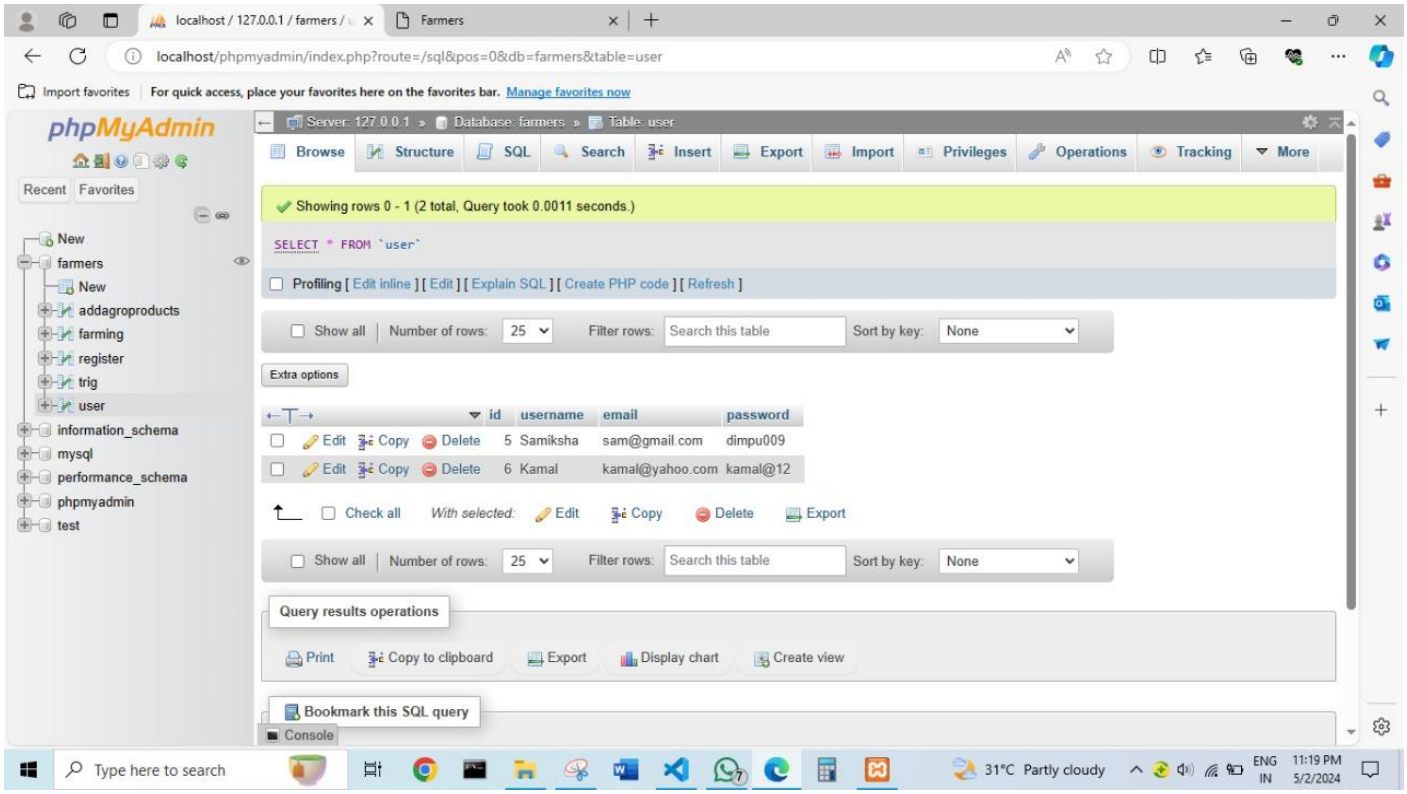


Database Pages:-

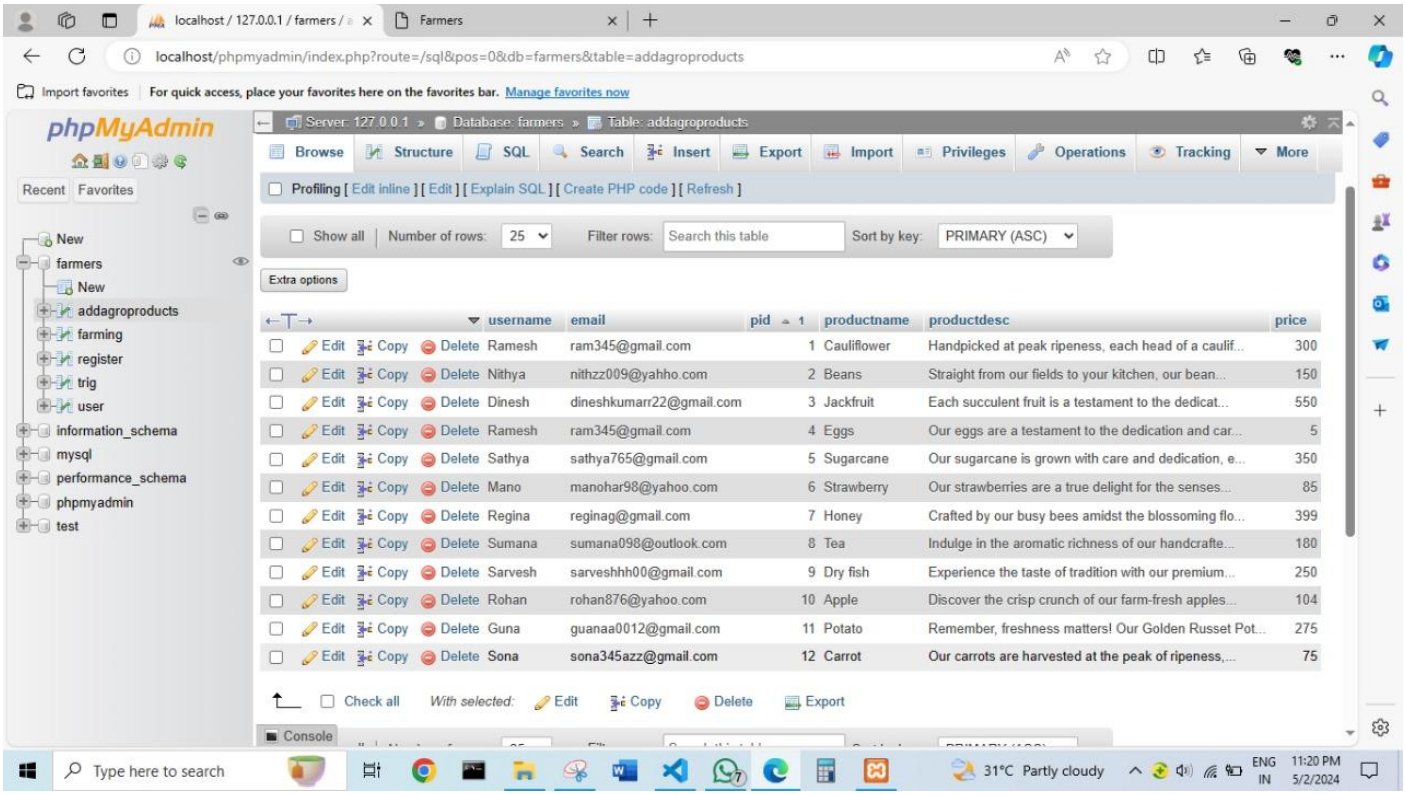
Register Table



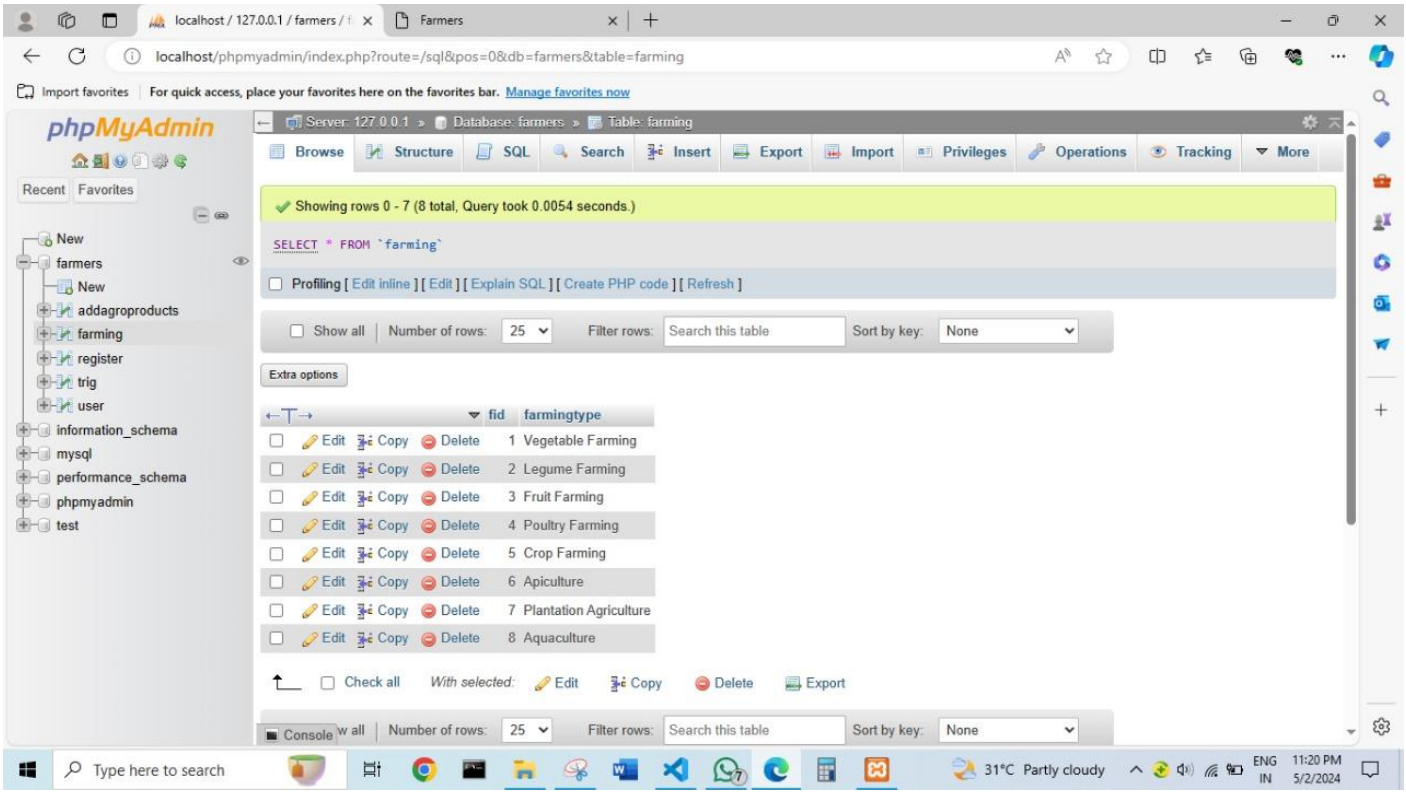
User Table



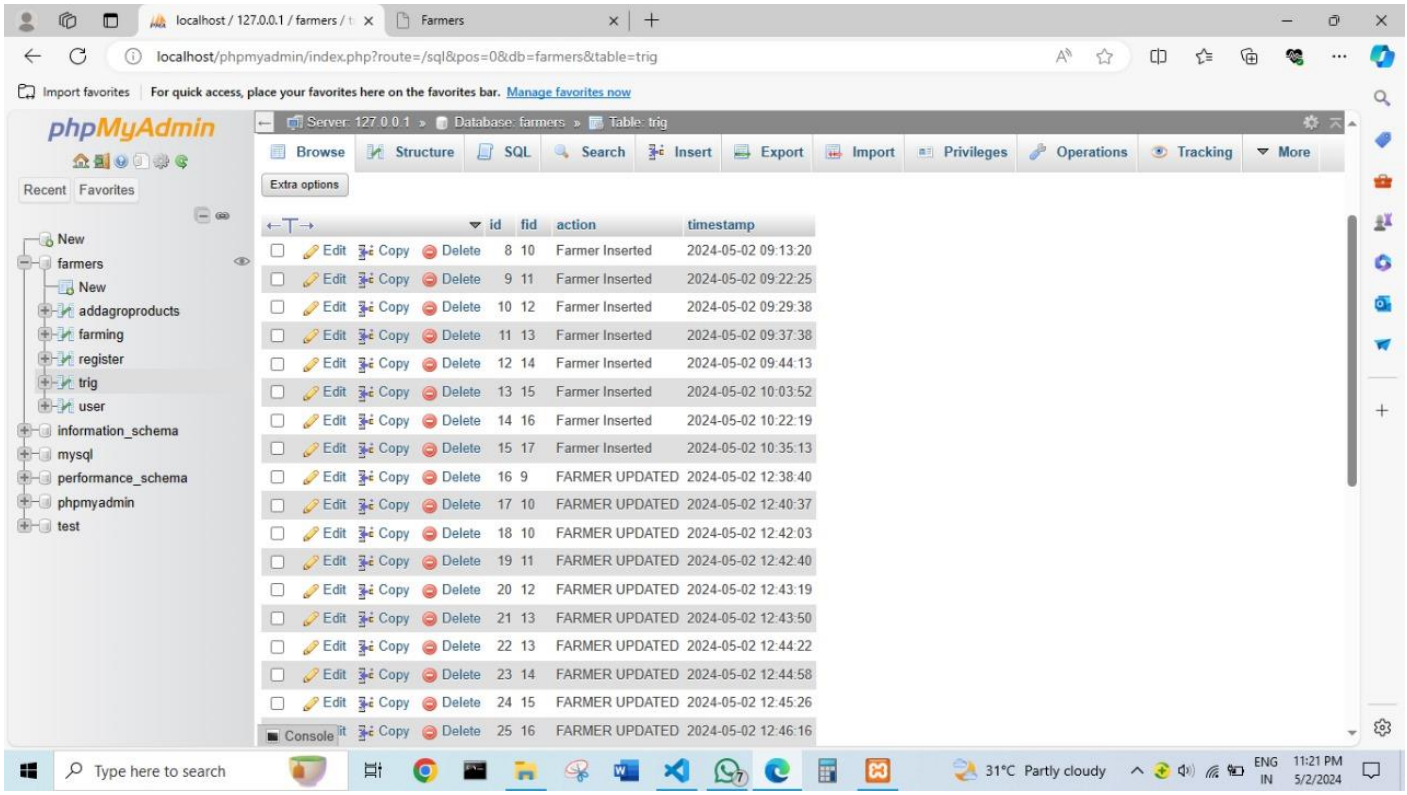
Add agro products Table

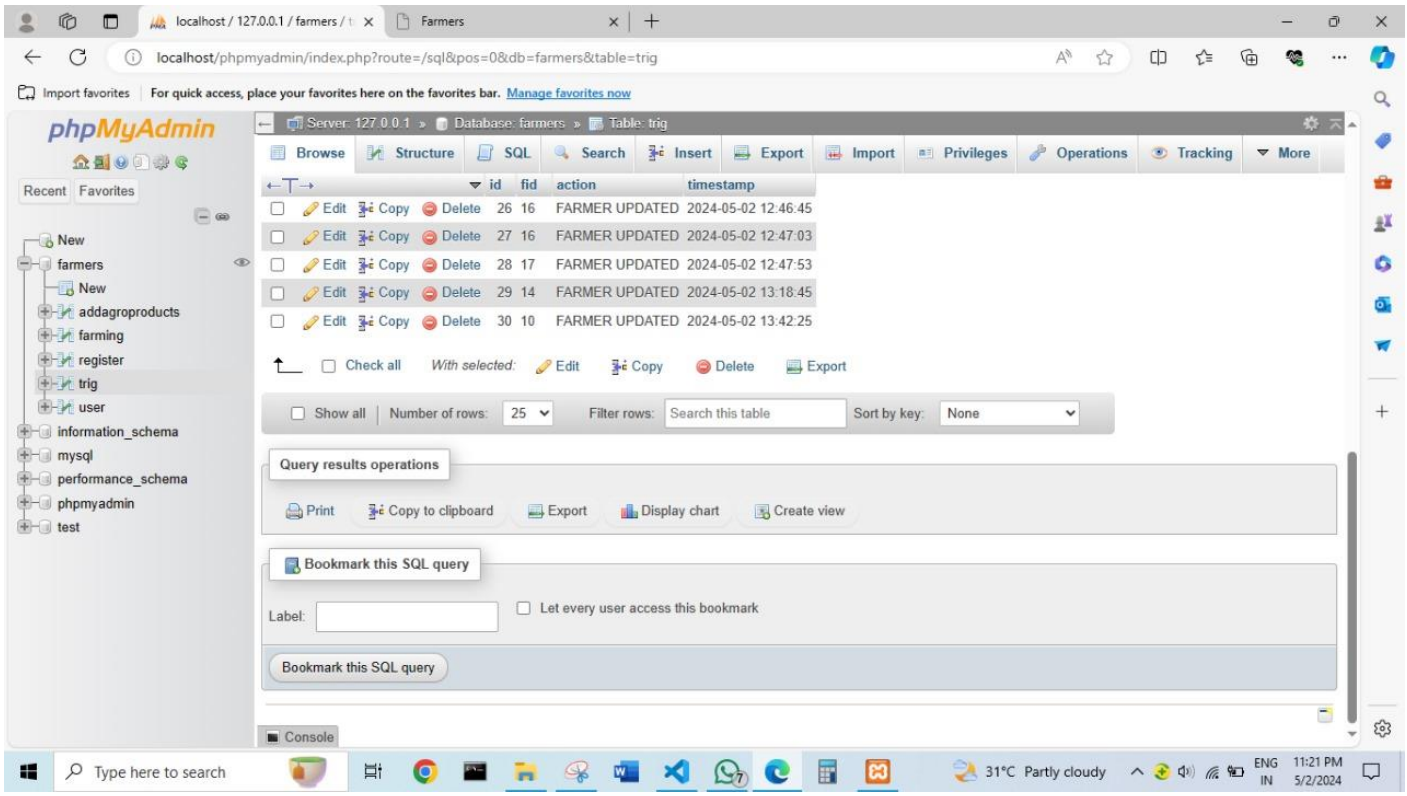


Farming Table



Trig Table


localhost / 127.0.0.1 / farmers / t
Farmers
localhost/phpmyadmin/index.php?route=/sql&pos=0&db=farmers&table=trig
Import favorites
For quick access, place your favorites here on the favorites bar. Manage favorites now
phpMyAdmin
Recent
Favorites
New
farmers
New
addagroproducts
farming
register
trig
user
information_schema
mysql
performance_schema
phpmyadmin
test
Server: 127.0.0.1 » Database: farmers » Table: trig
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Tracking
More
Extra options
id fid action timestamp
Edit Copy Delete 8 10 Farmer Inserted 2024-05-02 09:13:20
Edit Copy Delete 9 11 Farmer Inserted 2024-05-02 09:22:25
Edit Copy Delete 10 12 Farmer Inserted 2024-05-02 09:29:38
Edit Copy Delete 11 13 Farmer Inserted 2024-05-02 09:37:38
Edit Copy Delete 12 14 Farmer Inserted 2024-05-02 09:44:13
Edit Copy Delete 13 15 Farmer Inserted 2024-05-02 10:03:52
Edit Copy Delete 14 16 Farmer Inserted 2024-05-02 10:22:19
Edit Copy Delete 15 17 Farmer Inserted 2024-05-02 10:35:13
Edit Copy Delete 16 9 FARMER UPDATED 2024-05-02 12:38:40
Edit Copy Delete 17 10 FARMER UPDATED 2024-05-02 12:40:37
Edit Copy Delete 18 10 FARMER UPDATED 2024-05-02 12:42:03
Edit Copy Delete 19 11 FARMER UPDATED 2024-05-02 12:42:40
Edit Copy Delete 20 12 FARMER UPDATED 2024-05-02 12:43:19
Edit Copy Delete 21 13 FARMER UPDATED 2024-05-02 12:43:50
Edit Copy Delete 22 13 FARMER UPDATED 2024-05-02 12:44:22
Edit Copy Delete 23 14 FARMER UPDATED 2024-05-02 12:44:58
Edit Copy Delete 24 15 FARMER UPDATED 2024-05-02 12:45:26
Edit Copy Delete 25 16 FARMER UPDATED 2024-05-02 12:46:16
Console
Type here to search
31°C Partly cloudy
ENG IN
11:21 PM 5/2/2024


localhost / 127.0.0.1 / farmers / t
Farmers
localhost/phpmyadmin/index.php?route=/sql&pos=0&db=farmers&table=trig
Import favorites
For quick access, place your favorites here on the favorites bar. Manage favorites now
phpMyAdmin
Recent
Favorites
New
farmers
New
addagroproducts
farming
register
trig
user
information_schema
mysql
performance_schema
phpmyadmin
test
Server: 127.0.0.1 » Database: farmers » Table: trig
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Tracking
More
id fid action timestamp
Edit Copy Delete 26 16 FARMER UPDATED 2024-05-02 12:46:45
Edit Copy Delete 27 16 FARMER UPDATED 2024-05-02 12:47:03
Edit Copy Delete 28 17 FARMER UPDATED 2024-05-02 12:47:53
Edit Copy Delete 29 14 FARMER UPDATED 2024-05-02 13:18:45
Edit Copy Delete 30 10 FARMER UPDATED 2024-05-02 13:42:25
Check all
With selected:
Edit
Copy
Delete
Export
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key: None
Query results operations
Print
Copy to clipboard
Export
Display chart
Create view
Bookmark this SQL query
Label:
Let every user access this bookmark
Bookmark this SQL query
Console
Type here to search
31°C Partly cloudy
ENG IN
11:21 PM 5/2/2024

Chapter 8

Online Course Certification

Scalar certificates:-

1. SAMIKSHA S (RA2211003011100)

S. Samiksha

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

74 Video Tutorials 16 Modules 16 Challenges 17 April 2024

Anshuman Singh
Co-founder SCALER

CERTIFICATE OF EXCELLENCE BY SCALER

2. SADWIKA REDDY (RA2211003011072)

3. BHARATH ROYAL (RA2211003011082)