AGROCULTURE

A PROJECT REPORT

Submitted by

G.N.R.S.S Charan Reddy [Reg No: RA2211003011287]

Mallela Gnanamrutha [Reg No: RA2211003011294]

Kasthuri Harshini [Reg No: RA2211003011299]

Under the Guidance of

Dr. R. Jebakumar

Associate Professor, Department of Computing Technologies

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTING TECHNOLOGIES

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR– 603 203

MAY 2024

BONAFIDE CERTIFICATE

Certified that this project report "**AGROCULTURE**" is the bonafide work of **G.R.N.S.S Charan Reddy [Reg. No. RA2211003011287], M.Gnanamrutha [Reg. No. RA2211003011294], and K.Harshini [Reg. No. RA2211003011299]** who carried out the project work under my supervision in the project course **Database Management Systems [21CSC205P]** for the academic year 2023-2024 [II year / IV semester].

Date:

Faculty in Charge
Dr. R. Jebakumar
Associate Professor
Department of Computing Technologies
SRMSIT -KTR

HEAD OF THE DEPARTMENT
Dr. M. Pushpalatha
Professor
Department of Computing Technologies
SRMIST - KTR

ACKNOWLEDGEMENTS

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professor Dr. C. Lakshmi, Professor, Department of Computational Intelligence** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our m Course Faculty **Dr. R. Jebakumar, Associate Professor, Department of Computing Technologies,** for his assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD Dr. M. Pushpalatha, Professor, Department of Computing Technologies** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

G.R.N.S.S Charan Reddy [RA2211003011287]

Mallela Gnanamrutha [RA2211003011294]

Kasturi Harshini [RA2211003011299]

ABSTRACT

Agroculture is the farmer system where they can plan, monitor, and analyze the activity of the farmer's production system. It manages farmer operation with one system and organizes data in one place. It helps smart farmers become even smarter. This creates in partnership with growers and buyers. It inspires farmer to produce and buyers to consume fresh goods. Agro culture system will make better connection among farmers and buyers ensure quality food. Standardize and increase efficiency of agroculture process.

The adoption of AgroCulture holds immense potential for addressing pressing global challenges, such as food security and climate change. By promoting sustainable farming practices and resource-efficient techniques, this system enables farmers to mitigate environmental impact while enhancing resilience to climate variability. Through precision agriculture and smart irrigation methods, AgroCulture minimizes water usage and chemical inputs, reducing carbon emissions and preserving natural ecosystems. Additionally, by facilitating traceability and transparency in food supply chains, AgroCulture empowers consumers to make informed choices, supporting the demand for ethically produced, environmentally friendly products. In this way, AgroCulture not only ensures the long-term viability of agricultural systems but also contributes to broader efforts towards building a more sustainable and equitable future for all.

TABLE OF CONTENTS

1. Problem Statement

In the rapidly evolving landscape of agricultural commerce, effective inventory management remains a critical challenge for stakeholders within the agro-industry. Many farmers and buyers struggle with outdated or manual inventory tracking systems, resulting in inefficiencies, inaccuracies, and increased operational costs. The absence of a centralized and automated inventory management solution often leads to stockouts, surplus situations, delayed order fulfillment, and difficulties in maintaining optimal inventory levels. Additionally, the lack of real-time visibility into inventory data hinders informed decision-making, including trend identification, demand forecasting, and procurement optimization. Given these challenges, there is a pressing need for a robust Online AgroCulture System that integrates seamlessly with Database Management System (DBMS) principles to streamline inventory processes, enhance data accuracy, and improve operational efficiency.

The aim of this mini project is to design, develop, and implement an OACS that addresses these challenges by offering comprehensive functionalities for inventory tracking, order management, supplier management, and inventory analysis. This web-based platform should provide a user-friendly interface accessible to both farmers and buyers, ensuring data integrity, supporting scalability, and facilitating real-time monitoring of inventory levels. Ultimately, the goal is to empower stakeholders within the agro-industry with a reliable and efficient OACS that optimizes inventory management practices, minimizes costs, and fosters growth in agricultural commerce.

This Agroculture should offer features such as real-time inventory tracking, demand forecasting, inventory optimization, and intuitive reporting tools. By addressing these challenges, the proposed Agroculture aims to empower farmers and buyers to achieve greater operational efficiency, reduce costs, enhance transactional experiences, and gain a competitive advantage in the online agricultural marketplace.

Requirement Analysis

User Requirements:

1. User-friendly interface for easy navigation and intuitive shopping experience.
2. Secure payment gateway to facilitate smooth transactions.
3. Detailed product descriptions, including origin, freshness, and nutritional information.
4. Customizable preferences for organic, locally sourced, or specialty produce.
5. Flexible delivery options, including time slots and subscription services.

Functional Requirements:

1. Comprehensive database of fruits and vegetables with categorization and search filters.
2. High-quality images and detailed descriptions for each product.
3. Pricing information with options for discounts and promotions. b. Shopping Cart and Checkout:
4. Easy addition/removal of items to/from the cart.
5. Multiple payment options including credit/debit cards, digital wallets, and cash on delivery.

SOFTWARE REQUIRMENT:

1.MySQL
2.Python with MySQL Connectivity
3.Python with Tkinter/PyCharm(GUI)

CHAPTER 1

2.1 ER DIAGRAM

ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system. It develops a conceptual design for the database. It also develops a very simple and easy to design view of data. In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

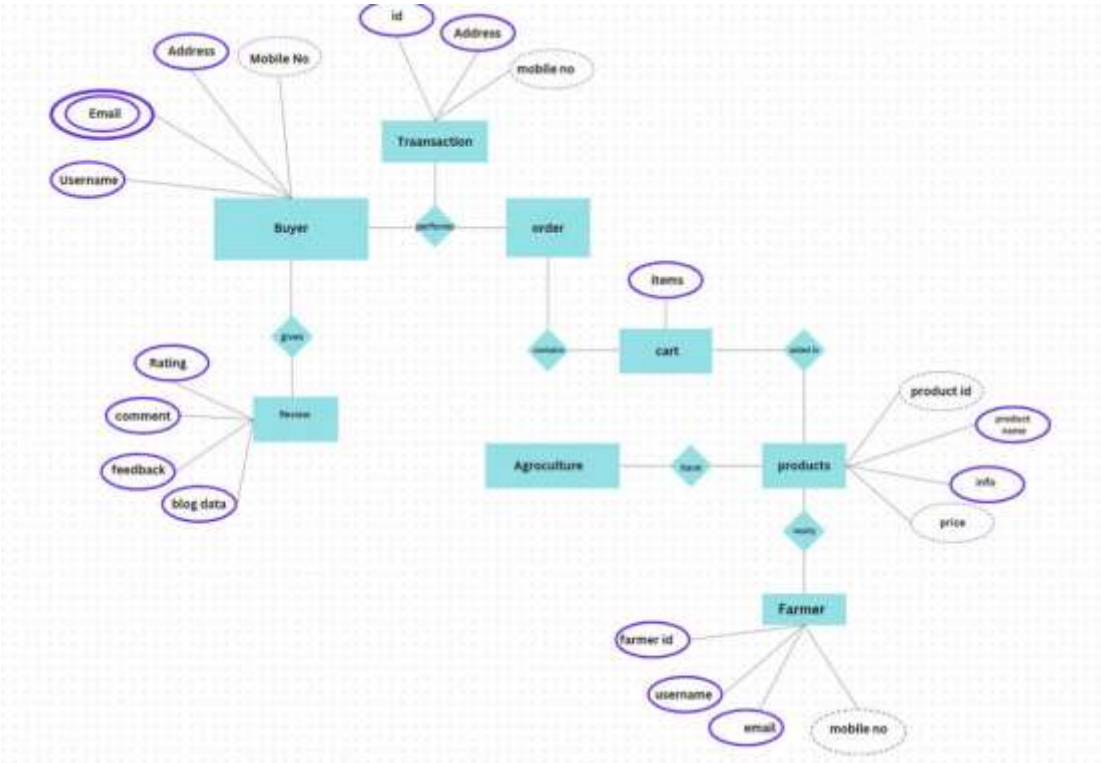

Fig 2.1 ER model

CHAPTER 2

3.RELATIONAL SCHEMA

After designing the conceptual model of the Database using ER diagram, we need to convert the conceptual model into a relational model which can be implemented using any RDBMS language like Oracle SQL, MySQL etc. The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as relations.

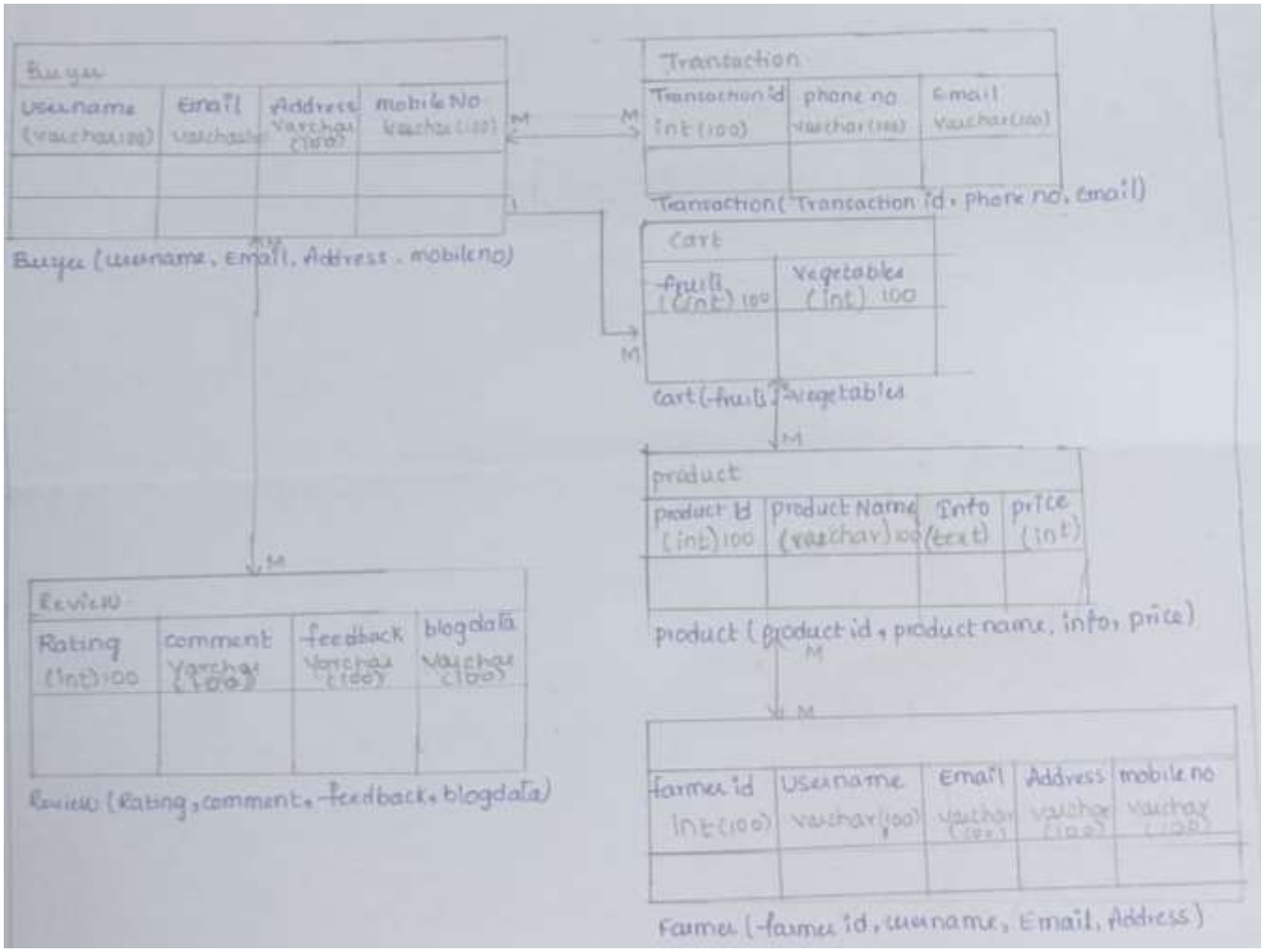

Fig 3.Relational model

CHAPTER 4

4.Complex queries using MYSQL

4.1.CREATING A DATABASE

To create a database in MySQL, you would use the CREATE DATABASE statement followed by the name of the database you want to create.

Here's the basic syntax:

CREATE DATABASE database_name;

CREATING DATABASE FOR AGROCULTURE:

```
mysql> create database agroculture;
Query OK, 1 row affected (0.02 sec)
```

```
mysql> use agroculture;
Database changed
```

4.2CREATING TABLES IN AGROCULTURE

Constraints in SQL are rules defined on columns or tables that enforce certain conditions or relationships. They help maintain the integrity, accuracy, and consistency of data within a database.

```
mysql> CREATE TABLE `farmer`(
    -> `fid` INT AUTO_INCREMENT PRIMARY KEY,
    -> `fname` varchar(255) NOT NULL,
    -> `fusername` varchar(255) NOT NULL,
    -> `fpassword` varchar(255) NOT NULL,
    -> `fhash` varchar(255) NOT NULL,
    -> `femail` varchar(255) NOT NULL,
    -> `fmobile` varchar(255) NOT NULL,
    -> `faddress` text NOT NULL,
    -> `factive` int(255) NOT NULL DEFAULT '0',
    -> `frating` int(11) NOT NULL DEFAULT '0',
    -> `picExt` varchar(255) NOT NULL DEFAULT 'png',
    -> `picStatus` int(10) NOT NULL DEFAULT '0'
    -> );
Query OK, 0 rows affected, 3 warnings (0.04 sec)

mysql> DESC FARMER;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| fid        | int          | NO   | PRI | NULL    | auto_increment |
| fname      | varchar(255) | NO   |     | NULL    |                |
| fusername  | varchar(255) | NO   |     | NULL    |                |
| fpassword  | varchar(255) | NO   |     | NULL    |                |
| fhash      | varchar(255) | NO   |     | NULL    |                |
| femail     | varchar(255) | NO   |     | NULL    |                |
| fmobile    | varchar(255) | NO   |     | NULL    |                |
| faddress   | text         | NO   |     | NULL    |                |
| factive    | int          | NO   |     | 0       |                |
| frating    | int          | NO   |     | 0       |                |
| picExt     | varchar(255) | NO   |     | png     |                |
| picStatus  | int          | NO   |     | 0       |                |
+------------+--------------+------+-----+---------+----------------+
```

Fig 4.2.1 creating database

```
mysql> DESC BUYER;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| bid       | int          | NO   | PRI | NULL    | auto_increment |
| bname     | varchar(100) | NO   |     | NULL    |                |
| busername | varchar(100) | NO   |     | NULL    |                |
| bpassword | varchar(100) | NO   |     | NULL    |                |
| bhash     | varchar(100) | NO   |     | NULL    |                |
| bemail    | varchar(100) | NO   |     | NULL    |                |
| bmobile   | varchar(100) | NO   |     | NULL    |                |
| baddress  | text         | NO   |     | NULL    |                |
| bactive   | int          | NO   |     | 0       |                |
+-----------+--------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)

mysql> CREATE TABLE `fproduct` (
    ->  `fid` int AUTO_INCREMENT PRIMARY KEY,
    ->  `pid` int(255) NOT NULL,
    ->  `product` varchar(255) NOT NULL,
    ->  `pcat` varchar(255) NOT NULL,
    ->  `pinfo` varchar(255) NOT NULL,
    ->  `price` float NOT NULL,
    ->  `pimage` varchar(255) NOT NULL DEFAULT 'blank.png',
    ->  `picStatus` int(10) NOT NULL DEFAULT '0'
    -> );
Query OK, 0 rows affected, 2 warnings (0.05 sec)

mysql> DESC fproduct;
+-----------+--------------+------+-----+-----------+----------------+
| Field     | Type         | Null | Key | Default   | Extra          |
+-----------+--------------+------+-----+-----------+----------------+
| fid       | int          | NO   | PRI | NULL      | auto_increment |
| pid       | int          | NO   |     | NULL      |                |
| product   | varchar(255) | NO   |     | NULL      |                |
| pcat      | varchar(255) | NO   |     | NULL      |                |
| pinfo     | varchar(255) | NO   |     | NULL      |                |
| price     | float        | NO   |     | NULL      |                |
| pimage    | varchar(255) | NO   |     | blank.png |                |
| picStatus | int          | NO   |     | 0         |                |
+-----------+--------------+------+-----+-----------+----------------+
8 rows in set (0.00 sec)

mysql> CREATE TABLE `transaction` (
    -> `tid` INT AUTO_INCREMENT PRIMARY KEY,
    -> `bid` int(10) NOT NULL,
    -> `pid` int(10) NOT NULL,
    -> `name` varchar(255) NOT NULL,
    -> `city` varchar(255) NOT NULL,
    -> `mobile` varchar(255) NOT NULL,
    -> `email` varchar(255) NOT NULL,
    -> `pincode` varchar(255) NOT NULL,
    -> `addr` varchar(255) NOT NULL
    -> );
Query OK, 0 rows affected, 2 warnings (0.05 sec)

mysql> DESC transaction;
+---------+--------------+------+-----+---------+----------------+
| Field   | Type         | Null | Key | Default | Extra          |
+---------+--------------+------+-----+---------+----------------+
| tid     | int          | NO   | PRI | NULL    | auto_increment |
| bid     | int          | NO   |     | NULL    |                |
| pid     | int          | NO   |     | NULL    |                |
| name    | varchar(255) | NO   |     | NULL    |                |
| city    | varchar(255) | NO   |     | NULL    |                |
| mobile  | varchar(255) | NO   |     | NULL    |                |
| email   | varchar(255) | NO   |     | NULL    |                |
| pincode | varchar(255) | NO   |     | NULL    |                |
| addr    | varchar(255) | NO   |     | NULL    |                |
+---------+--------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
```

Fig 4.2.2 creating tables

4.3LIST OF THE TABLES IN THE DATABASE

```
mysql> show tables;
+------------------------+
| Tables_in_agroculture |
+------------------------+
| blogdata               |
| blogfeedback           |
| buyer                  |
| farmer                 |
| fproduct               |
| likedata               |
| mycart                 |
| transaction            |
+------------------------+
8 rows in set (0.00 sec)
```

4.4 TABLE DESCRIPTION IN THE ABROCULTURE DATABASE

```
mysql> DESC FARMER;
+-----------+---------------+------+-----+---------+----------------+
| Field     | Type          | Null | Key | Default | Extra          |
+-----------+---------------+------+-----+---------+----------------+
| fid       | int           | NO   | PRI | NULL    | auto_increment |
| fname     | varchar(255)  | NO   |     | NULL    |                |
| fusername | varchar(255)  | NO   |     | NULL    |                |
| fpassword | varchar(255)  | NO   |     | NULL    |                |
| fhash     | varchar(255)  | NO   |     | NULL    |                |
| femail    | varchar(255)  | NO   |     | NULL    |                |
| fmobile   | varchar(255)  | NO   |     | NULL    |                |
| faddress  | text          | NO   |     | NULL    |                |
| factive   | int           | NO   |     | 0       |                |
| frating   | int           | NO   |     | 0       |                |
| picExt    | varchar(255)  | NO   |     | png     |                |
| picStatus | int           | NO   |     | 0       |                |
+-----------+---------------+------+-----+---------+----------------+

mysql> DESC BUYER;
+-----------+---------------+------+-----+---------+----------------+
| Field     | Type          | Null | Key | Default | Extra          |
+-----------+---------------+------+-----+---------+----------------+
| bid       | int           | NO   | PRI | NULL    | auto_increment |
| bname     | varchar(100)  | NO   |     | NULL    |                |
| busername | varchar(100)  | NO   |     | NULL    |                |
| bpassword | varchar(100)  | NO   |     | NULL    |                |
| bhash     | varchar(100)  | NO   |     | NULL    |                |
| bemail    | varchar(100)  | NO   |     | NULL    |                |
| bmobile   | varchar(100)  | NO   |     | NULL    |                |
| baddress  | text          | NO   |     | NULL    |                |
| bactive   | int           | NO   |     | 0       |                |
+-----------+---------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)

mysql> DESC fproduct;
+-----------+---------------+------+-----+------------+----------------+
| Field     | Type          | Null | Key | Default    | Extra          |
+-----------+---------------+------+-----+------------+----------------+
| fid       | int           | NO   | PRI | NULL       | auto_increment |
| pid       | int           | NO   |     | NULL       |                |
| product   | varchar(255)  | NO   |     | NULL       |                |
| pcat      | varchar(255)  | NO   |     | NULL       |                |
| pinfo     | varchar(255)  | NO   |     | NULL       |                |
| price     | float         | NO   |     | NULL       |                |
| pimage    | varchar(255)  | NO   |     | blank.png  |                |
| picStatus | int           | NO   |     | 0          |                |
+-----------+---------------+------+-----+------------+----------------+
8 rows in set (0.00 sec)
```

Fig 4.4.1 description

```
mysql> DESC transaction;
+---------+-------------+------+-----+---------+----------------+
| Field   | Type        | Null | Key | Default | Extra          |
+---------+-------------+------+-----+---------+----------------+
| tid     | int         | NO   | PRI | NULL    | auto_increment |
| bid     | int         | NO   |     | NULL    |                |
| pid     | int         | NO   |     | NULL    |                |
| name    | varchar(255)| NO   |     | NULL    |                |
| city    | varchar(255)| NO   |     | NULL    |                |
| mobile  | varchar(255)| NO   |     | NULL    |                |
| email   | varchar(255)| NO   |     | NULL    |                |
| pincode | varchar(255)| NO   |     | NULL    |                |
| addr    | varchar(255)| NO   |     | NULL    |                |
+---------+-------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)

+-------------+-------------+------+-----+-------------------+-------------------+
| Field       | Type        | Null | Key | Default           | Extra             |
+-------------+-------------+------+-----+-------------------+-------------------+
| blogId      | int         | NO   | PRI | NULL              | auto_increment    |
| blogUser    | varchar(256)| NO   |     | NULL              |                   |
| blogTitle   | varchar(256)| NO   |     | NULL              |                   |
| blogContent | longtext    | NO   |     | NULL              |                   |
| blogTime    | timestamp   | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| likes       | int         | NO   |     | 0                 |                   |
+-------------+-------------+------+-----+-------------------+-------------------+
6 rows in set (0.00 sec)

mysql> desc blogfeedback;
+-------------+-------------+------+-----+-------------------+-------------------+
| Field       | Type        | Null | Key | Default           | Extra             |
+-------------+-------------+------+-----+-------------------+-------------------+
| blogId      | int         | NO   | PRI | NULL              | auto_increment    |
| comment     | varchar(256)| NO   |     | NULL              |                   |
| commentUser | varchar(256)| NO   |     | NULL              |                   |
| commentPic  | varchar(256)| NO   |     | profile0.png      |                   |
| commentTime | timestamp   | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-------------+-------------+------+-----+-------------------+-------------------+
5 rows in set (0.00 sec)

mysql> desc review;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| pid     | int         | NO   |     | NULL    |       |
| name    | varchar(255)| NO   |     | NULL    |       |
| rating  | int         | NO   |     | NULL    |       |
| comment | text        | NO   |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> desc mycart;
+-------+------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+-------+------+------+-----+---------+-------+
| bid   | int  | NO   |     | NULL    |       |
| pid   | int  | NO   |     | NULL    |       |
+-------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> desc likedata;
+------------+------+------+-----+---------+-------+
| Field      | Type | Null | Key | Default | Extra |
+------------+------+------+-----+---------+-------+
| blogId     | int  | NO   |     | NULL    |       |
| blogUserId | int  | NO   |     | NULL    |       |
+------------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Fig 4.4.2 description

4.5 DATA DEFINITION LANGUAGE (DDL)

4.5.1 CREATE farmer table

CREATE TABLE: This is the SQL command used to create a new table in a MySQL database. Columns for id, name, username, password, hash, email, mobile, address, active, rating, pic Status are defined with their respective data types.

```
mysql> CREATE TABLE `farmer`(
    -> `fid` INT AUTO_INCREMENT PRIMARY KEY,
    -> `fname` varchar(255) NOT NULL,
    -> `fusername` varchar(255) NOT NULL,
    -> `fpassword` varchar(255) NOT NULL,
    -> `fhash` varchar(255) NOT NULL,
    -> `femail` varchar(255) NOT NULL,
    -> `fmobile` varchar(255) NOT NULL,
    -> `faddress` text NOT NULL,
    -> `factive` int(255) NOT NULL DEFAULT '0',
    -> `frating` int(11) NOT NULL DEFAULT '0',
    -> `picExt` varchar(255) NOT NULL DEFAULT 'png',
    -> `picStatus` int(10) NOT NULL DEFAULT '0'
    -> );
Query OK, 0 rows affected, 3 warnings (0.04 sec)

mysql> DESC FARMER;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| fid       | int          | NO   | PRI | NULL    | auto_increment |
| fname     | varchar(255) | NO   |     | NULL    |                |
| fusername | varchar(255) | NO   |     | NULL    |                |
| fpassword | varchar(255) | NO   |     | NULL    |                |
| fhash     | varchar(255) | NO   |     | NULL    |                |
| femail    | varchar(255) | NO   |     | NULL    |                |
| fmobile   | varchar(255) | NO   |     | NULL    |                |
| faddress  | text         | NO   |     | NULL    |                |
| factive   | int          | NO   |     | 0       |                |
| frating   | int          | NO   |     | 0       |                |
| picExt    | varchar(255) | NO   |     | png     |                |
| picStatus | int          | NO   |     | 0       |                |
+-----------+--------------+------+-----+---------+----------------+
```

Fig 4.5.1 description

4.5.2DELETE

SQL's Command 'DROP TABLE' used to delete database objects such as tables, indexes, views, or databases themselves.

```
mysql> drop table review;
Query OK, 0 rows affected (0.04 sec)

mysql> desc review;
ERROR 1146 (42S02): Table 'agroculture.review' doesn't exist
```

Fig 4.5.2 delete table

4.5.3ALTER

The ‘ALTER’ Command in SQL is used to modify existing database objects, such as tables, views, or indexes.

```
mysql> select * from buyer;
+-----+-------------+------------+-----------+-------+---------------------+------------+-----------+---------+--------+
| bid | bname       | busername  | bpassword | bhash | bemail              | bmobile    | baddress  | bactive | review |
+-----+-------------+------------+-----------+-------+---------------------+------------+-----------+---------+--------+
|   1 | harshini    | harshini123 | 1234      | 1     | harshini123@gmail.com | 987654321  | bapatla   |       1 | good   |
|   2 | Gnanamrutha | amrutha123 | 5678      | 2     | amrutha123@gmail.com | 123456789  | anantapur |       2 | good   |
|   3 | rupesh      | rupesh123  | 4321      | 3     | rupesh123@gmail.com | 9898989898 | chennai   |       3 | good   |
+-----+-------------+------------+-----------+-------+---------------------+------------+-----------+---------+--------+
3 rows in set (0.02 sec)

mysql> ALTER table buyer
    -> DROP column review;
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from buyer;
+-----+-------------+------------+-----------+-------+---------------------+------------+-----------+---------+
| bid | bname       | busername  | bpassword | bhash | bemail              | bmobile    | baddress  | bactive |
+-----+-------------+------------+-----------+-------+---------------------+------------+-----------+---------+
|   1 | harshini    | harshini123 | 1234      | 1     | harshini123@gmail.com | 987654321  | bapatla   |       1 |
|   2 | Gnanamrutha | amrutha123 | 5678      | 2     | amrutha123@gmail.com | 123456789  | anantapur |       2 |
|   3 | rupesh      | rupesh123  | 4321      | 3     | rupesh123@gmail.com | 9898989898 | chennai   |       3 |
+-----+-------------+------------+-----------+-------+---------------------+------------+-----------+---------+
3 rows in set (0.01 sec)
```

Fig 4.5.3 alter

4.6 DATA MANIPULATION LANGUAGE (DML)

4.6.1 INSERT

The INSERT statement in SQL is used to insert new records into a table.

```
mysql> INSERT INTO buyer (bid, bname, busername, bpassword, bhash, bemail, bmobile, baddress, bactive) VALUES
    ->      (1, 'harshini', 'harshini123', '1234', '1', 'harshini123@gmail.com', '987654321', 'bapatla',1),
    ->      (2, 'amrutha', 'amrutha123', '5678', '2', 'amrutha123@gmail.com', '123456789', 'anantapur',2),
    ->      (3, 'nani', 'nani123', '8765', '3', 'nani123@gmail.com', '9898989898', 'hyderabad',3),
    ->       (4, 'ooha', 'ooha123', '4321', '4', 'ooha123@gmail.com', '1231231231', 'bengaluru',4);
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select * from buyer;
+-----+----------+-------------+-----------+-------+---------------------+------------+-----------+---------+
| bid | bname    | busername   | bpassword | bhash | bemail              | bmobile    | baddress  | bactive |
+-----+----------+-------------+-----------+-------+---------------------+------------+-----------+---------+
|   1 | harshini | harshini123 | 1234      | 1     | harshini123@gmail.com | 987654321  | bapatla   |       1 |
|   2 | amrutha  | amrutha123  | 5678      | 2     | amrutha123@gmail.com  | 123456789  | anantapur |       2 |
|   3 | nani     | nani123     | 8765      | 3     | nani123@gmail.com     | 9898989898 | hyderabad |       3 |
|   4 | ooha     | ooha123     | 4321      | 4     | ooha123@gmail.com     | 1231231231 | bengaluru |       4 |
+-----+----------+-------------+-----------+-------+---------------------+------------+-----------+---------+
4 rows in set (0.00 sec)

mysql> INSERT INTO buyer (bid, bname, busername, bpassword, bhash, bemail, bmobile, baddress, bactive)
    -> VALUES (5, 'lavanya', 'lavanya123', '9876', '5', 'lavanya123@gmail.com', '7676767676', 'chennai', 5);
Query OK, 1 row affected (0.01 sec)

mysql> select * from buyer;
+-----+----------+-------------+-----------+-------+---------------------+------------+-----------+---------+
| bid | bname    | busername   | bpassword | bhash | bemail              | bmobile    | baddress  | bactive |
+-----+----------+-------------+-----------+-------+---------------------+------------+-----------+---------+
|   1 | harshini | harshini123 | 1234      | 1     | harshini123@gmail.com | 987654321  | bapatla   |       1 |
|   2 | amrutha  | amrutha123  | 5678      | 2     | amrutha123@gmail.com  | 123456789  | anantapur |       2 |
|   3 | nani     | nani123     | 8765      | 3     | nani123@gmail.com     | 9898989898 | hyderabad |       3 |
|   4 | ooha     | ooha123     | 4321      | 4     | ooha123@gmail.com     | 1231231231 | bengaluru |       4 |
|   5 | lavanya  | lavanya123  | 9876      | 5     | lavanya123@gmail.com  | 7676767676 | chennai   |       5 |
+-----+----------+-------------+-----------+-------+---------------------+------------+-----------+---------+
5 rows in set (0.00 sec)
```

Fig 4.6 insertion

4.6.2 UPDATE

The UPDATE statement is used to modify existing records in a table.

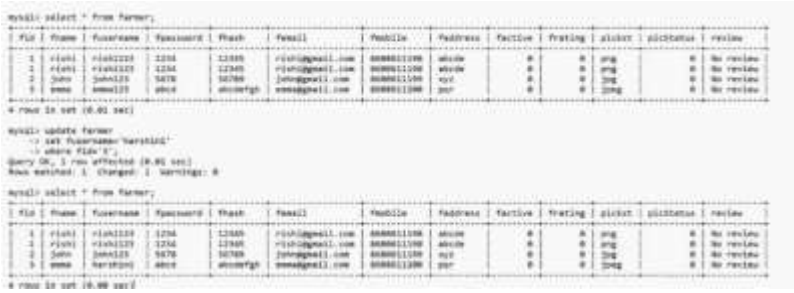

Fig 4.6.2.update

4.6.3 DELETE

The DELETE statement is used to remove existing records from a table in a database.
You can use it to delete specific records based on a condition, or delete all records from a table.

```
mysql> select * from farmer;
+-----+-------+-----------+-----------+---------+-----------------+------------+----------+---------+---------+--------+-----------+-----------+
| fid | fname | fusername | fpassword | fhash   | femail          | fmobile    | faddress | factive | frating | picExt | picStatus | review    |
+-----+-------+-----------+-----------+---------+-----------------+------------+----------+---------+---------+--------+-----------+-----------+
|   1 | rishi | rishi123  | 1234      | 12345   | rishi@gmail.com | 8600611198 | abcde    |       0 |       0 | png    |         0 | No review |
|   1 | rishi | rishi123  | 1234      | 12345   | rishi@gmail.com | 8600611198 | abcde    |       0 |       0 | png    |         0 | No review |
|   2 | john  | john123   | 5678      | 56789   | john@gmail.com  | 8600611199 | xyz      |       0 |       0 | jpg    |         0 | No review |
|   3 | emma  | harshini  | abcd      | abcdefgh | emma@gmail.com | 8600611200 | pqr      |       0 |       0 | jpeg   |         0 | No review |
+-----+-------+-----------+-----------+---------+-----------------+------------+----------+---------+---------+--------+-----------+-----------+
4 rows in set (0.00 sec)

mysql> delete from farmer
    -> where fid=1;
Query OK, 2 rows affected (0.01 sec)

mysql> select * from farmer;
+-----+-------+-----------+-----------+---------+-----------------+------------+----------+---------+---------+--------+-----------+-----------+
| fid | fname | fusername | fpassword | fhash   | femail          | fmobile    | faddress | factive | frating | picExt | picStatus | review    |
+-----+-------+-----------+-----------+---------+-----------------+------------+----------+---------+---------+--------+-----------+-----------+
|   2 | john  | john123   | 5678      | 56789   | john@gmail.com  | 8600611199 | xyz      |       0 |       0 | jpg    |         0 | No review |
|   3 | emma  | harshini  | abcd      | abcdefgh | emma@gmail.com | 8600611200 | pqr      |       0 |       0 | jpeg   |         0 | No review |
+-----+-------+-----------+-----------+---------+-----------------+------------+----------+---------+---------+--------+-----------+-----------+
2 rows in set (0.00 sec)
```

Fig 4.6.3 delete

4.6.4 SELECT

The SELECT statement is used to retrieve data from one or more tables in a database.
You can specify which columns you want to retrieve, as well as filter the rows based on certain conditions.

```
mysql> select * from transaction;
+-----+-----+-----+----------+-----------+------------+----------------------+---------+-----------+
| tid | bid | pid | name     | city      | mobile     | email                | pincode | addr      |
+-----+-----+-----+----------+-----------+------------+----------------------+---------+-----------+
|   1 |   3 |  28 | harshini | bapatla   | 987654321  | harshini123@gmail.com | 12345   | bapatla   |
|   2 |   4 |  29 | amrutha  | chennai   | 876543219  | ammu@gmail.com       | 887766  | tamilnadu |
|   5 |   7 |  34 | hema     | hyderabad | 9833221421 | hema123@gmail.com    | 1298745 | hyd       |
+-----+-----+-----+----------+-----------+------------+----------------------+---------+-----------+
```

Fig 4.6.4 select

4.7 TRANSACTION CONTROL LANGUAGE (TCL)

Transactional Control Language (TCL) commands in SQL are used to manage transactions within a database.

COMMIT: The COMMIT command is used to permanently save changes made during the current transaction to the database.

ROLLBACK: The ROLLBACK command is used to undo changes made during the current transaction and restore the database to its state before the transaction began.

SAVEPOINT: The SAVEPOINT command is used to set a named point within a transaction to which you can later roll back.

```
mysql> select * from fproduct;
+-----+-----+---------+----------+----------------+-------+-----------+-----------+--------+
| fid | pid | product | pcat     | pinfo          | price | pimage    | picStatus | review |
+-----+-----+---------+----------+----------------+-------+-----------+-----------+--------+
|   1 | 123 | grapes  | fruit    | fresh fruit    |   100 | Mango3.jpeg |         1 | good   |
|   2 | 456 | apple   | fruit    | fresh fruit    |   150 | apple.jpeg  |         2 | good   |
|   3 | 789 | tomato  | vegetable| fresh vegetable|    50 | tomato.jpeg |         3 | good   |
|   4 | 987 | onoin   | vegetable| fresh vegetable|   200 | onion.jpeg  |         4 | good   |
+-----+-----+---------+----------+----------------+-------+-----------+-----------+--------+
4 rows in set (0.01 sec)

mysql> start transaction
    -> ;
Query OK, 0 rows affected (0.01 sec)

mysql> savepoint beforeupdates;
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE fproduct
    -> SET review = 'nice'
    -> WHERE fid = 3;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from fproduct;
+-----+-----+---------+----------+----------------+-------+-----------+-----------+--------+
| fid | pid | product | pcat     | pinfo          | price | pimage    | picStatus | review |
+-----+-----+---------+----------+----------------+-------+-----------+-----------+--------+
|   1 | 123 | grapes  | fruit    | fresh fruit    |   100 | Mango3.jpeg |         1 | good   |
|   2 | 456 | apple   | fruit    | fresh fruit    |   150 | apple.jpeg  |         2 | good   |
|   3 | 789 | tomato  | vegetable| fresh vegetable|    50 | tomato.jpeg |         3 | nice   |
|   4 | 987 | onoin   | vegetable| fresh vegetable|   200 | onion.jpeg  |         4 | good   |
+-----+-----+---------+----------+----------------+-------+-----------+-----------+--------+
4 rows in set (0.00 sec)

mysql> ROLLBACK TO SAVEPOINT beforeupdates;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from fproduct;
+-----+-----+---------+----------+----------------+-------+-----------+-----------+--------+
| fid | pid | product | pcat     | pinfo          | price | pimage    | picStatus | review |
+-----+-----+---------+----------+----------------+-------+-----------+-----------+--------+
|   1 | 123 | grapes  | fruit    | fresh fruit    |   100 | Mango3.jpeg |         1 | good   |
|   2 | 456 | apple   | fruit    | fresh fruit    |   150 | apple.jpeg  |         2 | good   |
|   3 | 789 | tomato  | vegetable| fresh vegetable|    50 | tomato.jpeg |         3 | good   |
|   4 | 987 | onoin   | vegetable| fresh vegetable|   200 | onion.jpeg  |         4 | good   |
+-----+-----+---------+----------+----------------+-------+-----------+-----------+--------+
4 rows in set (0.00 sec)
```

Fig 4.7.1 transaction table

CHAPTER 5

5.JOIN COMMANDS

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

```
mysql> select *
    -> from fproduct
    -> natural join farmer;
Empty set (0.01 sec)

mysql> select * from fproduct
    -> natural join likedata;
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+--------+--------+------------+
| fid | pid | product | pcat      | pinfo           | price | pimage      | picStatus | review | blogId | blogUserId |
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+--------+--------+------------+
|   1 | 123 | grapes  | Fruit     | fresh fruit     |   100 | Mango3.jpeg |         1 | good   |     19 |          3 |
|   2 | 456 | apple   | Fruit     | fresh fruit     |   150 | apple.jpeg  |         2 | good   |     19 |          3 |
|   3 | 789 | tomato  | vegetable | fresh vegetable |    50 | tomato.jpeg |         3 | good   |     19 |          3 |
|   4 | 987 | onoin   | vegetable | fresh vegetable |   200 | onion.jpeg  |         4 | good   |     19 |          3 |
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+--------+--------+------------+
4 rows in set (0.00 sec)
```

Fig 5. join table

5.1.1 INNER JOIN

An INNER JOIN in SQL is used to combine rows from two or more tables based on a related column between them. It returns only the rows where there is a match between the columns in the tables being joined.

```
mysql> select * from fproduct;
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+--------+--------+------------+
| fid | pid | product | pcat      | pinfo           | price | pimage      | picStatus | review | blogId | blogUserId |
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+--------+--------+------------+
|   1 | 123 | grapes  | Fruit     | fresh fruit     |   100 | Mango3.jpeg |         1 | good   |     19 |          3 |
|   2 | 456 | apple   | Fruit     | fresh fruit     |   150 | apple.jpeg  |         2 | good   |     19 |          3 |
|   3 | 789 | tomato  | vegetable | fresh vegetable |    50 | tomato.jpeg |         3 | good   |     19 |          3 |
|   4 | 987 | onoin   | vegetable | fresh vegetable |   200 | onion.jpeg  |         4 | good   |     19 |          3 |
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+--------+--------+------------+
4 rows in set (0.00 sec)

mysql> select * from review;
+-----+----------+--------+---------+
| pid | name     | rating | comment |
+-----+----------+--------+---------+
| 123 | harshini |      5 | good    |
| 987 | amrutha  |      4 | nice    |
| 456 | hema     |      3 | better  |
+-----+----------+--------+---------+
3 rows in set (0.00 sec)

mysql> select fproduct.product,fproduct.price,review.name,review.rating
    -> from fproduct
    -> INNER JOIN review on fproduct.pid=review.pid;
+---------+-------+----------+--------+
| product | price | name     | rating |
+---------+-------+----------+--------+
| grapes  |   100 | harshini |      5 |
| apple   |   150 | hema     |      3 |
| onoin   |   200 | amrutha  |      4 |
+---------+-------+----------+--------+
3 rows in set (0.00 sec)
```

Fig 5.1.1 inner join table

5.1.2 LEFT JOIN

A LEFT JOIN in SQL is used to return all rows from the left table (the table specified before the LEFT JOIN keyword), along with matched rows from the right table (the table specified after the LEFT JOIN keyword). If there is no match in the right table, NULL values are returned for the columns from the right table.

```
mysql> select * from fproduct
    -> left join review
    -> on fproduct.pid=review.pid;
+-----+-----+---------+-----------+----------------+-------+-----------+-----------+--------+------+----------+--------+---------+
| fid | pid | product | pcat      | pinfo          | price | pimage    | picStatus | review | pid  | name     | rating | comment |
+-----+-----+---------+-----------+----------------+-------+-----------+-----------+--------+------+----------+--------+---------+
|   1 | 123 | grapes  | Fruit     | fresh fruit    |   100 | Mango3.jpeg |         1 | good   |  123 | harshini |      5 | good    |
|   2 | 456 | apple   | Fruit     | fresh fruit    |   150 | apple.jpeg |         2 | good   |  456 | hema     |      3 | better  |
|   3 | 789 | tomato  | vegetable | fresh vegetable |    50 | tomato.jpeg |         3 | good   | NULL | NULL     |   NULL | NULL    |
|   4 | 987 | onoin   | vegetable | fresh vegetable |   200 | onion.jpeg |         4 | good   |  987 | amrutha  |      4 | nice    |
+-----+-----+---------+-----------+----------------+-------+-----------+-----------+--------+------+----------+--------+---------+
4 rows in set (0.00 sec)
```

5.1.3 RIGHT JOIN

A RIGHT JOIN in SQL is similar to a LEFT JOIN, but it returns all rows from the right table (the table specified after the RIGHT JOIN keyword), along with matched rows from the left table. If there is no match in the left table, NULL values are returned for the columns from the left table.

```
mysql> select * from fproduct;
+-----+-----+---------+-----------+----------------+-------+-----------+-----------+--------+
| fid | pid | product | pcat      | pinfo          | price | pimage    | picStatus | review |
+-----+-----+---------+-----------+----------------+-------+-----------+-----------+--------+
|   1 | 123 | grapes  | Fruit     | fresh fruit    |   100 | Mango3.jpeg |         1 | good   |
|   2 | 456 | apple   | Fruit     | fresh fruit    |   150 | apple.jpeg |         2 | good   |
|   3 | 789 | tomato  | vegetable | fresh vegetable |    50 | tomato.jpeg |         3 | good   |
|   4 | 987 | onoin   | vegetable | fresh vegetable |   200 | onion.jpeg |         4 | good   |
+-----+-----+---------+-----------+----------------+-------+-----------+-----------+--------+
4 rows in set (0.00 sec)

mysql> select * from review;
+-----+----------+--------+---------+
| pid | name     | rating | comment |
+-----+----------+--------+---------+
| 123 | harshini |      5 | good    |
| 987 | amrutha  |      4 | nice    |
| 456 | hema     |      3 | better  |
+-----+----------+--------+---------+

mysql> select * from fproduct
    -> right join review
    -> on fproduct.pid=review.pid;
+------+------+---------+-----------+----------------+-------+-----------+-----------+--------+------+----------+--------+---------+
| fid  | pid  | product | pcat      | pinfo          | price | pimage    | picStatus | review | pid  | name     | rating | comment |
+------+------+---------+-----------+----------------+-------+-----------+-----------+--------+------+----------+--------+---------+
|    1 |  123 | grapes  | Fruit     | fresh fruit    |   100 | Mango3.jpeg |         1 | good   |  123 | harshini |      5 | good    |
|    4 |  987 | onoin   | vegetable | fresh vegetable |   200 | onion.jpeg |         4 | good   |  987 | amrutha  |      4 | nice    |
|    2 |  456 | apple   | Fruit     | fresh fruit    |   150 | apple.jpeg |         2 | good   |  456 | hema     |      3 | better  |
+------+------+---------+-----------+----------------+-------+-----------+-----------+--------+------+----------+--------+---------+
3 rows in set (0.00 sec)
```

5.2.UNION

UNION is used to combine the results of two or more SELECT statements.

```
mysql> select * from fproduct;
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+--------+
| fid | pid | product | pcat      | pinfo           | price | pimage      | picStatus | review |
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+--------+
|   1 | 123 | grapes  | Fruit     | fresh fruit     |   100 | Mango3.jpeg |         1 | good   |
|   2 | 456 | apple   | Fruit     | fresh fruit     |   150 | apple.jpeg  |         2 | good   |
|   3 | 789 | tomato  | vegetable | fresh vegetable |    50 | tomato.jpeg |         3 | good   |
|   4 | 987 | onoin   | vegetable | fresh vegetable |   200 | onion.jpeg  |         4 | good   |
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+--------+
4 rows in set (0.00 sec)

mysql> select * from review;
+-----+----------+--------+---------+
| pid | name     | rating | comment |
+-----+----------+--------+---------+
| 123 | harshini |      5 | good    |
| 987 | amrutha  |      4 | nice    |
| 456 | hema     |      3 | better  |
+-----+----------+--------+---------+


mysql> select pid from fproduct
    -> union
    -> select pid from review
    -> ORDER BY pid;
+-----+
| pid |
+-----+
| 123 |
| 456 |
| 789 |
| 987 |
+-----+
4 rows in set (0.01 sec)
```

Fig 5.2 using union for fproduct

5.3.TRIGGER

A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

```
mysql> CREATE TRIGGER update_password_trigger_update
    -> BEFORE UPDATE ON buyer
    -> FOR EACH ROW
    -> BEGIN
    ->     IF NEW.bid IN (1, 2) THEN
    ->         SET NEW.bpassword = '4545';
    ->     END IF;
    -> END//
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;


mysql> UPDATE buyer SET bname = 'Harshini','Amrutha' bpassword = 'newpassword' WHERE bid IN (1, 2);
Query OK, 2 rows affected (0.01 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from buyer;
+-----+----------+------------+-----------+-------+----------------------+------------+-----------+---------+
| bid | bname    | busername  | bpassword | bhash | bemail               | bmobile    | baddress  | bactive |
+-----+----------+------------+-----------+-------+----------------------+------------+-----------+---------+
|   1 | Harshini | harshini123 | 4545      | 1     | harshini123@gmail.com | 987654321  | bapatla   |       1 |
|   2 | Amrutha  | amrutha123  | 4545      | 2     | amrutha123@gmail.com  | 123456789  | anantapur |       2 |
|   3 | rupesh   | rupesh123   | 4321      | 3     | rupesh123@gmail.com   | 9898989898 | chennai   |       3 |
|   4 | john     | john123     | 3456      | 5     | john@gmail.com        | 9876543210 | newyork   |       4 |
+-----+----------+------------+-----------+-------+----------------------+------------+-----------+---------+
4 rows in set (0.00 sec)
```

Fig 5,3 trigger

5.4.cursor

The implicit cursors are automatically generated by Oracle while an SQL statement is executed, if you don't use an explicit cursor for the statement.

These are created by default to process the statements when DML statements like INSERT, UPDATE, DELETE etc. are executed

```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE update_price()
    -> BEGIN
    ->     DECLARE done INT DEFAULT FALSE;
    ->     DECLARE v_fid INT;
    ->     DECLARE v_price DECIMAL(10,2);
    ->
    ->     -- Declare cursor for selecting fid and price from fproduct table
    ->     DECLARE cur CURSOR FOR
    ->         SELECT fid, price FROM fproduct;
    ->
    ->     -- Declare continue handler to exit loop
    ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    ->
    ->     OPEN cur;
    ->
    ->     -- Loop through cursor
    ->     read_loop: LOOP
    ->         -- Fetch fid and price from cursor
    ->         FETCH cur INTO v_fid, v_price;
    ->
    ->         -- Check if done
    ->         IF done THEN
    ->             LEAVE read_loop;
    ->         END IF;
    ->
    ->         -- Update price by adding 100
    ->         UPDATE fproduct
    ->         SET price = price + 100
    ->         WHERE fid = v_fid;
    ->
    ->         -- Printing feedback
    ->         SELECT CONCAT('Price updated for product with fid ', v_fid);
    ->
    ->     END LOOP;
    ->
    ->     CLOSE cur;
    -> END //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL update_price();
+-----------------------------------------------------+
| CONCAT('Price updated for product with fid ', v_fid) |
+-----------------------------------------------------+
| Price updated for product with fid 1                |
+-----------------------------------------------------+
1 row in set (0.01 sec)

+-----------------------------------------------------+
| CONCAT('Price updated for product with fid ', v_fid) |
+-----------------------------------------------------+
| Price updated for product with fid 2                |
+-----------------------------------------------------+
1 row in set (0.02 sec)

+-----------------------------------------------------+
| CONCAT('Price updated for product with fid ', v_fid) |
+-----------------------------------------------------+
| Price updated for product with fid 3                |
+-----------------------------------------------------+
1 row in set (0.02 sec)

+-----------------------------------------------------+
| CONCAT('Price updated for product with fid ', v_fid) |
+-----------------------------------------------------+
| Price updated for product with fid 4                |
+-----------------------------------------------------+
1 row in set (0.03 sec)

Query OK, 0 rows affected (0.03 sec)
```

Fig 5.4 cursor table

5.12.CONSTRAINTS

```
mysql> SELECT
    ->     t.TABLE_NAME,
    ->     c.COLUMN_NAME,
    ->     tc.CONSTRAINT_NAME,
    ->     tc.CONSTRAINT_TYPE
    -> FROM
    ->     information_schema.tables t
    -> JOIN
    ->     information_schema.table_constraints tc
    -> ON
    ->     t.TABLE_SCHEMA = tc.TABLE_SCHEMA
    ->     AND t.TABLE_NAME = tc.TABLE_NAME
    -> JOIN
    ->     information_schema.columns c
    -> ON
    ->     t.TABLE_SCHEMA = c.TABLE_SCHEMA
    ->     AND t.TABLE_NAME = c.TABLE_NAME
    -> LEFT JOIN
    ->     information_schema.key_column_usage kcu
    -> ON
    ->     tc.CONSTRAINT_NAME = kcu.CONSTRAINT_NAME
    ->     AND tc.TABLE_SCHEMA = kcu.TABLE_SCHEMA
    ->     AND tc.TABLE_NAME = kcu.TABLE_NAME
    ->     AND c.COLUMN_NAME = kcu.COLUMN_NAME
    -> WHERE
    ->     t.TABLE_SCHEMA = 'agroculture';
+--------------+-------------+--------------------------------+-----------------+
| TABLE_NAME   | COLUMN_NAME | CONSTRAINT_NAME                | CONSTRAINT_TYPE |
+--------------+-------------+--------------------------------+-----------------+
| fproduct     | fid         | PRIMARY                        | PRIMARY KEY     |
| fproduct     | pid         | pid_ibfk_1                     | FOREIGN KEY     |
| transaction  | tid         | PRIMARY                        | PRIMARY KEY     |
| transaction  | bid         | PRIMARY                        | PRIMARY KEY     |
| transaction  | pid         | PRIMARY                        | PRIMARY KEY     |
| transaction  | name        | transaction_name_ibfk_2        | FOREIGN KEY     |
| blogdata     | blogId      | PRIMARY                        | PRIMARY KEY     |
| blogdata     | blogUser    | PRIMARY                        | PRIMARY KEY     |
| blogdata     | likes       | PRIMARY                        | PRIMARY KEY     |
| blogfeedback | blogId      | PRIMARY                        | PRIMARY KEY     |
| blogfeedback | comment     | PRIMARY                        | PRIMARY KEY     |
| buyer        | bid         | PRIMARY                        | PRIMARY KEY     |
| buyer        | bname       | PRIMARY                        | PRIMARY KEY     |
| buyer        | busername   | PRIMARY                        | PRIMARY KEY     |
| buyer        | bpassword   | buyer_bpassword_ibfk_3         | FOREIGN KEY     |
+--------------+-------------+--------------------------------+-----------------+
15 rows in set (0.05 sec)
```

Fig 5.5 constraints

CHAPTER 6

6A. PITFALLS IN RELATIONAL DATABASE DESIGN

6.1 Redundancy

Redundancy refers to the inclusion of extra components, elements, or data that serve as backups or duplicates to enhance reliability, fault tolerance, or performance. It's a common concept applied in various fields, including engineering, telecommunications, computer science, and information systems.

6.2 Inconsistency

Inconsistency refers to a lack of uniformity, coherence, or harmony within a system, dataset, or process. In the context of databases and information systems, inconsistency typically arises when data is incorrect, contradictory, or not synchronized across different parts of the system. It can manifest in various ways and poses significant challenges to data integrity, reliability, and usability.

6.3 Inefficiency

Inefficiency in database design stems from poor indexing, over-normalization, and inadequate query optimization. It leads to slow query performance, resource wastage, and scalability limitations. Prioritizing indexing on frequently queried columns, balancing normalization with performance, and optimizing queries can mitigate inefficiency, enhancing overall database performance.

6.4 Complexity

Complexity in database design results from over-normalization, inconsistent naming conventions, and inadequate documentation. It

leads to difficulties in understanding and maintaining the database schema, increasing the likelihood of errors and inefficiencies. Simplifying normalization, establishing clear naming conventions, and thorough documentation can alleviate complexity, facilitating easier database management and development.

6.B FUNCTIONAL DEPENDENCY AGROCULTURE

Functional dependency in a database means that if you know the value of one attribute, you can predict the value of another attribute. It's like a rule that tells us how one piece of information relates to another in a table. This concept helps organize data efficiently and avoid repeating the same information unnecessarily

6.1Functional Dependencies in the 'Buyer' Table

```
mysql> select * from buyer;
+-----+----------+-------------+-----------+-------+----------------------+-----------+-----------+---------+
| bid | bname    | busername   | bpassword | bhash | bemail               | bmobile   | baddress  | bactive |
+-----+----------+-------------+-----------+-------+----------------------+-----------+-----------+---------+
|   1 | harshini | harshini123 | 1234      | 1     | harshini123@gmail.com | 987654321 | bapatla   |       1 |
|   2 | amrutha  | amrutha123  | 5678      | 2     | amrutha123@gmail.com  | 123456789 | anantapur |       2 |
+-----+----------+-------------+-----------+-------+----------------------+-----------+-----------+---------+
```

Fig 6.1.1 buyer table

Functional Dependency:

bid -> (bname, busername, passwords, bhash, bemail, bmobile, baddress, bactive)

Generated Functional Dependencies :

•**Reflexivity:ID**→{bname,busername,bpassword,bhash,bemail,bmobile,baddress,bactive}

•**Augumentation :** ID → bName

ID → bUserName

ID → bMobileNumber

ID → bEmail

ID → bPassword

ID → baddress

6.2 Functional Dependencies in the 'Farmer' Table:

```
mysql> select * from farmer;
+-----+-------+-----------+-----------+---------+---------------+-----------+----------+---------+---------+--------+-----------+-----------+
| fid | fname | fusername | fpassword | fhash   | femail        | fmobile   | faddress | factive | frating | picExt | picStatus | review    |
+-----+-------+-----------+-----------+---------+---------------+-----------+----------+---------+---------+--------+-----------+-----------+
|   1 | rishi | rishi123  | 1234      | 12345   | rishi@gmail.com | 8600611198 | abcde    |       0 |       0 | png    |         0 | No review |
|   2 | john  | john123   | 5678      | 56789   | john@gmail.com  | 8600611199 | xyz      |       0 |       0 | jpg    |         0 | No review |
|   3 | emma  | emma123   | abcd      | abcdefgh | emma@gmail.com | 8600611200 | pqr      |       0 |       0 | jpeg   |         0 | No review |
+-----+-------+-----------+-----------+---------+---------------+-----------+----------+---------+---------+--------+-----------+-----------+
```

Fig 6.2 farmer table

Functional Dependency:

fid -> (fname, fusername, fpassword, fhash, femail, fmobile, faddress, factive, frating, picExt, picStatus, review)

Generated Functional Dependencies *:*

Reflexivity : fid -> (fname, fusername, fpassword, fhash, femail, fmobile, faddress, factive, frating, picExt, picStatus, review)

6.3 Functional Dependencies in the 'Product' Table

```
mysql> select * from fproduct;
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+--------+
| fid | pid | product | pcat      | pinfo           | price | pimage      | picStatus | review |
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+--------+
|   1 | 123 | Mango   | Fruit     | fresh fruit     |   100 | Mango3.jpeg |         8 | good   |
|   2 | 456 | apple   | Fruit     | fresh fruit     |   150 | apple.jpeg  |         0 | good   |
|   3 | 789 | tomato  | vegetable | fresh vegetable |    50 | tomato.jpeg |         5 | good   |
|   4 | 987 | onoin   | vegetable | fresh vegetable |   200 | onion.jpeg  |         7 | good   |
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+--------+
```

Fig 6.3 fproduct table

Functional Dependency:

fid -> (pid, product, pcat, pinfo, price, pimage, picStatus, review)

Generated Functional Dependencies :

Reflexivity : fid -> (pid, product, pcat, pinfo, price, pimage, picStatus, review)

6.4 Functional Dependencies in the 'Transaction' Table

```
mysql> select * from transaction;
+-----+-----+-----+----------+-----------+------------+----------------------+---------+-----------+
| tid | bid | pid | name     | city      | mobile     | email                | pincode | addr      |
+-----+-----+-----+----------+-----------+------------+----------------------+---------+-----------+
|   1 |   3 |  28 | harshini | bapatla   | 987654321  | harshini123@gmail.com | 12345   | bapatla   |
|   2 |   4 |  29 | amrutha  | chennai   | 876543219  | ammu@gmail.com       | 887766  | tamilnadu |
|   5 |   7 |  34 | hema     | hyderabad | 9833221421 | hema123@gmail.com    | 1298745 | hyd       |
+-----+-----+-----+----------+-----------+------------+----------------------+---------+-----------+
```

Functional Dependency:

tid -> (bid, pid, name, city, mobile, email, pincode, addr)

Generated Functional Dependencies :

Reflexivity : tid -> (bid, pid, name, city, mobile, email, pincode, addr)

4.5 Functional Dependencies in the 'Review' Table

```
mysql> select * from review;
+-----+----------+--------+----------+
| pid | name     | rating | comment  |
+-----+----------+--------+----------+
| 123 | harshini |      5 | good     |
| 987 | amrutha  |      4 | nice     |
| 456 | hema     |      3 | better   |
+-----+----------+--------+----------+
```

Functional Dependency:

pid -> (name, rating, comment)

Generated Functional Dependencies :

Reflexivity : pid -> (name, rating, comment)

6.5 Functional Dependencies in the 'Blogdata' Table

```
mysql> select * from blogdata;
+--------+----------+----------------+-----------------------------------------+---------------------+-------+
| blogId | blogUser | blogTitle      | blogContent                             | blogTime            | likes |
+--------+----------+----------------+-----------------------------------------+---------------------+-------+
|      1 | user1    | Blog Post      | This is the content of the second blog post. | 2024-04-18 10:30:00 |    15 |
|      2 | user2    | second Blog Post | Content for the third blog post.      | 2024-04-18 11:45:00 |     8 |
|      3 | user3    | third Blog Post | Content for the fourth blog post.      | 2024-04-18 13:20:00 |    20 |
+--------+----------+----------------+-----------------------------------------+---------------------+-------+
```

Functional Dependency:

blogId -> {blogUser, blogTitle, blogContent, blogTime, likes}

Generated Functional Dependencies :

Reflexivity : blogId -> {blogUser, blogTitle, blogContent, blogTime, likes}

6.6 Functional Dependencies in the 'blogfeedback' Table

```
mysql> select * from blogfeedback;
+--------+--------------------------------+-------------+---------------+---------------------+
| blogId | comment                        | commentUser | commentPic    | commentTime         |
+--------+--------------------------------+-------------+---------------+---------------------+
|      1 | Great post!                    | user5       | user5_pic.jpg | 2024-04-18 10:45:00 |
|      1 | I really enjoyed reading this. | user6       | user6_pic.jpg | 2024-04-18 11:15:00 |
|      2 | Interesting perspective.       | user7       | user7_pic.jpg | 2024-04-18 12:00:00 |
+--------+--------------------------------+-------------+---------------+---------------------+
```

Functional Dependency:

blogId -> {comment, commentUser, commentPic, commentTime}

Generated Functional Dependencies :

Reflexivity: blogId -> {comment, commentUser, commentPic, commentTime}

6.7 Functional Dependencies in the 'mycart' Table

```
mysql> select * from mycart;
+-----+-----+
| bid | pid |
+-----+-----+
|   3 |  27 |
|   3 |  30 |
+-----+-----+
```

Functional Dependency:

bid -> pid

Each value of bid uniquely determines the corresponding value of pid

Generated Functional Dependencies :

Reflexivity : bid -> pid

6.C NORMALIZATION:

Normalization is a database design technique used to organize the attributes and tables of a relational database to minimize redundancy and dependency. It involves breaking down large tables into smaller ones and defining relationships between them. The normalization process typically involves several normal forms, with each normal form addressing specific types of anomalies that can occur in a database.

TYPES OF NORMALIZATION:

- **First Normal Form** :

  Atomicity of Values

- **Second Normal Form** (2NF):

  No Partial Dependencies

- **Third Normal Form** (3NF):

  No Transitive Dependencies
- **Boyce-Codd Normal Form** (BCNF):

  Every Determinant a Candidate Key
- **Fourth Normal Form** (4NF):

  Multi-valued Dependencies
- **Fifth Normal Form** (5NF):

  Join Dependencies

6.1 FIRST NORMAL FORM (1NF)

- Each cell in a table should hold a single, indivisible value.
- Each column in a table must have a unique name, and the order of columns should not matter.
- Avoid storing multiple values in a single field. Each field should represent a single piece of data.
- The order in which data is stored should not impact its interpretation or retrieval.

- **'farmer' Table –** Rules not Violated
- **'buyer' Table -** Rules not Violated
- **'fproduct' Table -** Rules not Violated

- **‘transaction’ Table -** Rules not Violated
- **‘blogdata’ Table -** Rules not Violated
- **‘blogfeedback’ Table -** Rules not Violated
- **‘review’ Table -** Rules not Violated
- **‘mycart’ Table -** Rules not Violated
- **‘likedata’ Table –** Rules not Violated

6.2 SECOND NORMAL FORM (2NF)

- The table must already be in First Normal Form (1NF), meaning it satisfies the criteria of atomic values in each cell.
- Each non-prime attribute (attribute not part of any candidatekey) must be fully functionally dependent on the entire primary key, ensuring that no attribute is dependent on only a subset of the primary key. This eliminates partial dependencies, where part of the primary key determines some attributes' values independently.
- **‘farmer’ Table –** Rules not Violated
- **‘buyer’ Table -** Rules not Violated
- **‘fproduct’ Table -** Rules not Violated
- **‘transaction’ Table -** Rules not Violated
- **‘blogdata’ Table -** Rules not Violated

- **‘blogfeedback’ Table -** Rules not Violated
- **‘review’ Table -** Rules not Violated

- **‘mycart’ Table -** Rules not Violated
- **‘likedata’ Table –** Rules not Violated

6.3 THIRD NORMAL FORM (3NF)

- The table must already satisfy the criteria of Second Normal Form (2NF).
- There should be no transitive dependencies, meaning that no non-prime attribute should depend on another non-prime attribute. All non-prime attributes must depend only on the primary key.

- **TRANSACTION TABLE:** This violates 3NF because non-key attributes (Name, City, MobileNumber, Email, Pincode,Address) depend on a non-super key attribute (bid). In 3NF, we remove transitive dependencies.

  Attributes city and addr are transitively dependent on pincode. We'll remove this dependency by creating a new table

INPUT TABLE

```
mysql> select * from transaction;
+-----+-----+-----+----------+-----------+------------+----------------------+---------+-----------+
| tid | bid | pid | name     | city      | mobile     | email                | pincode | addr      |
+-----+-----+-----+----------+-----------+------------+----------------------+---------+-----------+
|   1 |   3 |  28 | harshini | bapatla   | 987654321  | harshini123@gmail.com | 12345   | bapatla   |
|   2 |   4 |  29 | amrutha  | chennai   | 876543219  | ammu@gmail.com        | 887766  | tamilnadu |
|   5 |   7 |  34 | hema     | hyderabad | 9833221421 | hema123@gmail.com     | 1298745 | hyd       |
+-----+-----+-----+----------+-----------+------------+----------------------+---------+-----------+
```

DECOMPOSED TABLES

```
mysql> select * from transactiondetails;
+-----+-----+-----+----------+------------+----------------------+---------+
| tid | bid | pid | name     | mobile     | email                | pincode |
+-----+-----+-----+----------+------------+----------------------+---------+
| 1   | 3   | 28  | harshini | 987654321  | harshini123@gmail.com | 12345   |
| 2   | 4   | 29  | amrutha  | 876543219  | ammu@gmail.com        | 887766  |
| 5   | 7   | 34  | hema     | 9833221421 | hema123@gmail.com     | 1298745 |
+-----+-----+-----+----------+------------+----------------------+---------+

mysql> select * from transaction;
+---------+---------+-----------+---------+
| pincode | city    | addr      | tid     |
+---------+---------+-----------+---------+
| 12345   | bapatla | bapatla   | 1       |
| 887766  | chennai | tamilnadu | 2       |
| 1298745 | hyd     | hyd       | 5       |
+---------+---------+-----------+---------+
```

Fig 6.3.1 transaction table

- **FPRODUCT TABLE**: To achieve 3NF, we need to remove transitive dependencies. Let's identify the functional dependencies:

  {pid} -> {product, pcat, pinfo, price, pimage, picStatus, review}

  {pcat} -> {pinfo}

INPUT TABLE

```
mysql> select * from fproduct;
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+--------+
| fid | pid | product | pcat      | pinfo           | price | pimage      | picStatus | review |
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+--------+
|   1 | 123 | grapes  | Fruit     | fresh fruit     |   100 | Mango3.jpeg |         1 | good   |
|   2 | 456 | apple   | Fruit     | fresh fruit     |   150 | apple.jpeg  |         2 | good   |
|   3 | 789 | tomato  | vegetable | fresh vegetable |    50 | tomato.jpeg |         3 | good   |
|   4 | 987 | onoin   | vegetable | fresh vegetable |   200 | onion.jpeg  |         4 | good   |
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+--------+
```

DECOMPOSED TABLES

```
mysql> select * from fproductdetails;
+-----+---------+-----------+-----------------+-------+
| pid | product | pcat      | pinfo           | price |
+-----+---------+-----------+-----------------+-------+
| 123 | Mango   | Fruit     | fresh fruit     |   100 |
| 456 | apple   | Fruit     | fresh fruit     |   150 |
| 789 | tomato  | vegetable | fresh vegetable |    50 |
| 987 | onoin   | vegetable | fresh vegetable |   200 |
+-----+---------+-----------+-----------------+-------+

mysql> select * from fproduct;
+-----+-------------+-----------+--------+
| pid | pimage      | picStatus | review |
+-----+-------------+-----------+--------+
| 123 | Mango3.jpeg |         8 | good   |
| 456 | apple.jpeg  |         0 | good   |
| 789 | tomato.jpeg |         5 | good   |
| 987 | onion.jpeg  |         7 | good   |
+-----+-------------+-----------+--------+
```

Fig 6.3.2 fproduct table

- **'buyer' Table -** Rules not Violated
- **'farmer' Table -** Rules not Violated
- **'review' Table -** Rules not Violated
- **'blogfeedback' Table -** Rules not Violated
- **'blogdata' Table -** Rules not Violated
- **'mycart' Table -** Rules not Violated
- **'likedata' Table –** Rules not Violated

6.4 BOYCE CODD NORMAL FORM (BCNF)

- The table must already satisfy the criteria of Third Normal Form (3NF).
- Every non-trivial functional dependency must be a dependency on a superkey, meaning that if A determines B, then A must be a superkey. This ensures that there are no non-trivial dependencies on attributes that are not part of any candidate key.

➢ **'buyer' Table –** Rules not Violated

➢ **'farmer' Table -** Rules not Violated

➢ **'fproduct' Table -** Rules not Violated

➢ **'review' Table -** Rules not Violated

➢ **'blogdata' Table -** Rules not Violated

➢ **'blogfeedback' Table -** Rules not Violated

➢ **'transaction' Table -** Rules not Violated

➢ **'mycart' Table -** Rules not Violated

➢ **'likedata' Table –** Rules not Violated

6.5 FOURTH NORMAL FORM (4NF)

- The table must already satisfy the criteria of Boyce-Codd Normal Form (BCNF).
- There should be no non-trivial multivalued dependencies between attributes. This means that the values in one set of

attributes should not determine the values in another set of attributes independently of the primary key.

- **'buyer' Table –** Rules not Violated
- **'farmer' Table -** Rules not Violated
- **'fproduct' Table -** Rules not Violated
- **'review' Table -** Rules not Violated
- **'transaction' Table -** Rules not Violated
- **'blogdata' Table -** Rules not Violated
- **'blogfeedback' Table -** Rules not Violated
- **'mycart' Table -** Rules not Violated
- **'likedata' Table –** Rules not Violated

6.6 FIFTH NORMAL FORM (5NF)

- The table must already satisfy the criteria of Fourth Normal Form (4NF).
- All join dependencies are satisfied, meaning that every decomposition into smaller tables preserves certain implied relationships, ensuring that no redundancies or anomalies occur when joining these tables back together.

- **'buyer' Table –** Rules not Violated
- **'farmer' Table -** Rules not Violated

- **‘fproduct’ Table -** Rules not Violated
- **‘review’ Table -** Rules not Violated
- **‘transaction’ Table -** Rules not Violated
- **‘blogdata’ Table -** Rules not Violated
- **‘blogfeedback’ Table -** Rules not Violated
- **‘mycart’ Table -** Rules not Violated
- **‘likedata’ Table –** Rules not Violated

6.7 NORMALIZED SCHEMA OF AGROCULTURE:

```
mysql> DESC FARMER;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| fid       | int          | NO   | PRI | NULL    | auto_increment |
| fname     | varchar(255) | NO   |     | NULL    |                |
| fusername | varchar(255) | NO   |     | NULL    |                |
| fpassword | varchar(255) | NO   |     | NULL    |                |
| fhash     | varchar(255) | NO   |     | NULL    |                |
| femail    | varchar(255) | NO   |     | NULL    |                |
| fmobile   | varchar(255) | NO   |     | NULL    |                |
| faddress  | text         | NO   |     | NULL    |                |
| factive   | int          | NO   |     | 0       |                |
| frating   | int          | NO   |     | 0       |                |
| picExt    | varchar(255) | NO   |     | png     |                |
| picStatus | int          | NO   |     | 0       |                |
+-----------+--------------+------+-----+---------+----------------+
```

6.71 FIG desc farmer

```
mysql> DESC BUYER;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| bid       | int          | NO   | PRI | NULL    | auto_increment |
| bname     | varchar(100) | NO   |     | NULL    |                |
| busername | varchar(100) | NO   |     | NULL    |                |
| bpassword | varchar(100) | NO   |     | NULL    |                |
| bhash     | varchar(100) | NO   |     | NULL    |                |
| bemail    | varchar(100) | NO   |     | NULL    |                |
| bmobile   | varchar(100) | NO   |     | NULL    |                |
| baddress  | text         | NO   |     | NULL    |                |
| bactive   | int          | NO   |     | 0       |                |
+-----------+--------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
```

```
mysql> DESC fproduct;
+-----------+--------------+------+-----+-----------+----------------+
| Field     | Type         | Null | Key | Default   | Extra          |
+-----------+--------------+------+-----+-----------+----------------+
| fid       | int          | NO   | PRI | NULL      | auto_increment |
| pid       | int          | NO   |     | NULL      |                |
| product   | varchar(255) | NO   |     | NULL      |                |
| pcat      | varchar(255) | NO   |     | NULL      |                |
| pinfo     | varchar(255) | NO   |     | NULL      |                |
| price     | float        | NO   |     | NULL      |                |
| pimage    | varchar(255) | NO   |     | blank.png |                |
| picStatus | int          | NO   |     | 0         |                |
+-----------+--------------+------+-----+-----------+----------------+
```

```
mysql> DESC transaction;
+---------+--------------+------+-----+---------+----------------+
| Field   | Type         | Null | Key | Default | Extra          |
+---------+--------------+------+-----+---------+----------------+
| tid     | int          | NO   | PRI | NULL    | auto_increment |
| bid     | int          | NO   |     | NULL    |                |
| pid     | int          | NO   |     | NULL    |                |
| name    | varchar(255) | NO   |     | NULL    |                |
| city    | varchar(255) | NO   |     | NULL    |                |
| mobile  | varchar(255) | NO   |     | NULL    |                |
| email   | varchar(255) | NO   |     | NULL    |                |
| pincode | varchar(255) | NO   |     | NULL    |                |
| addr    | varchar(255) | NO   |     | NULL    |                |
+---------+--------------+------+-----+---------+----------------+
```

Fig 6.7.2 transaction

```
+-------------+-------------+------+-----+-------------------+-------------------+
| Field       | Type        | Null | Key | Default           | Extra             |
+-------------+-------------+------+-----+-------------------+-------------------+
| blogId      | int         | NO   | PRI | NULL              | auto_increment    |
| blogUser    | varchar(256)| NO   |     | NULL              |                   |
| blogTitle   | varchar(256)| NO   |     | NULL              |                   |
| blogContent | longtext    | NO   |     | NULL              |                   |
| blogTime    | timestamp   | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| likes       | int         | NO   |     | 0                 |                   |
+-------------+-------------+------+-----+-------------------+-------------------+
```

```
mysql> desc blogfeedback;
+-------------+-------------+------+-----+-------------------+-------------------+
| Field       | Type        | Null | Key | Default           | Extra             |
+-------------+-------------+------+-----+-------------------+-------------------+
| blogId      | int         | NO   | PRI | NULL              | auto_increment    |
| comment     | varchar(256)| NO   |     | NULL              |                   |
| commentUser | varchar(256)| NO   |     | NULL              |                   |
| commentPic  | varchar(256)| NO   |     | profile0.png      |                   |
| commentTime | timestamp   | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-------------+-------------+------+-----+-------------------+-------------------+
```

```
mysql> desc review;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| pid     | int          | NO   |     | NULL    |       |
| name    | varchar(255) | NO   |     | NULL    |       |
| rating  | int          | NO   |     | NULL    |       |
| comment | text         | NO   |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> desc mycart;
+-------+------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+-------+------+------+-----+---------+-------+
| bid   | int  | NO   |     | NULL    |       |
| pid   | int  | NO   |     | NULL    |       |
+-------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> desc likedata;
+------------+------+------+-----+---------+-------+
| Field      | Type | Null | Key | Default | Extra |
+------------+------+------+-----+---------+-------+
| blogId     | int  | NO   |     | NULL    |       |
| blogUserId | int  | NO   |     | NULL    |       |
+------------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Fig 6.73 desc review

CHAPTER 7

7.1 Implementation of concurrency control and recovery mechanism

7.1 TRANSACTION CONTROL

A transaction is a unit of program execution that accesses and possibly updates various data items. Transaction is a single operation of processing that can have many operations. Transaction is needed when more than one user wants to access same database. Transaction has ACID properties.

Atomicity: Either all operations of the transaction are properly reflected in the database or none are.

Consistency: Execution of a transaction in isolation preserves the consistency of the database.

Isolation: Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions.

Durability: After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

7.1 Agroculture with review and beforeupdates

```
mysql> select * from fproduct;
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+
| fid | pid | product | pcat      | pinfo           | price | pimage      | picStatus |
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+
|   1 | 123 | Mango   | Fruit     | fresh fruit     |   100 | Mango3.jpeg |         1 |
|   2 | 456 | apple   | Fruit     | fresh fruit     |   150 | apple.jpeg  |         2 |
|   3 | 789 | tomato  | vegetable | fresh vegetable |    50 | tomato.jpeg |         3 |
|   4 | 987 | onion   | vegetable | fresh vegetable |   200 | onion.jpeg  |         4 |
+-----+-----+---------+-----------+-----------------+-------+-------------+-----------+
4 rows in set (0.00 sec)

mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> savepoint beforeupdates;
Query OK, 0 rows affected (0.00 sec)

mysql> update fproduct
    -> set review='nice'
    -> where fid=3;
ERROR 1054 (42S22): Unknown column 'review' in 'field list'
mysql> update fproduct
    -> set product='grapes'
    -> where fid=1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> rollback to beforeupdates;
Query OK, 0 rows affected (0.01 sec)
mysql> ROLLBACK TO start_event;
Query OK, 0 rows affected (0.01 sec)
```

FIG 7.1 Transaction command

7.2 Purchase Equipment Procedure:

This stored procedure facilitates the price of product. It starts a transaction, inserts a new record into the Equipment table with provided details, and creates a savepoint. It then checks if the total price of equipment purchased exceeds 3000. If so, it rolls back to the savepoint and displays a budget exceeded message; otherwise, it commits the transaction, confirming successful equipment purchase.

```
mysql> DELIMITER //

mysql> CREATE PROCEDURE PRODUCTprice(
    ->fid,
    ->pid,
    ->product,
    ->pcat,
    ->pinfo,
    ->price,
    ->pimage,
    ->picStatus
    -> )
    ->BEGIN
    ->START TRANSACTION;
    ->INSERT INTO Equipment (fid, pid, product, pcat, Price, pimage, pimage, PicStatus)
             VALUES (4,987, onion, vegetable, 200, price, onion.jpg, 4);
    ->SAVEPOINT product_purchased;
    -> IF (
    ->       SELECT SUM(price)
    ->       FROM Fproduct
    ->       WHERE pid=fid
    ->     ) >10000 THEN
    ->ROLLBACK TO product_purchased;
              SELECT 'price exceeded. Rollback performed.' AS Message;
    ->ELSE
    ->COMMIT;
    ->         SELECT 'Product purchased Succesfully.' AS Message;
    ->      END IF;
    -> END //
Query OK, 0 rows affected (0.03 sec)
mysql> DELIMITER ;
mysql> CALL PRODUCTprice(1, 123, apple, fruit, 100,apple.jpg,1);
+----------------------------------------+
| Message                                |
+----------------------------------------+
| price exceeded. Rollback performed     |
+----------------------------------------+
1 row in set (0.02 sec)
```

FIG 7.2 transaction command

7.3 updating city name :

This transaction handles the updating city name for a buyer, including updating their pincode and recording pincode information directly in the transaction table.

```
mysql> START TRANSACTION;

Query OK, 0 rows affected (0.00 sec)

mysql> SAVEPOINT transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE transaction
-> SET = 'city'= chennai
-> WHERE equipment_id = 1;

Query OK, 1 row affected (0.00 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE transaction

-> SET pincode = '603203'

-> WHERE equipment_id = 1;

Query OK, 1 row affected (0.00 sec)

Rows matched: 1 Changed: 1 Warnings: 0

+-----+-----+-----+---------+-----------+------------+-----------------------+---------+-----------+
| tid | bid | pid | name    | city      | mobile     | email                 | pincode | addr      |
+-----+-----+-----+---------+-----------+------------+-----------------------+---------+-----------+
|   1 |   3 |  28 | harshini | chennai   | 987654321  | harshini123@gmail.com | 603203  | bapatla   |
|   2 |   4 |  29 | amrutha  | chennai   | 876543219  | ammu@gmail.com        | 887766  | tamilnadu |
|   5 |   7 |  34 | hema     | hyderabad | 9833221421 | hema123@gmail.com     | 1298745 | hyd       |
+-----+-----+-----+---------+-----------+------------+-----------------------+---------+-----------+

mysql> ROLLBACK TO Transaction;
Query OK, 0 row affected (0.00 sec)
```

FIG 7.3 UPDATING CITY NAME

CHAPTER 8

8.CODE for project

HTML CODE(profile page)

```
<!DOCTYPE html>
   <html >
    <head>
      <title>AgroCulture</title>
      <meta charset="utf-8" />
      <meta name="viewport" content="width=device-width, initial-scale=1"
/>
      <link href="../bootstrap\css\bootstrap.min.css" rel="stylesheet">
            <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></sc
ript>
      <script src="../bootstrap\js\bootstrap.min.js"></script>
      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
            <meta name="description" content="" />
            <meta name="keywords" content="" />
            <!--[if lte IE 8]><script
src="css/ie/html5shiv.js"></script><![endif]-->
            <script src="../js/jquery.min.js"></script>
            <script src="../js/skel.min.js"></script>
            <script src="../js/skel-layers.min.js"></script>
            <script src="../js/init.js"></script>
            <link rel="stylesheet" href="../css/skel.css" />
            <link rel="stylesheet" href="../css/style.css" />
            <link rel="stylesheet" href="../css/style-xlarge.css" />
   </head>

   <body>
```

```
<?php
   require 'menu.php';
?>

<section id="banner" class="wrapper">
   <div class="container">
      <header class="major">
         <h2>Welcome</h2>
      </header>
      <p>
      <?php
         if ( isset($_SESSION['message']) )
         {
            echo $_SESSION['message'];
            unset( $_SESSION['message'] );
         }
      ?>
      </p>

      <?php
         if ( !$active )
         {
            echo
            "<div>
               Account is not verified! Please confirm your email by
clicking
               on the email link!
            </div>";
         }
      ?>
       <h2><?php echo $name; ?></h2>
       <p><?= $email ?></p>

       <?php if($_SESSION['Category'] == 1): ?>
         <div class="row uniform">
```

```
                <div class="6u 12u$(xsmall)">
                    <a href=../profileView.php class="button special">My Profile</a>
                </div>
                <div class="6u 12u$(xsmall)">
                    <a href="logout.php" class="button special">LOG OUT</a>
                </div>
            </div>
        <?php else: ?>
            <div class="row uniform">
                <div class="6u 12u$(xsmall)">
                    <a href=../market.php class="button special">Digital Market</a>
                </div>
                <div class="6u 12u$(xsmall)">
                    <a href="logout.php" class="button special">LOG OUT</a>
                </div>
            </div>

        <?php endif; ?>


    </body>
</html>
```

CSS CODE(PROFILE)

```
input[type=text], input[type=password]
{
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
```

```
    display: inline-block;
    border: 1px solid #ccc;
    box-sizing: border-box;
    position: relative;
}

/* Set a style for all buttons */
button {
    background-color: #4CAF50;
    color: white;
    padding: 14px 20px;
    margin: 8px 0;
    border: none;
    cursor: pointer;
    width: 60%;
    position: relative;
}

button:hover {
    opacity: 0.8;
}

/* Extra styles for the cancel button */
.cancelbtn {
    width: auto;
    padding: 10px 18px;
    background-color: #f44336;
}

/* Center the image and position the close button */
.imgcontainer {
    text-align: center;
    margin: 24px 0 12px 0;
    position: relative;
}

img.avatar {
    width: 40%;
    border-radius: 50%;
}
```

```
.container {
   padding: 16px;
}

span.psw {
   float: right;
   padding-top: 16px;
}

/* The Modal (background) */
.modal {
   display: none; /* Hidden by default */
   position: fixed; /* Stay in place */
   z-index: 1; /* Sit on top */
   left: 0;
   top: 0;
   width: 100%; /* Full width */
   height: 100%; /* Full height */
   overflow: auto; /* Enable scroll if needed */
   background-color: rgb(0,0,0); /* Fallback color */
   background-color: rgba(0,0,0,0.4); /* Black w/ opacity */
   padding-top: 60px;
}

/* Modal Content/Box */
.modal-content {
   background-color: #fefefe;
   margin: 5% auto 15% auto; /* 5% from the top, 15% from the bottom and
centered */
   border: 1px solid #888;
   width: 50%; /* Could be more or less, depending on screen size */
}

/* The Close Button (x) */
.close {
   position: absolute;
   right: 25px;
   top: 0;
   color: #000;
```

```
  font-size: 35px;
  font-weight: bold;
}

.close:hover,
.close:focus {
  color: red;
  cursor: pointer;
}

/* Add Zoom Animation */
.animate {
  -webkit-animation: animatezoom 0.6s;
  animation: animatezoom 0.6s
}

@-webkit-keyframes animatezoom {
  from {-webkit-transform: scale(0)}
  to {-webkit-transform: scale(1)}
}

@keyframes animatezoom {
  from {transform: scale(0)}
  to {transform: scale(1)}
}

/* Change styles for span and cancel button on extra small screens */
@media screen and (max-width: 300px) {
  span.psw {
    display: block;
    float: none;
  }
  .cancelbtn {
    width: 100%;
  }
}
```

9.Output of the code(RESULT)

FIG 9.1 LOGIN PAGE

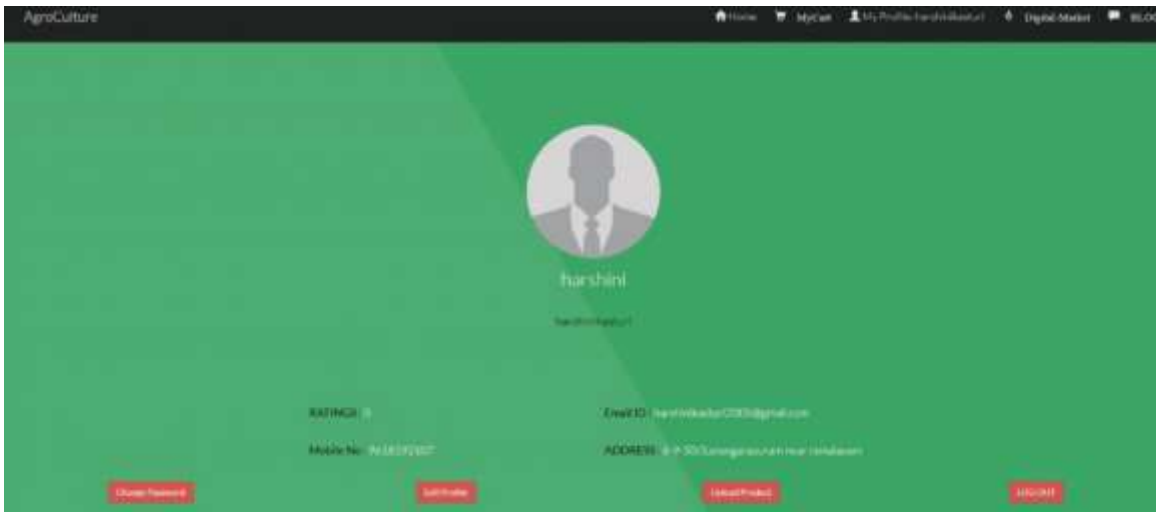

FIG 9.2 PROFILE PAGE

Home: This is likely the main page or landing page of the AgroCulture platform.
MyCart: This feature allows users to add products to a virtual shopping cart and proceed to checkout.
My Profile: harshinikasturi: This is a personalized area for a user named "harshinikasturi" to manage their account settings, view order history, and more.
Digital-Market: This may be a section of the platform where users can buy and sell agricultural products or related services.
BLOG: This is likely a section of the website where AgroCulture publishes articles, news, and updates related to agriculture.
LOGIN: This feature allows users to access their accounts by entering their credentials.
REGISTER: This feature enables new users to create an account on the AgroCulture platform.
Your Product Our Market: This tagline suggests that AgroCulture is a marketplace where users can sell their agricultural products.

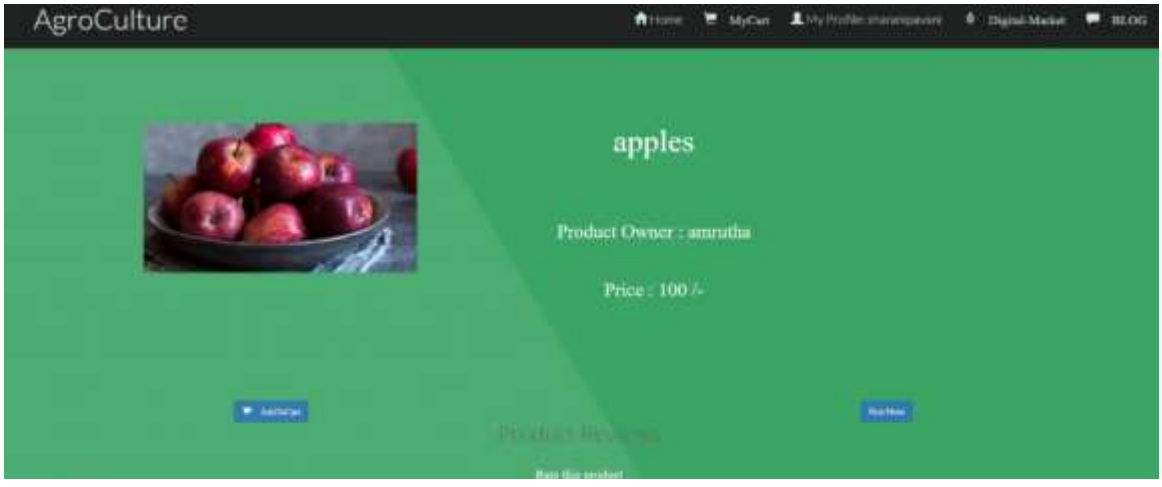

FIG 9.3 MYCART

FIG 9.4 TRANSACTION DETAILS

This Agroculture should offer features such as real-time inventory tracking, demand forecasting, inventory optimization, and intuitive reporting tools. By addressing these challenges, the proposed Agroculture aims to empower farmers and buyers to achieve greater operational efficiency, reduce costs, enhance transactional experiences, and gain a competitive advantage in the online agricultural marketplace.

10.ONLINE COURSE CERTIFICATION

Name : G.N.R.S.S Charan REDDY
Register Number : RA2211003011287

Name : Mallela Gnanamrutha
Register Number : RA2211003011294

Name : Kasturi Harshini
Register Number : RA2211003011299

11.CONCLUSION

The development and implementation of an online AgroCulture application utilizing MySQL as its backend database system represents a significant leap forward in modernizing agricultural practices. This innovative platform harnesses the power of web technology and database management to bridge the gap between farmers and buyers, facilitating seamless transactions and communication within the agricultural supply chain.

One of the paramount advantages of the online AgroCulture app lies in its ability to streamline inventory management processes. By leveraging MySQL's robust data storage and retrieval capabilities, farmers can efficiently track and manage their inventory of crops, seeds, and other agricultural inputs. This ensures optimal utilization of resources and minimizes wastage, contributing to improved productivity and profitability in agriculture.

Moreover, the app's comprehensive order management functionalities empower buyers to easily browse and purchase fresh produce directly from farmers. MySQL facilitates secure and reliable transaction processing, ensuring that orders are accurately captured, processed, and fulfilled in a timely manner. This seamless integration of inventory and order management enhances the overall efficiency of agricultural commerce.

The utilization of MySQL in the online AgroCulture app also enables advanced inventory analysis capabilities. Farmers can leverage MySQL's data analytics features to gain valuable insights into sales trends, demand patterns, and stock levels. This data-driven

approach empowers farmers to make informed decisions, optimize inventory levels, and adapt to changing market conditions more effectively.

Furthermore, the scalability and security features offered by MySQL ensure that the online AgroCulture app can accommodate the evolving needs of the agricultural industry. As the platform grows and expands to serve a larger user base, MySQL's scalability features allow for seamless scaling of resources to meet increasing demand. Additionally, MySQL's robust security mechanisms safeguard sensitive agricultural data, protecting it from unauthorized access and ensuring the integrity and confidentiality of transactions.

In conclusion, the online AgroCulture application, powered by MySQL, represents a transformative tool for revolutionizing agricultural commerce. By facilitating efficient inventory management, seamless order processing, and advanced data analytics, this platform empowers farmers and buyers alike to thrive in an increasingly digital agricultural ecosystem, driving growth, sustainability, and innovation in the agricultural sector.

12.REFERENCES

1. MySQL Documentation: https://dev.mysql.com/doc/

2. Database System Concepts, 6th Edition, Abraham Silberschatz, Henry F. Korth, S. Sudarshan.

3. Java Swing Documentation: https://docs.oracle.com/javase/tutorial/uiswing/

4. Feuerstein, S.; Pribyl, B. Oracle PL/SQL Programming, 6th ed.; O'Reilly Media: Sebastopol, CA, USA, 2016. [Google Scholar]

5.Repository Annotation Library. Available online: https://www.org.springframework (accessed on 20 February 2021)