**DBMS Project on Indian Agricultural System**

The **Indian Agricultural System** database management project aims to create a well-structured database to manage and organize data related to farmers, their lands, the crops they grow, and related agricultural resources such as fertilizers, pesticides, and irrigation systems. The goal of the project is to track key agricultural information, such as which farmers own which lands, what crops are cultivated in different regions, and the market prices of crops, to improve decision-making, agricultural planning, and resource management.

The project utilizes core SQL concepts like Data Definition Language (DDL), Data Manipulation Language (DML), Transaction Control Language (TCL), and View Definition Language (VDL) to design, manage, and query the database.

**Project Goals:**

1. **Efficient Data Management**: To create a system that efficiently stores, retrieves, and manages data about farmers, lands, crops, fertilizers, and other agricultural components.
2. **Enhanced Decision-Making**: To enable government agencies, agricultural planners, and farmers to make data-driven decisions regarding crop selection, land usage, and resource allocation.
3. **Improved Resource Planning**: To provide insights into crop yield, land distribution, irrigation systems, and market prices, improving agricultural productivity and sustainability.
4. **Simplified Reporting**: To generate reports and views that provide an easy-to-understand summary of the data for analysis, such as total land owned by each farmer or market trends for crops.

**Database Schema:**

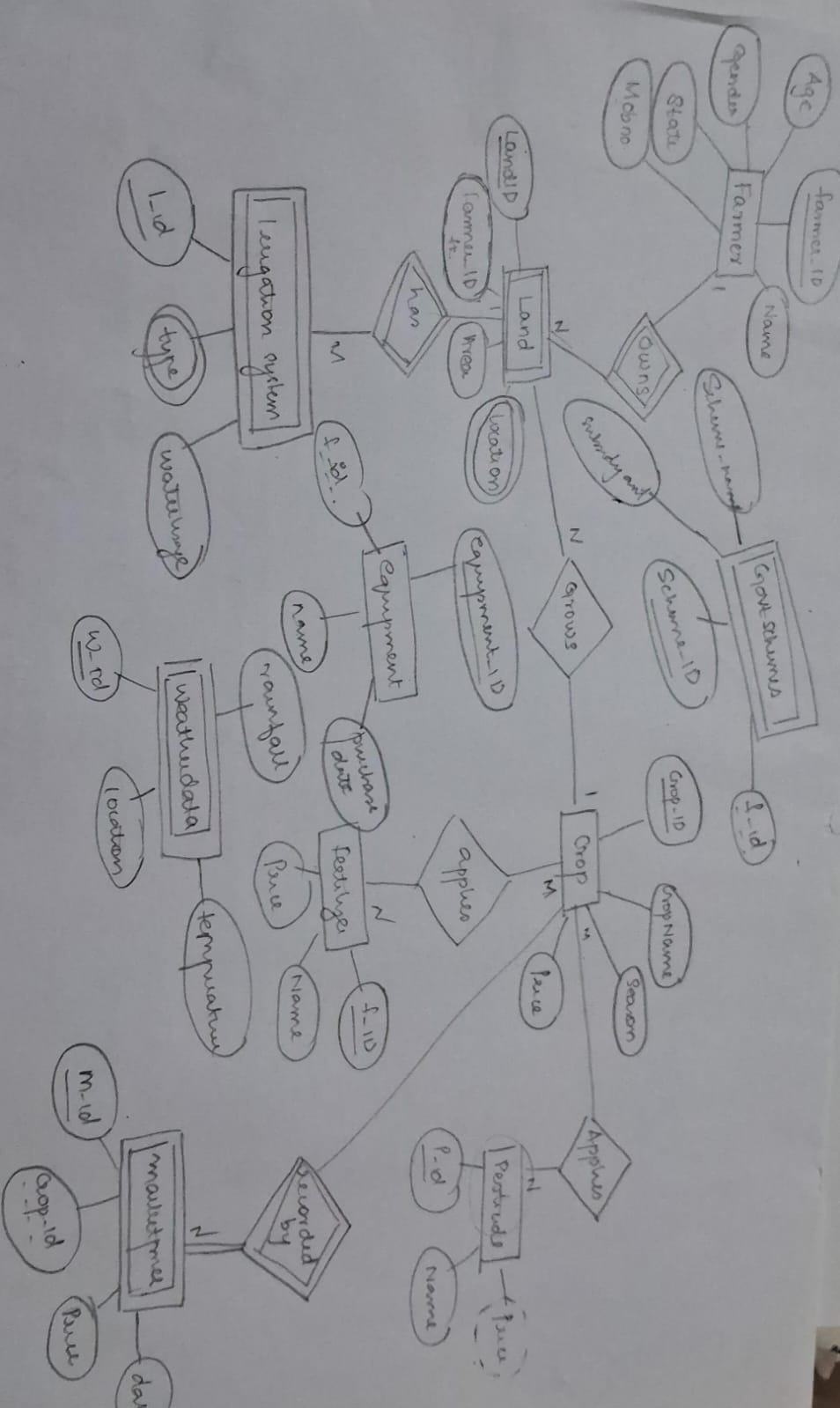
The database schema is designed based on the **Entity-Relationship (ER) Diagram**, which defines the relationships between key entities in the agricultural system. These entities include:

1. **Farmers**: The individuals who own and manage agricultural lands. Key attributes include FarmerID, Name, Age, Gender, and State.
2. **Lands**: The plots of land owned by farmers where crops are cultivated. Attributes include LandID, FarmerID (foreign key), Area, and Location.
3. **Crops**: The crops grown on the lands, including attributes such as CropID, CropName, Season, and PricePerKg.
4. **Fertilizers**: Details of fertilizers used for specific crops. Attributes include FertilizerID, Name, and PricePerKg.
5. **Pesticides**: Information on pesticides applied to crops, including PesticideID, Name, and PricePerLitre.
6. **MarketPrices**: Records of the market prices of crops at different times. Attributes include MarketID, CropID, PricePerKg, and Date.
7. **IrrigationSystems**: Data about the irrigation systems installed on various lands. Attributes include IrrigationID, LandID, Type, and WaterUsage.
8. **Equipment**: Agricultural equipment owned by farmers, including EquipmentID, FarmerID, Name, and PurchaseDate.
9. **WeatherData**: Records of weather conditions such as temperature and rainfall in various locations. Attributes include WeatherID, Location, Temperature, and Date.
10. **GovtSchemes**: Government schemes and subsidies availed by farmers. Attributes include SchemeID, FarmerID, SchemeName, and SubsidyAmount.

**Relationships:**

* A **Farmer** can own multiple **Lands** (1
* relationship).
* Each **Land** can grow multiple **Crops** (N:1 relationship).
* **Crops** can be associated with various **Fertilizers** and **Pesticides**.
* **Irrigation Systems** are installed on **Lands**.
* Market prices are recorded for **Crops** at different times and locations.
* The database schema efficiently organizes the data, ensuring proper relationships between different entities, and allows complex queries for analysis and reporting. The schema is implemented using **SQL** and covers all essential SQL concepts including DDL, DML, TCL, and VDL commands to build, manipulate, and control the database.

**ER Diagram**

****

**SQL Commands Explanation:**

* **DDL (Data Definition Language):**

DDL commands are used to define, modify, and manage the structure of a database. The most common DDL commands include CREATE, ALTER, and DROP.

1. CREATE Command

The CREATE command is used to create a new table in the database.

2. ALTER Command

The ALTER command is used to modify the structure of an existing table. You can add, delete, or modify columns.

3. DROP Command

The DROP command is used to delete an entire table from the database.

Summary

* CREATE: Defines a new table structure.
* ALTER: Modifies an existing table structure.
* DROP: Deletes a table entirely.

These commands are crucial for setting up and maintaining the database schema in any relational database management system (RDBMS).

**2. Transaction Control Language (TCL)**

TCL is used to manage transactions in a database. The main TCL commands include:

* BEGIN: Starts a transaction, allowing for a series of operations to be treated as a single unit.
* COMMIT: Saves all changes made during the current transaction, making them permanent.
* ROLLBACK: Undoes all changes made during the current transaction, reverting to the last committed state.
* SAVEPOINT: Sets a point within a transaction that allows for partial rollbacks, enabling the user to roll back to a specific state without affecting other operations.

**3. View Definition Language (VDL)**

VDL is used for defining and managing views in a database. A view is a virtual table based on the result of a SELECT query. Key aspects include:

* Creating Views: Allows users to define a view that simplifies complex queries or presents data in a specific format.
* Modifying Views: Users can alter the definition of an existing view to reflect changes in underlying tables or desired output.

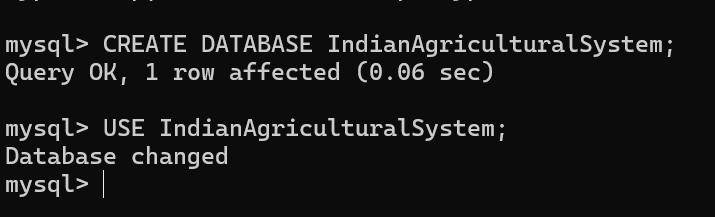
**4. Join Operations**

Join operations are used to combine rows from two or more tables based on related columns. The main types of joins include:

* INNER JOIN: Returns only the rows with matching values in both tables, excluding non-matching rows.
* LEFT JOIN (or LEFT OUTER JOIN): Returns all rows from the left table and the matched rows from the right table; non-matching rows from the right table will return NULL.
* RIGHT JOIN (or RIGHT OUTER JOIN): Returns all rows from the right table and the matched rows from the left table; non-matching rows from the left table will return NULL.
* FULL OUTER JOIN: Combines the results of both LEFT and RIGHT JOINs, returning all rows from both tables and filling in NULLs for non-matching rows.
* CROSS JOIN: Returns the Cartesian product of both tables, meaning it combines every row from the first table with every row from the second table.

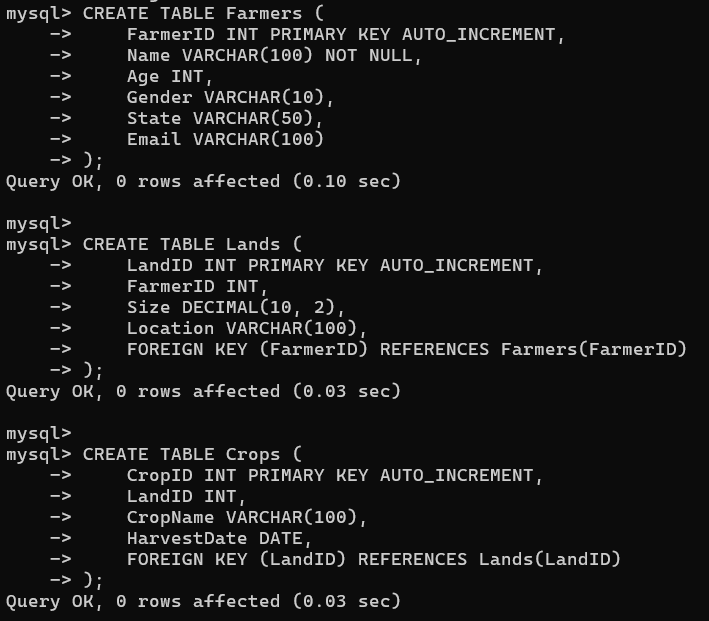
**SQL COMMANDS**

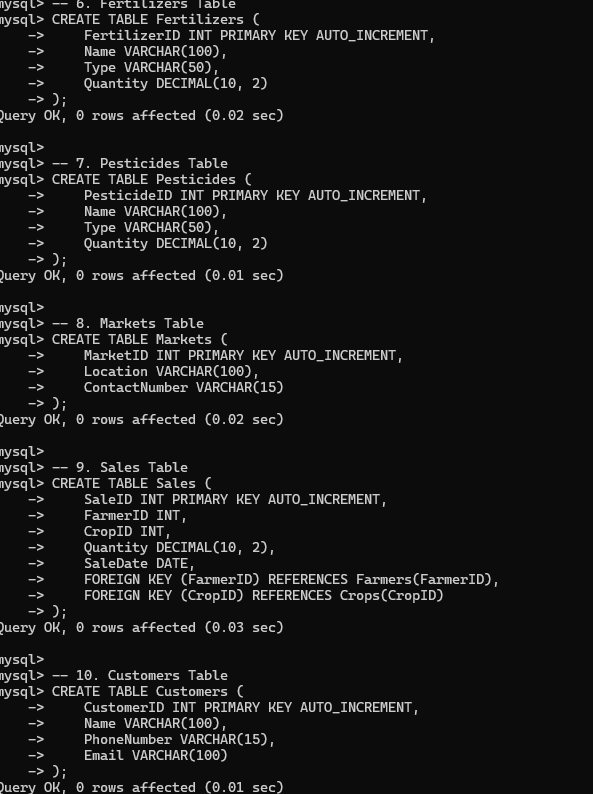
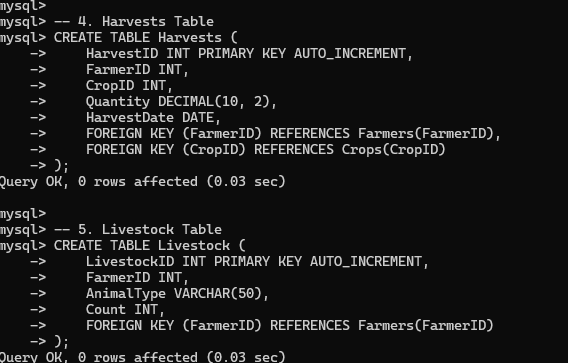
1. **Create a New Database:**

****

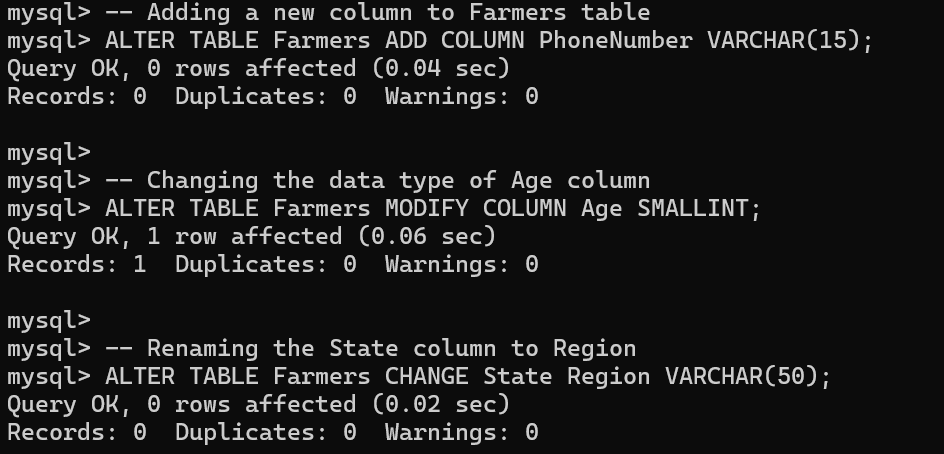
**2. Data Manipulation Language (DML)**

**a. Creating Tables**

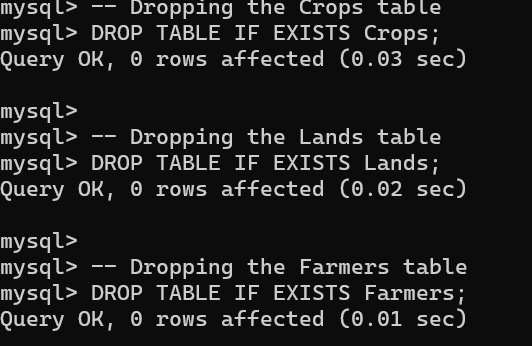
****

****

**b. Altering Tables**

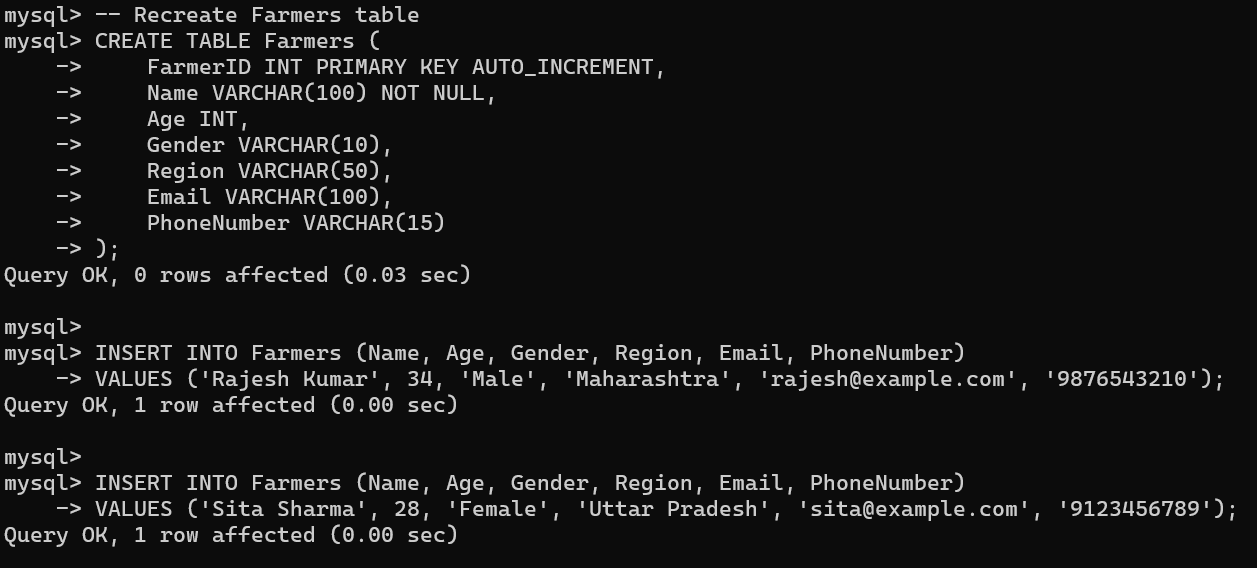
****

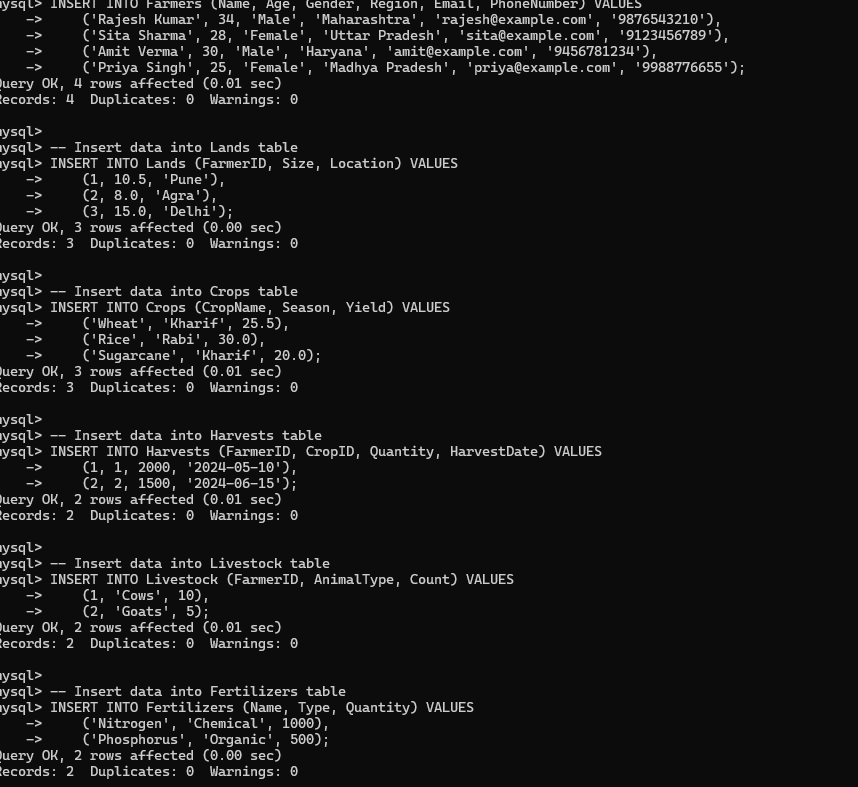
**c. Dropping Tables**

****

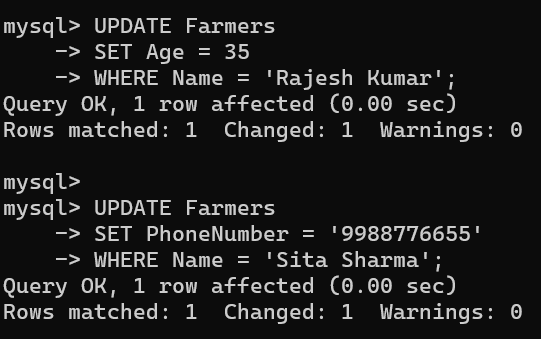
**3. DML Commands**

**a. Inserting Data**

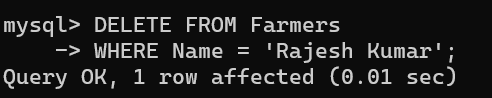
****

****

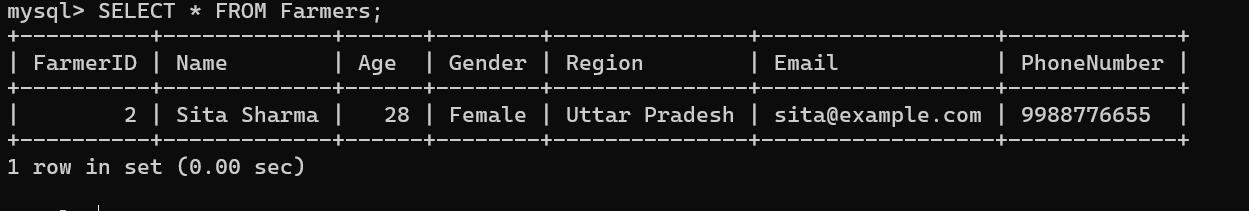
**b. Updating Data**

****

**c. Deleting Data**

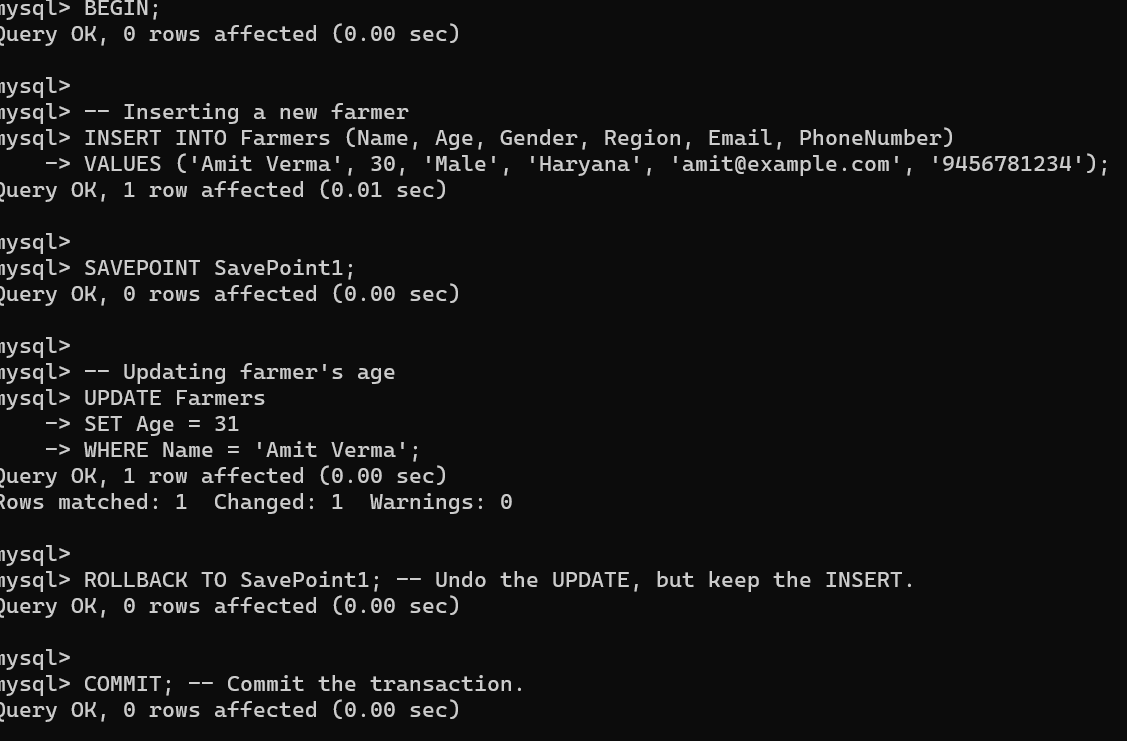
****

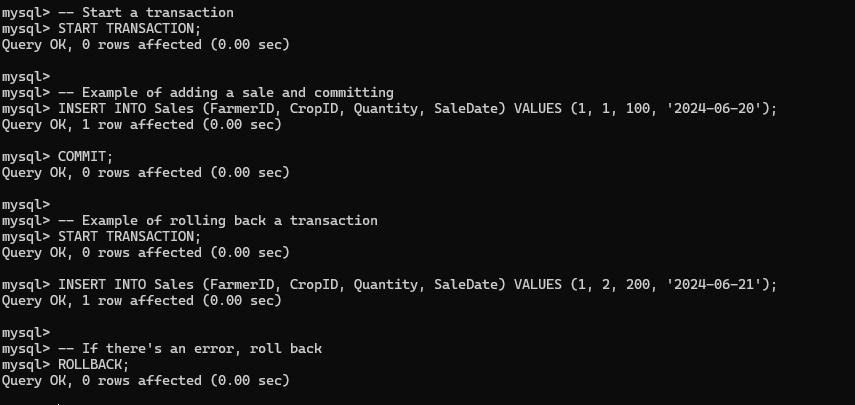
**d. Selecting Data**

****

**4. Transaction Control Language (TCL)**

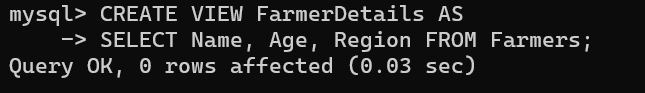
**a. Transaction Example**

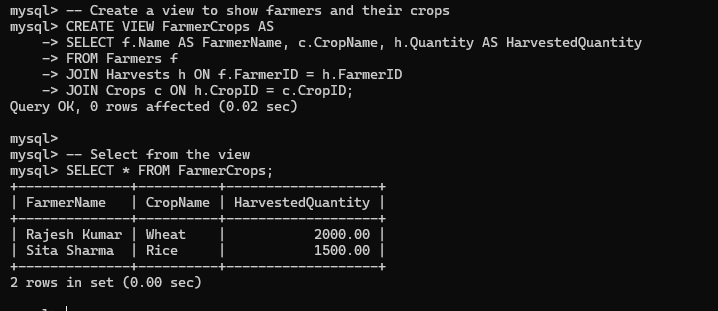
****

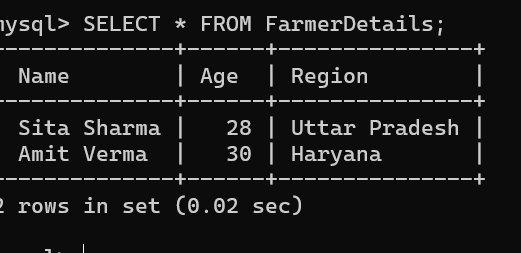
****

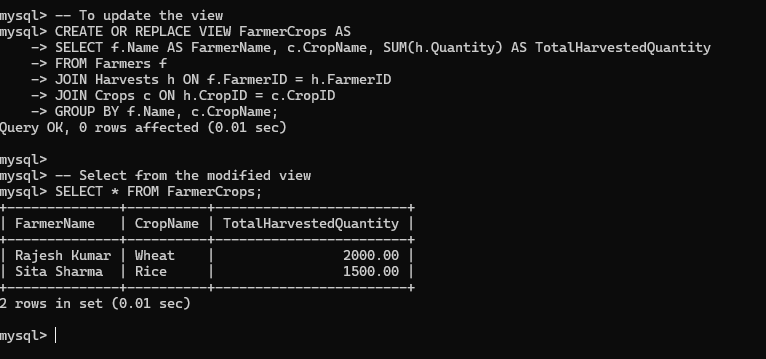
**5. View Definition Language (VDL)**

**a. Creating a View**

****

****

**b. Querying the View **

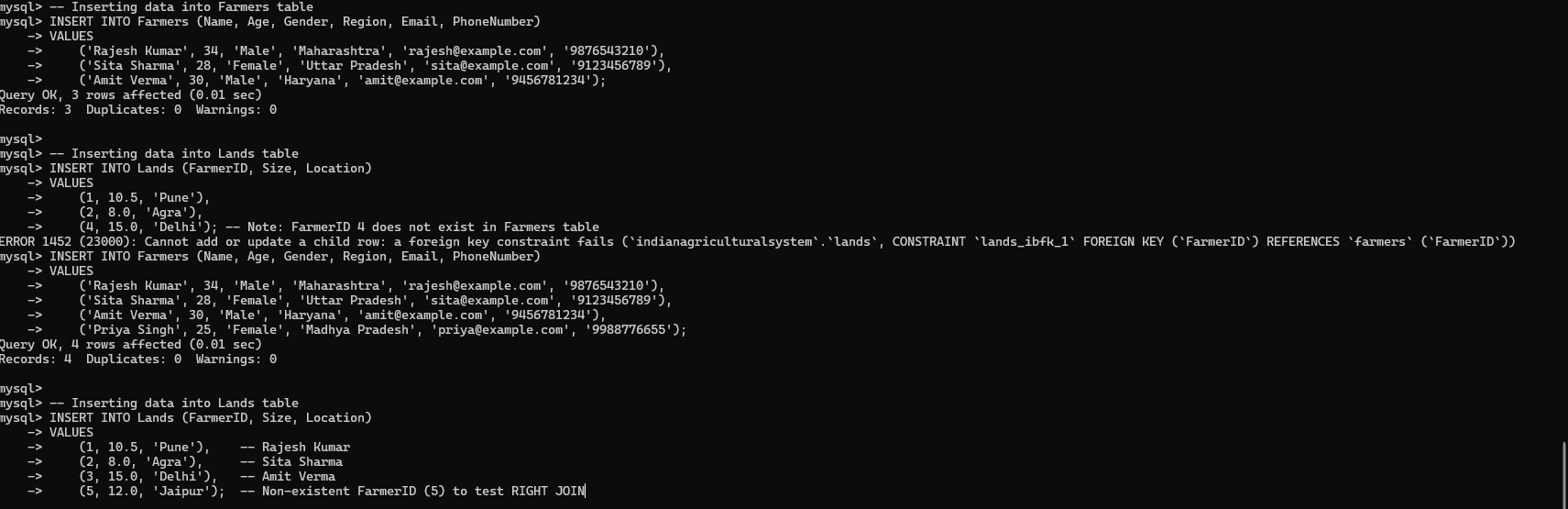
**c.Updating View **

**6. Join Operations**

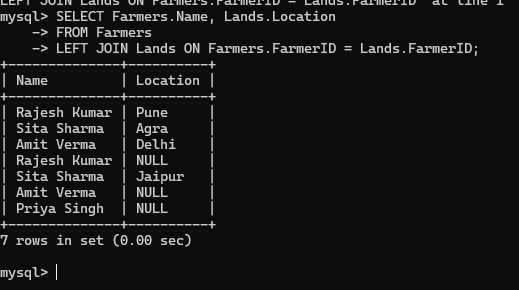
**a. INNER JOIN Example**

**Assuming you have re-created the Lands table after dropping it earlier:**

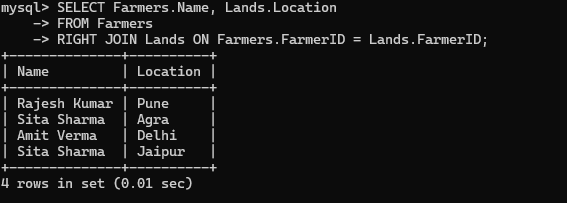
****

****

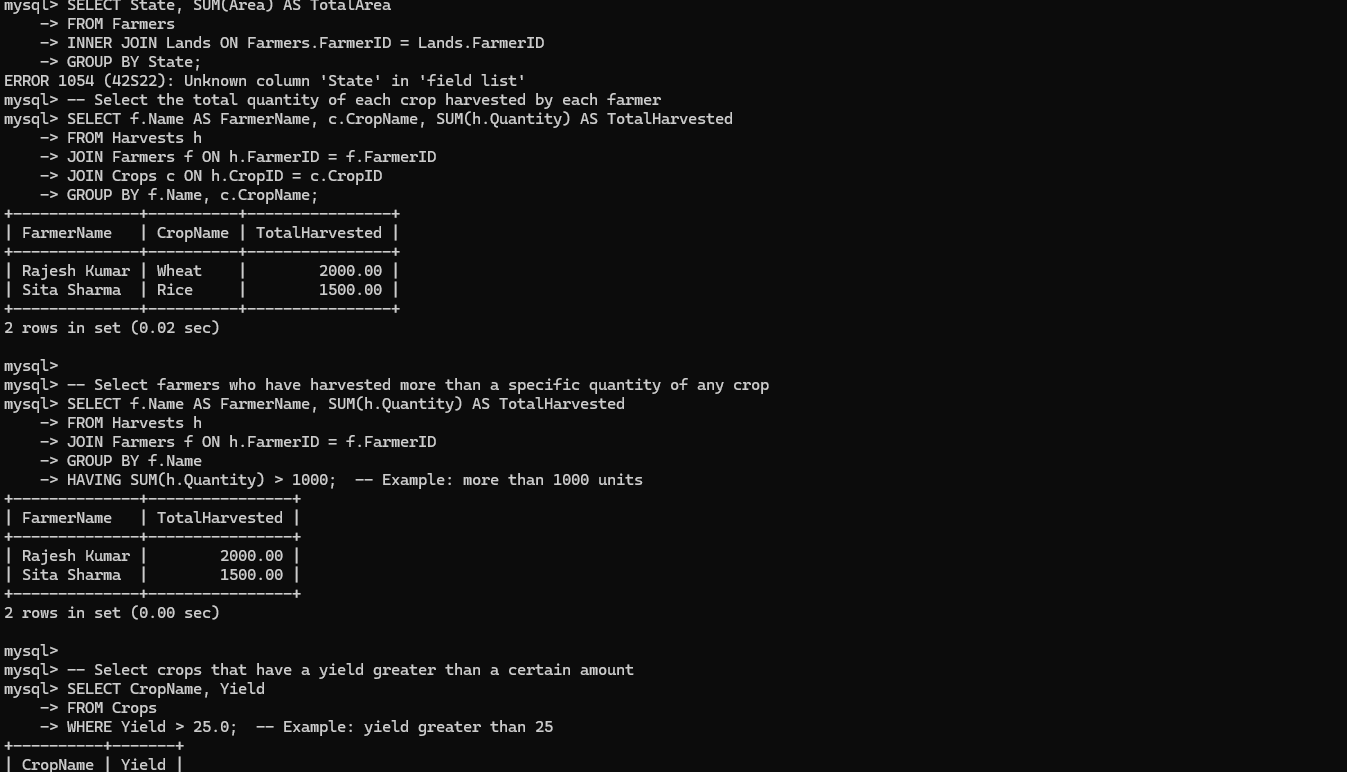
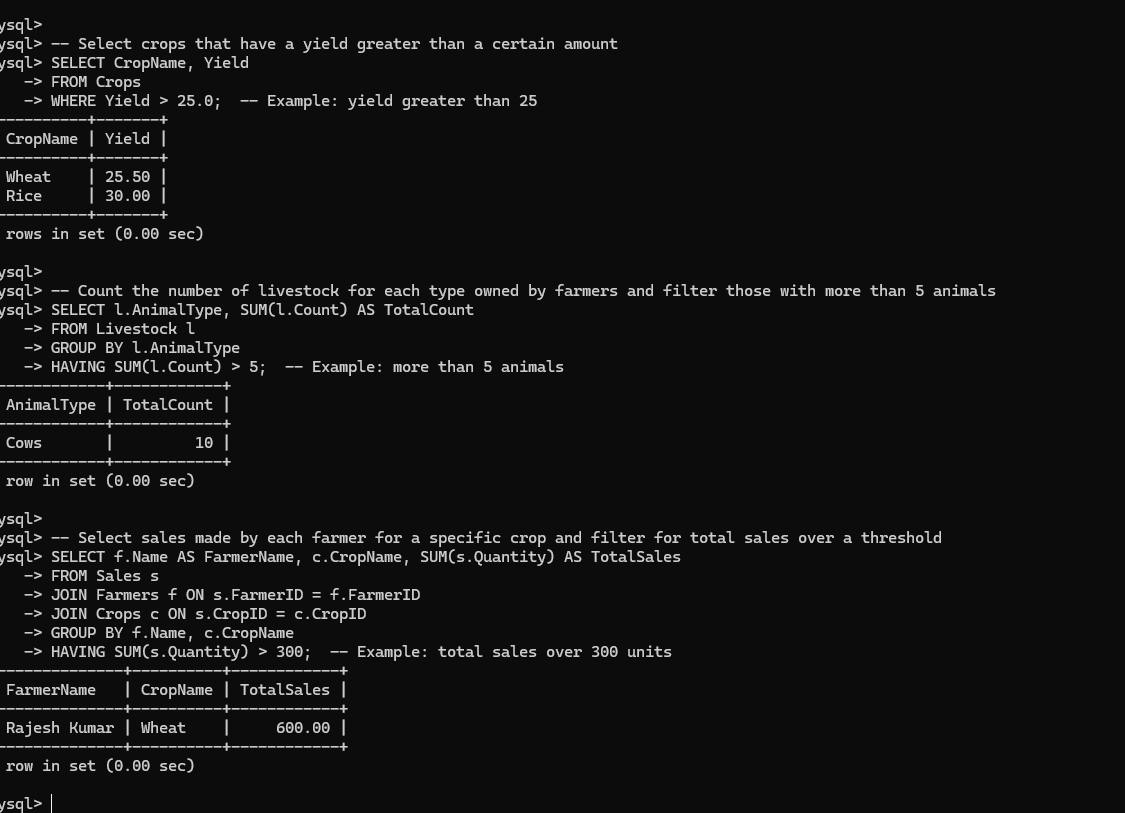
**b. LEFT JOIN Example**

****

**c. Right join**

****

**Additional SQL Queries to Include:**

****

**SQL Queries Only:**

**-- Create a new database**

**CREATE DATABASE IndianAgriculturalSystem;**

**USE IndianAgriculturalSystem;**

**-- 1. Farmers Table**

**CREATE TABLE Farmers (**

**FarmerID INT PRIMARY KEY AUTO\_INCREMENT,**

**Name VARCHAR(100) NOT NULL,**

**Age INT,**

**Gender VARCHAR(10),**

**Region VARCHAR(50),**

**Email VARCHAR(100),**

**PhoneNumber VARCHAR(15)**

**);**

**-- 2. Lands Table**

**CREATE TABLE Lands (**

**LandID INT PRIMARY KEY AUTO\_INCREMENT,**

**FarmerID INT,**

**Size DECIMAL(10, 2),**

**Location VARCHAR(100),**

**FOREIGN KEY (FarmerID) REFERENCES Farmers(FarmerID)**

**);**

**-- 3. Crops Table**

**CREATE TABLE Crops (**

**CropID INT PRIMARY KEY AUTO\_INCREMENT,**

**CropName VARCHAR(100) NOT NULL,**

**Season VARCHAR(50),**

**Yield DECIMAL(10, 2)**

**);**

**-- 4. Harvests Table**

**CREATE TABLE Harvests (**

**HarvestID INT PRIMARY KEY AUTO\_INCREMENT,**

**FarmerID INT,**

**CropID INT,**

**Quantity DECIMAL(10, 2),**

**HarvestDate DATE,**

**FOREIGN KEY (FarmerID) REFERENCES Farmers(FarmerID),**

**FOREIGN KEY (CropID) REFERENCES Crops(CropID)**

**);**

**-- 5. Livestock Table**

**CREATE TABLE Livestock (**

**LivestockID INT PRIMARY KEY AUTO\_INCREMENT,**

**FarmerID INT,**

**AnimalType VARCHAR(50),**

**Count INT,**

**FOREIGN KEY (FarmerID) REFERENCES Farmers(FarmerID)**

**);**

**-- 6. Fertilizers Table**

**CREATE TABLE Fertilizers (**

**FertilizerID INT PRIMARY KEY AUTO\_INCREMENT,**

**Name VARCHAR(100),**

**Type VARCHAR(50),**

**Quantity DECIMAL(10, 2)**

**);**

**-- 7. Pesticides Table**

**CREATE TABLE Pesticides (**

**PesticideID INT PRIMARY KEY AUTO\_INCREMENT,**

**Name VARCHAR(100),**

**Type VARCHAR(50),**

**Quantity DECIMAL(10, 2)**

**);**

**-- 8. Markets Table**

**CREATE TABLE Markets (**

**MarketID INT PRIMARY KEY AUTO\_INCREMENT,**

**Location VARCHAR(100),**

**ContactNumber VARCHAR(15)**

**);**

**-- 9. Sales Table**

**CREATE TABLE Sales (**

**SaleID INT PRIMARY KEY AUTO\_INCREMENT,**

**FarmerID INT,**

**CropID INT,**

**Quantity DECIMAL(10, 2),**

**SaleDate DATE,**

**FOREIGN KEY (FarmerID) REFERENCES Farmers(FarmerID),**

**FOREIGN KEY (CropID) REFERENCES Crops(CropID)**

**);**

**-- 10. Customers Table**

**CREATE TABLE Customers (**

**CustomerID INT PRIMARY KEY AUTO\_INCREMENT,**

**Name VARCHAR(100),**

**PhoneNumber VARCHAR(15),**

**Email VARCHAR(100)**

**);**

**-- Insert data into Farmers table**

**INSERT INTO Farmers (Name, Age, Gender, Region, Email, PhoneNumber) VALUES**

**('Rajesh Kumar', 34, 'Male', 'Maharashtra', 'rajesh@example.com', '9876543210'),**

**('Sita Sharma', 28, 'Female', 'Uttar Pradesh', 'sita@example.com', '9123456789'),**

**('Amit Verma', 30, 'Male', 'Haryana', 'amit@example.com', '9456781234'),**

**('Priya Singh', 25, 'Female', 'Madhya Pradesh', 'priya@example.com', '9988776655');**

**-- Insert data into Lands table**

**INSERT INTO Lands (FarmerID, Size, Location) VALUES**

**(1, 10.5, 'Pune'),**

**(2, 8.0, 'Agra'),**

**(3, 15.0, 'Delhi');**

**-- Insert data into Crops table**

**INSERT INTO Crops (CropName, Season, Yield) VALUES**

**('Wheat', 'Kharif', 25.5),**

**('Rice', 'Rabi', 30.0),**

**('Sugarcane', 'Kharif', 20.0);**

**-- Insert data into Harvests table**

**INSERT INTO Harvests (FarmerID, CropID, Quantity, HarvestDate) VALUES**

**(1, 1, 2000, '2024-05-10'),**

**(2, 2, 1500, '2024-06-15');**

**-- Insert data into Livestock table**

**INSERT INTO Livestock (FarmerID, AnimalType, Count) VALUES**

**(1, 'Cows', 10),**

**(2, 'Goats', 5);**

**-- Insert data into Fertilizers table**

**INSERT INTO Fertilizers (Name, Type, Quantity) VALUES**

**('Nitrogen', 'Chemical', 1000),**

**('Phosphorus', 'Organic', 500);**

**-- Insert data into Pesticides table**

**INSERT INTO Pesticides (Name, Type, Quantity) VALUES**

**('Insecticide A', 'Insecticide', 300),**

**('Fungicide B', 'Fungicide', 200);**

**-- Insert data into Markets table**

**INSERT INTO Markets (Location, ContactNumber) VALUES**

**('Pune Market', '1234567890'),**

**('Agra Market', '2345678901');**

**-- Insert data into Sales table**

**INSERT INTO Sales (FarmerID, CropID, Quantity, SaleDate) VALUES**

**(1, 1, 500, '2024-05-12'),**

**(2, 2, 300, '2024-06-17');**

**-- Insert data into Customers table**

**INSERT INTO Customers (Name, PhoneNumber, Email) VALUES**

**('Ravi Mehta', '9876543210', 'ravi@example.com'),**

**('Anjali Bhatia', '1234567890', 'anjali@example.com');**

**-- Update a farmer's contact information**

**UPDATE Farmers SET PhoneNumber = '9999999999' WHERE FarmerID = 1;**

**-- Update a crop's yield**

**UPDATE Crops SET Yield = 28.0 WHERE CropID = 1;**

**-- Delete a customer**

**DELETE FROM Customers WHERE CustomerID = 2;**

**-- Delete a farmer**

**DELETE FROM Farmers WHERE FarmerID = 3;**

**-- Select all farmers**

**SELECT \* FROM Farmers;**

**-- Select all crops**

**SELECT \* FROM Crops;**

**-- Select all sales records**

**SELECT \* FROM Sales;**

**-- Start a transaction**

**START TRANSACTION;**

**-- Example of adding a sale and committing**

**INSERT INTO Sales (FarmerID, CropID, Quantity, SaleDate) VALUES (1, 1, 100, '2024-06-20');**

**COMMIT;**

**-- Example of rolling back a transaction**

**START TRANSACTION;**

**INSERT INTO Sales (FarmerID, CropID, Quantity, SaleDate) VALUES (1, 2, 200, '2024-06-21');**

**-- If there's an error, roll back**

**ROLLBACK;**

**-- Create a view to show farmers and their crops**

**CREATE VIEW FarmerCrops AS**

**SELECT f.Name AS FarmerName, c.CropName, h.Quantity AS HarvestedQuantity**

**FROM Farmers f**

**JOIN Harvests h ON f.FarmerID = h.FarmerID**

**JOIN Crops c ON h.CropID = c.CropID;**

**-- Select from the view**

**SELECT \* FROM FarmerCrops;**

**-- To update the view**

**CREATE OR REPLACE VIEW FarmerCrops AS**

**SELECT f.Name AS FarmerName, c.CropName, SUM(h.Quantity) AS TotalHarvestedQuantity**

**FROM Farmers f**

**JOIN Harvests h ON f.FarmerID = h.FarmerID**

**JOIN Crops c ON h.CropID = c.CropID**

**GROUP BY f.Name, c.CropName;**

**-- Select from the modified view**

**SELECT \* FROM FarmerCrops;**

**-- Aggregate Queries**

**-- Total harvested quantity by farmer and crop**

**SELECT f.Name AS FarmerName, c.CropName, SUM(h.Quantity) AS TotalHarvested**

**FROM Harvests h**

**JOIN Farmers f ON h.FarmerID = f.FarmerID**

**JOIN Crops c ON h.CropID = c.CropID**

**GROUP BY f.Name, c.CropName;**

**-- Farmers with significant harvests**

**SELECT f.Name AS FarmerName, SUM(h.Quantity) AS TotalHarvested**

**FROM Harvests h**

**JOIN Farmers f ON h.FarmerID = f.FarmerID**

**GROUP BY f.Name**

**HAVING SUM(h.Quantity) > 1000; -- Example: more than 1000 units**

**-- Crops with high yield**

**SELECT CropName, Yield**

**FROM Crops**

**WHERE Yield > 25.0; -- Example: yield greater than 25**

**-- Count of livestock for each type owned by farmers**

**SELECT l.AnimalType, SUM(l.Count) AS TotalCount**

**FROM Livestock l**

**GROUP BY l.AnimalType**

**HAVING SUM(l.Count) > 5; -- Example: more than 5 animals**

**-- Sales by farmer and crop**

**SELECT f.Name AS FarmerName, c.CropName, SUM(s.Quantity) AS TotalSales**

**FROM Sales s**

**JOIN Farmers f ON s.FarmerID = f.FarmerID**

**JOIN Crops c ON s.CropID = c.CropID**

**GROUP BY f.Name, c.CropName**

**HAVING SUM(s.Quantity) > 300; -- Example: total sales over 300 units**