**DBMS CIA-3**

**Indian Agricultural System**

**DESCRIPTION**

The Indian Agricultural System is a dynamic, complex network that plays a pivotal role in the economy of India. The agricultural sector encompasses all participants in the form of farmers, crops, markets, and transactions of selling and buying of crops. The proposed DBMS models some of the most striking features related to the agricultural system, which include data management related to farmers, crops they produce, the markets they interact with, and the transactions made.  
  
The database has been developed to systematically store and oversee essential information, including details regarding farmers, varieties of crops grown, market sites, and sales records. This initiative aids in optimizing agricultural operations, guaranteeing precise and effective data storage, retrieval, and analysis.  
  
This database system encompasses multiple tables, such as Farmer, Crop, Market, and Transaction. The relationships between these tables are modeled with the use of primary and foreign keys to ensure data integrity in the system. In fact, the system allows several SQL operations, including data manipulation, querying, and reporting, to manage agricultural information efficiently.

* **Create a Database:**

The first function perform is to create a database called Agriculture.

**CODE:**

> create database Agriculture;

**OUTPUT:**

Query OK, 1 row affected (0.00 sec)

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

DDL commands are used to define the data structures and the structure of the database. There are 5 different commands in DDL ; Rename, Create, Drop, Alter, Truncate.

**CODE:**

CREATE TABLE Farmer (

-> Farmer\_ID INT PRIMARY KEY AUTO\_INCREMENT,

-> Name VARCHAR(50) NOT NULL,

-> Age INT CHECK (Age > 18),

-> Address VARCHAR(100) NOT NULL,

-> Contact\_Number VARCHAR(15) UNIQUE NOT NULL

-> );

**OUTPUT:**

Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Crop (

-> Crop\_ID INT PRIMARY KEY AUTO\_INCREMENT,

-> Name VARCHAR(50) NOT NULL,

-> Type VARCHAR(30) NOT NULL,

-> Season VARCHAR(20) DEFAULT 'Kharif',

-> Farmer\_ID INT,

-> FOREIGN KEY (Farmer\_ID) REFERENCES Farmer(Farmer\_ID) ON DELETE CASCADE

-> );

Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Market (

-> Market\_ID INT PRIMARY KEY AUTO\_INCREMENT,

-> Location VARCHAR(100) NOT NULL,

-> Market\_Type VARCHAR(30),

-> Crop\_ID INT,

-> FOREIGN KEY (Crop\_ID) REFERENCES Crop(Crop\_ID) ON DELETE SET NULL

-> );

Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Transaction (

-> Transaction\_ID INT PRIMARY KEY AUTO\_INCREMENT,

-> Farmer\_ID INT,

-> Market\_ID INT,

-> Transaction\_Date DATE NOT NULL,

-> Amount DECIMAL(10, 2) NOT NULL,

-> FOREIGN KEY (Farmer\_ID) REFERENCES Farmer(Farmer\_ID) ON DELETE CASCADE,

-> FOREIGN KEY (Market\_ID) REFERENCES Market(Market\_ID) ON DELETE CASCADE

-> );

Query OK, 0 rows affected (0.02 sec)

**CODE**

ALTER TABLE Crop ADD COLUMN Season VARCHAR(20) DEFAULT 'Kharif';

**OUTPUT**

Query OK, 0 rows affected

Records: 0 Duplicates: 0 Warnings: 0

* **CONSTRAINTS**

1. AUTO\_INCREMENT: Automatically generates a unique Farmer\_ID.
2. PRIMARY KEY: Ensures Farmer\_ID is unique and not null.
3. NOT NULL: Fields like Name and Contact\_Number must have values.
4. CHECK: Ensures Age is greater than 18.
5. UNIQUE: Ensures that Contact\_Number is unique.

* **DML (Data Manipulation Language)**

DML statements are used to manipulate data stored in the tables. The types of the DML commands are SELECT , INSERT , DELETE , and UPDATE.

**CODE:**

INSERT INTO Farmer (Name, Age, Address, Contact\_Number)

-> VALUES

-> ('Ravi Kumar', 45, 'Village XYZ', '9876543210'),

-> ('Anita Sharma', 35, 'Village ABC', '8765432109'),

-> ('Rahul Verma', 50, 'Village LMN', '7654321098'),

-> ('Pooja Singh', 28, 'Village DEF', '6543210987');

**OUTPUT:**

Query OK, 4 rows affected (0.01 sec)

Records: 4 Duplicates: 0 Warnings: 0

mysql> INSERT INTO Crop (Name, Type, Season, Farmer\_ID)

-> VALUES

-> ('Wheat', 'Cereal', 'Rabi', 1),

-> ('Rice', 'Cereal', 'Kharif', 2),

-> ('Sugarcane', 'Cash Crop', 'Kharif', 3),

-> ('Cotton', 'Cash Crop', 'Kharif', 4),

-> ('Barley', 'Cereal', 'Rabi', 1);

Query OK, 5 rows affected (0.00 sec)

Records: 5 Duplicates: 0 Warnings: 0

mysql> INSERT INTO Market (Location, Market\_Type, Crop\_ID)

-> VALUES

-> ('Delhi', 'Wholesale', 1),

-> ('Mumbai', 'Retail', 2),

-> ('Chennai', 'Wholesale', 3),

-> ('Kolkata', 'Retail', 4),

-> ('Hyderabad', 'Wholesale', 5);

Query OK, 5 rows affected (0.01 sec)

Records: 5 Duplicates: 0 Warnings: 0

mysql> INSERT INTO Transaction (Farmer\_ID, Market\_ID, Transaction\_Date, Amount)

-> VALUES

-> (1, 1, '2023-09-15', 50000.00),

-> (2, 2, '2023-10-01', 65000.00),

-> (3, 3, '2023-10-05', 45000.00),

-> (4, 4, '2023-10-10', 30000.00),

-> (1, 5, '2023-09-20', 55000.00);

Query OK, 5 rows affected (0.00 sec)

Records: 5 Duplicates: 0 Warnings: 0

**CODE:**

DELETE FROM Farmer WHERE Farmer\_ID = 4;

**OUTPUT:**

Query OK, 1 row affected (0.00 sec)

**CODE:**

UPDATE Crop SET Season = 'Rabi' WHERE Crop\_ID = 1;

**OUTPUT:**

Query OK, 0 rows affected (0.00 sec)

Rows matched: 1 Changed: 0 Warnings: 0

**CODE:**

Select\* from market:

**OUTPUT:**

+-----------+-----------+-------------+---------+

| Market\_ID | Location | Market\_Type | Crop\_ID |

+-----------+-----------+-------------+---------+

| 1 | Delhi | Wholesale | 1 |

| 2 | Mumbai | Retail | 2 |

| 3 | Chennai | Wholesale | 3 |

| 4 | Kolkata | Retail | NULL |

| 5 | Hyderabad | Wholesale | 5 |

+-----------+-----------+-------------+---------+

5 rows in set (0.00 sec)

* **TCL (Transaction Control Language):**

TCL commands manage transactions within the database.

**CODE:**

mysql> BEGIN; -- Start transaction

**OUTPUT:**

Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Transaction (Farmer\_ID, Market\_ID, Transaction\_Date, Amount)

-> VALUES (1, 1, '2023-09-15', 50000.00);

Query OK, 1 row affected (0.00 sec)

**CODE :**

mysql> COMMIT; -- Save the transaction

**OUTPUT:**

Query OK, 0 rows affected (0.00 sec)

mysql> BEGIN;

Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE Farmer SET Age = 30 WHERE Farmer\_ID = 1;

Query OK, 1 row affected (0.00 sec)

Rows matched: 1 Changed: 1 Warnings: 0

**CODE :**

mysql> ROLLBACK; -- Reverts the change

**OUTPUT:**

Query OK, 0 rows affected (0.00 sec)

* **VDL (View Definition Language):**

VDL commands manage database views.

**CODE:**

mysql> CREATE VIEW Crop\_Sales AS

-> SELECT Farmer.Name, Crop.Name AS Crop\_Name, Transaction.Amount

-> FROM Farmer

-> JOIN Crop ON Farmer.Farmer\_ID = Crop.Farmer\_ID

-> JOIN Transaction ON Farmer.Farmer\_ID = Transaction.Farmer\_ID;

**OUTPUT:**

Query OK, 0 rows affected (0.01 sec)

* **WHERE Clause:**

Filters records based on specific conditions.

**CODE:**

mysql> SELECT \* FROM Crop WHERE Type = 'Cereal';

**OUTPUT:**

+---------+--------+--------+--------+-----------+

| Crop\_ID | Name | Type | Season | Farmer\_ID |

+---------+--------+--------+--------+-----------+

| 1 | Wheat | Cereal | Rabi | 1 |

| 2 | Rice | Cereal | Kharif | 2 |

| 5 | Barley | Cereal | Rabi | 1 |

+---------+--------+--------+--------+-----------+

3 rows in set (0.00 sec)

* **GROUP BY Clause:**

Groups rows that have the same values into summary rows.

**CODE:**

mysql> SELECT Season, COUNT(Crop\_ID) AS Crop\_Count FROM Crop GROUP BY Season HAVING Crop\_Count > 1;

**OUTPUT:**

+--------+------------+

| Season | Crop\_Count |

+--------+------------+

| Rabi | 2 |

| Kharif | 2 |

+--------+------------+

2 rows in set (0.00 sec)

* **HAVING Clause;**

Used to filter groups after the GROUP BY clause.

**CODE:**

mysql> SELECT Season, COUNT(Crop\_ID) AS Crop\_Count FROM Crop GROUP BY Season HAVING Crop\_Count > 1;

**OUTPUT:**

+--------+------------+

| Season | Crop\_Count |

+--------+------------+

| Rabi | 2 |

| Kharif | 2 |

+--------+------------+

2 rows in set (0.00 sec)

* **ORDER BY Clause**

Sorts the result set in ascending or descending order.

**CODE:**

mysql> SELECT Name, Age FROM Farmer ORDER BY Age DESC;

**OUTPUT:**

+--------------+------+

| Name | Age |

+--------------+------+

| Rahul Verma | 50 |

| Ravi Kumar | 45 |

| Anita Sharma | 35 |

+--------------+------+

3 rows in set (0.00 sec)

* **DISTINCT and LIMIT Keywords:**

**DISTINCT**: Removes duplicate values.

**LIMIT**: Restricts the number of rows returned.

**CODE:**

mysql> SELECT DISTINCT Name FROM Crop LIMIT 3;

**OUTPUT:**

+-----------+

| Name |

+-----------+

| Wheat |

| Rice |

| Sugarcane |

+-----------+

3 rows in set (0.00 sec)

* **AGGREGATE Functions:**

The aggregate functions show some numerical and calculate the aggregates. They are SUM, AVG, COUNT, MAX, MIN.

**CODE:**

mysql> SELECT AVG(Amount) AS Average\_Amount FROM Transaction;

**OUTPUT:**

+----------------+

| Average\_Amount |

+----------------+

| 53000.000000 |

+----------------+

1 row in set (0.00 sec)

* **Pattern Matching (LIKE):**

Matches a specified pattern using wildcards.

**CODE:**

mysql> SELECT \* FROM Farmer WHERE Name LIKE 'R%';

**OUTPUT:**

+-----------+-------------+------+-------------+----------------+

| Farmer\_ID | Name | Age | Address | Contact\_Number |

+-----------+-------------+------+-------------+----------------+

| 1 | Ravi Kumar | 45 | Village XYZ | 9876543210 |

| 3 | Rahul Verma | 50 | Village LMN | 7654321098 |

+-----------+-------------+------+-------------+----------------+

2 rows in set (0.00 sec)

* **STRING Functions:**

The string functions is used to change the string functions into particular cases and so on.

**CODE:**

mysql> SELECT UPPER(Name) AS Uppercase\_Name FROM Farmer;

**OUTPUT:**

+----------------+

| Uppercase\_Name |

+----------------+

| RAVI KUMAR |

| ANITA SHARMA |

| RAHUL VERMA |

+----------------+

3 rows in set (0.00 sec)

* **NUMERIC Functions:**

The numeric function deals the numerals so that it can round up to something and be precise.ROUND Rounds numbers to a specified decimal point.

**CODE:**

mysql> SELECT ROUND(AVG(Age), 2) AS Rounded\_Age FROM Farmer;

**OUTPUT:**

+-------------+

| Rounded\_Age |

+-------------+

| 43.33 |

+-------------+

1 row in set (0.00 sec)

* **DATE-TIME Functions:**

The date and time functions show the date and time. **CURDATE()**: Returns the current date. **YEAR()**: Extracts the year from a date.

**CODE:**

mysql> SELECT CURDATE() AS Today, YEAR(Transaction\_Date) AS Year FROM Transaction;

5 rows in set (0.00 sec)

**OUTPUT:**

+------------+------+

| Today | Year |

+------------+------+

| 2024-10-16 | 2023 |

| 2024-10-16 | 2023 |

| 2024-10-16 | 2023 |

| 2024-10-16 | 2023 |

| 2024-10-16 | 2023 |

+------------+------+

### ****NESTED Query:****

### A query within another query.

### CODE:

### mysql> SELECT Name FROM Farmer WHERE Farmer\_ID IN (SELECT Farmer\_ID FROM Crop WHERE Type = 'Cereal');

### +--------------+

### | Name |

### +--------------+

### | Ravi Kumar |

### | Anita Sharma |

### +--------------+

### 2 rows in set (0.01 sec)

### ****Cartesian Product****

### Returns all possible combinations of rows from two or more tables.

### CODE:

### mysql> SELECT \* FROM Farmer, Crop;

### OUTPUT:

### +-----------+--------------+------+-------------+----------------+---------+-----------+-----------+--------+-----------+

### | Farmer\_ID | Name | Age | Address | Contact\_Number | Crop\_ID | Name | Type | Season | Farmer\_ID |

### +-----------+--------------+------+-------------+----------------+---------+-----------+-----------+--------+-----------+

### | 3 | Rahul Verma | 50 | Village LMN | 7654321098 | 1 | Wheat | Cereal | Rabi | 1 |

### | 2 | Anita Sharma | 35 | Village ABC | 8765432109 | 1 | Wheat | Cereal | Rabi | 1 |

### | 1 | Ravi Kumar | 45 | Village XYZ | 9876543210 | 1 | Wheat | Cereal | Rabi | 1 |

### | 3 | Rahul Verma | 50 | Village LMN | 7654321098 | 2 | Rice | Cereal | Kharif | 2 |

### | 2 | Anita Sharma | 35 | Village ABC | 8765432109 | 2 | Rice | Cereal | Kharif | 2 |

### | 1 | Ravi Kumar | 45 | Village XYZ | 9876543210 | 2 | Rice | Cereal | Kharif | 2 |

### | 3 | Rahul Verma | 50 | Village LMN | 7654321098 | 3 | Sugarcane | Cash Crop | Kharif | 3 |

### | 2 | Anita Sharma | 35 | Village ABC | 8765432109 | 3 | Sugarcane | Cash Crop | Kharif | 3 |

### | 1 | Ravi Kumar | 45 | Village XYZ | 9876543210 | 3 | Sugarcane | Cash Crop | Kharif | 3 |

### | 3 | Rahul Verma | 50 | Village LMN | 7654321098 | 5 | Barley | Cereal | Rabi | 1 |

### | 2 | Anita Sharma | 35 | Village ABC | 8765432109 | 5 | Barley | Cereal | Rabi | 1 |

### | 1 | Ravi Kumar | 45 | Village XYZ | 9876543210 | 5 | Barley | Cereal | Rabi | 1 |

### +-----------+--------------+------+-------------+----------------+---------+-----------+-----------+--------+-----------+

### 12 rows in set (0.00 sec)

### DIVISION: CODE:

### mysql> SELECT Farmer\_ID FROM Transaction WHERE Amount > 50000;

### OUTPUT:

### +-----------+

### | Farmer\_ID |

### +-----------+

### | 2 |

### | 1 |

### +-----------+

### 2 rows in set (0.00 sec)

* **JOINS:**

 **INNER JOIN**: Returns only matching rows from both tables.

 **LEFT JOIN**: Returns all rows from the left table and matching rows from the right table (NULL if no match).

 **RIGHT JOIN**: Returns all rows from the right table and matching rows from the left table (NULL if no match).

 **FULL JOIN**: Returns all rows from both tables (NULL where there is no match).

 **CROSS JOIN**: Returns the Cartesian product of both tables (all possible combinations).

 **SELF JOIN**: Joins a table with itself to compare its rows.

**CODE:**

mysql> SELECT Farmer.Name, Crop.Name AS Crop\_Name FROM Farmer

-> INNER JOIN Crop ON Farmer.Farmer\_ID = Crop.Farmer\_ID;

**OUTPUT:**

+--------------+-----------+

| Name | Crop\_Name |

+--------------+-----------+

| Ravi Kumar | Wheat |

| Ravi Kumar | Barley |

| Anita Sharma | Rice |

| Rahul Verma | Sugarcane |

+--------------+-----------+

4 rows in set (0.00 sec)

### CODE:

### mysql> SELECT Farmer.Name, Crop.Name AS Crop\_Name FROM Farmer

### -> LEFT JOIN Crop ON Farmer.Farmer\_ID = Crop.Farmer\_ID;

### OUTPUT:

### +--------------+-----------+

### | Name | Crop\_Name |

### +--------------+-----------+

### | Ravi Kumar | Wheat |

### | Ravi Kumar | Barley |

### | Anita Sharma | Rice |

### | Rahul Verma | Sugarcane |

### +--------------+-----------+

### 4 rows in set (0.00 sec)

### CODE:

### mysql> SELECT Farmer.Name, Crop.Name AS Crop\_Name

### -> FROM Farmer

### -> RIGHT JOIN Crop ON Farmer.Farmer\_ID = Crop.Farmer\_ID;

### OUTPUT:

### +--------------+-----------+

### | Name | Crop\_Name |

### +--------------+-----------+

### | Ravi Kumar | Wheat |

### | Anita Sharma | Rice |

### | Rahul Verma | Sugarcane |

### | Ravi Kumar | Barley |

### +--------------+-----------+

### 4 rows in set (0.00 sec)

### ER Diagram:

### 