**Project Report**

|  |  |  |
| --- | --- | --- |
| Program | B.Tech Artificial Intelligence | |
| Semester | 4th | |
| Name of the Project: | Dairy Management Database System | |
|  | | |
| Details of Project Members |  |  |
| Batch | Roll No. | Name |
| 2023-2027 | A009 | Daksh Soni |
| 2023-2027 | A022 | Priyanshu Nayak |
| 2023-2027 | A031 | Varun Sahu |
| Date of Submission: | | |

**Contribution of each project Members:**

|  |  |  |
| --- | --- | --- |
| Roll No. | Name: | Contribution |
| A009 | Daksh Soni | Components of Database Design, Entity Relationship Diagram, Relational Model |
| A022 | Priyanshu Nayak | Normalization, SQL Queries, Power BI |
| A031 | Varun Sahu | Website development, power BI dashboard |

**Project Report**

**Dairy Management Database System**

**by**

**Daksh Soni, Roll number: A009**

**Priyanshu Nayak, Roll number: A022**

**Varun Sahu, Roll number: A031**

**Course: DBMS**

**AY: 2024-25**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr no.** | **Topic** | **Page no.** |
| **1** | Storyline | 4 |
| **2** | Components of Database Design | 5-6 |
| **3** | Entity Relationship Diagram | 6 |
| **4** | Relational Model | 7-8 |
| **5** | Normalization | 9-11 |
| **6** | SQL Queries | 12-18 |
| **7** | Learning from the Project |  |
| **8** | Project Demonstration |  |
| **9** | Self-learning beyond classroom |  |
| **10** | Learning from the project |  |
| **11** | Challenges faced |  |
| **12** | Conclusion |  |

**I. Storyline**

Dairy Management Database System

In a small town, **Shree Dairy Farms** operates as a bridge between local farmers and city customers, ensuring fresh and high-quality milk delivery. The dairy collects milk twice daily—once in the morning and once in the evening—from a network of registered farmers. Each farmer is assigned a unique identification number, which helps track the milk they supply, including quantity, fat percentage, and other milk components.

* The pricing of the milk is determined based on its fat content, and payments to farmers are processed every ten days based on their total supplied quantity and quality. To ensure smooth operations, the dairy maintains a detailed financial record of these transactions.
* After collection, the milk is processed, packaged, and distributed to city households through a door-to-door service. Customers receive their milk daily and are billed on a monthly basis. Their records are maintained in the system to track payments, delivery schedules, and outstanding dues.
* To efficiently manage the entire system, a database is designed to handle farmer records, milk collection details, financial transactions, and customer information. This ensures transparency, accuracy, and seamless dairy operations.

### ****Key Features of Shree Dairy’s Operations****

* **Twice-Daily Milk Collection** – Regular pickups from registered farmers with detailed tracking
* **Quality-Based Pricing** – Payments calculated based on fat content and volume
* **Automated Billing System** – Monthly billing and dues tracking for customers
* **Transparent Farmer Payments** – Digital financial records and timely payouts
* **Reliable Door-to-Door Delivery** – Fresh milk delivered daily to city households
* **Centralized Database Management** – Handles all farmer, collection, financial, and customer data

**II. Components of Database Design**

 **Dairy**  
Represents the central dairy unit.  
Stores identity, location, and contact details of each dairy.

 **Farmer**  
Represents individual farmers associated with a dairy.  
Linked to the Dairy table. Supplies milk.

 **MilkCollection**  
Logs daily milk collected from each farmer.  
Includes session (Morning/Evening), quantity, fat %, and pricing details.

 **FarmerPayment**  
Tracks payments made to farmers based on milk supplied.  
Includes payment period, total supplied, and payment status.

 **MilkStorage**  
Tracks milk stored daily at the dairy.  
Summarizes morning and evening collection totals.

 **Customer**  
Represents individuals or businesses who buy milk.  
Linked to the Dairy. Stores contact and address.

 **MilkSales**  
Logs milk sold to customers.  
Includes quantity, rate, and total transaction amount.

 **CustomerPayment**  
Tracks payments from customers for milk purchased.  
Includes payment status and billing month.

 **Delivery**  
Manages delivery records of milk to customers.  
Linked to employees, customers, and sales.

 **Employee**  
Stores employee information working at the dairy.  
Includes role, contact, salary, and dairy assignment.

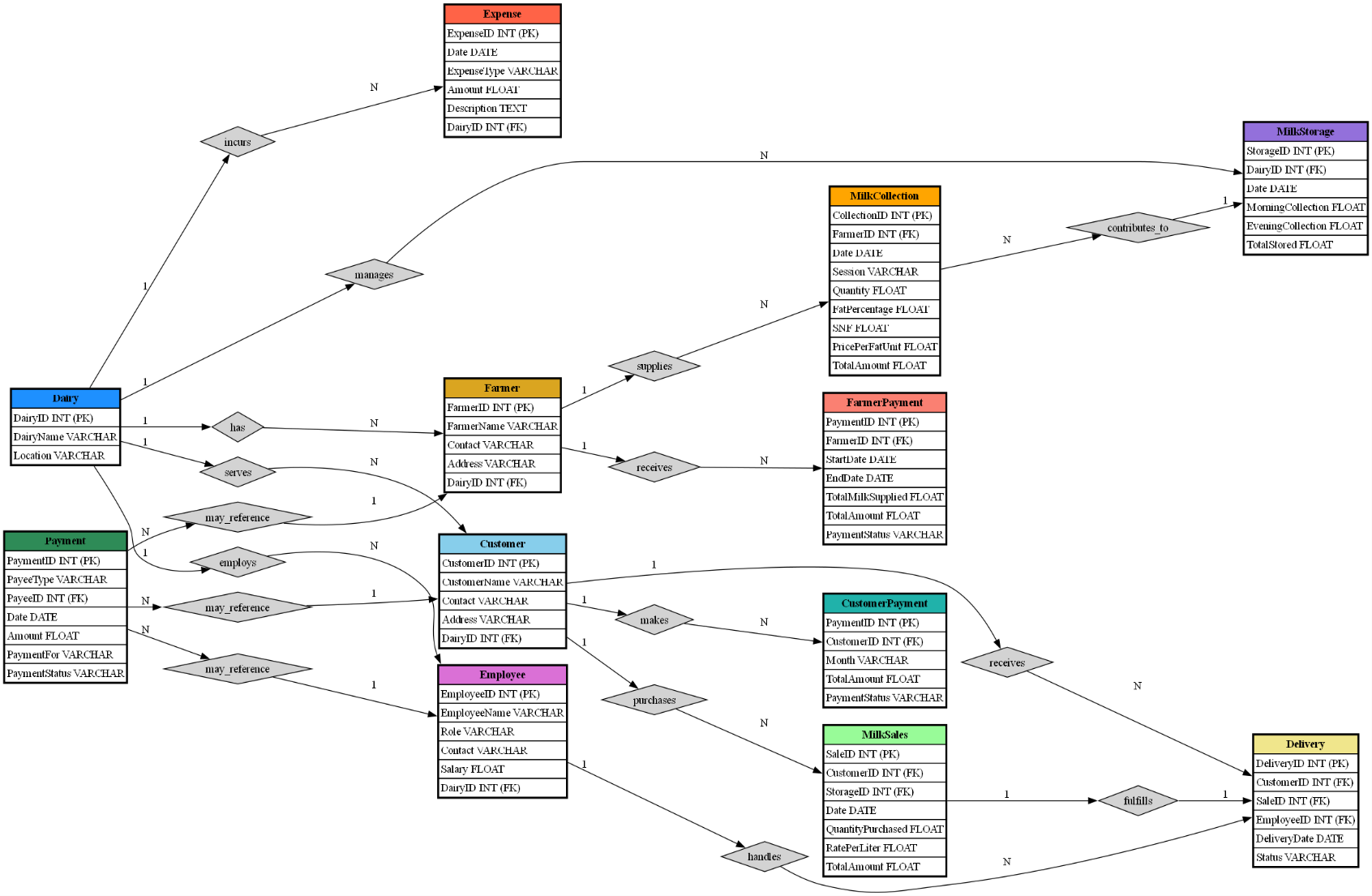
 **Expense**  
Tracks daily expenses incurred by the dairy.  
Includes type, amount, and description.

 **Payment**  
A unified table to log payments to farmers, customers (refunds), and employees.  
Uses PayeeType to distinguish the recipient type.

**Relationship:-**

* **Dairy <> Farmer** – (Has8) (1:many)
* **Dairy <> MilkStorage** – (Stores Collection) (1:many)
* **Dairy <> Customer** – (Serves) (1:many)
* **Dairy <> Employee** – (Employs) (1:many)
* **Dairy <> Expense** – (Incurs) (1:many)
* **Farmer <> MilkCollection** – (Supplies) (1:many)
* **Farmer <> FarmerPayment** – (Gets Paid Through) (1:many)
* **MilkStorage <> MilkSales** – (Provides Milk For) (1:many)
* **Customer <> MilkSales** – (Buys From) (1:many)
* **Customer <> CustomerPayment** – (Pays Through) (1:many)
* **Customer <> Delivery** – (Receives) (1:many)
* **MilkSales <> Delivery** – (Delivered Through) (1:many)
* **Employee <> Delivery** – (Handles) (1:many)
* **Payment <> Farmer / Customer / Employee** – (Tracks Payments To) (many:1 via PayeeID + PayeeType logic)

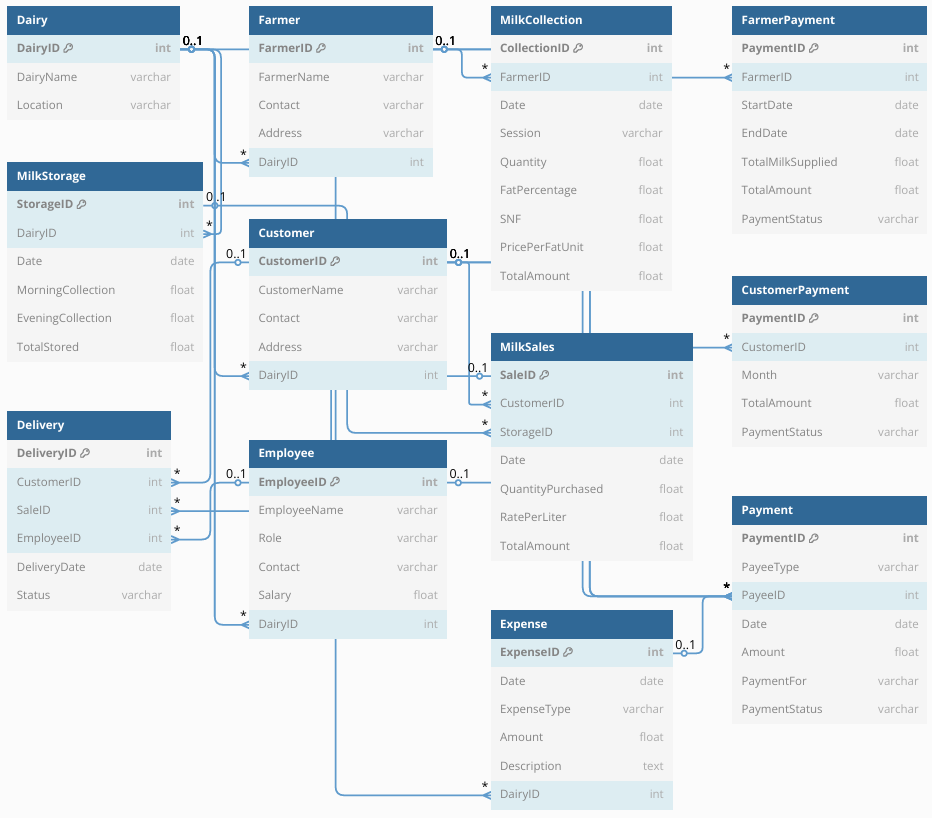
**III. Entity Relationship Diagram**



**IV. Relational Model**

|  |  |
| --- | --- |
| 1. **Farmer**  * FarmerID (Primary Key) * FarmerName * Contact * Address | 1. **MilkCollection**  * CollectionID (Primary Key) * FarmerID (Foreign Key) * Date * Session * Quantity * FatPercentage * SNF * PricePerFatUnit |
| 1. **FarmerPayment**  * PaymentID (Primary Key) * FarmerID (Foreign Key) * StartDate * EndDate * TotalMilkSupplied * PaymentStatus | 1. **MilkStorage**  * StorageID (Primary Key) * Date * MorningCollection * EveningCollection * TotalStored |

|  |  |
| --- | --- |
| 1. **Customer**  * CustomerID (Primary Key) * CustomerName * Contact * Address | 1. **MilkSales**  * SaleID (Primary Key) * CustomerID (Foreign Key) * Date * QuantityPurchased * RatePerLiter * TotalAmount |
| 1. **CustomerPayment**  * PaymentID (Primary Key) * CustomerID (Foreign Key) * Month * TotalAmount * PaymentStatus | 1. **Delivery**  * DeliveryID (Primary Key) * CustomerID (Foreign Key) * SaleID (Foreign Key) * DeliveryDate * Status * EmployeeID (planned, Foreign Key) (future implementation as per diagram notation) |
| 1. **Employee**  * EmployeeID (Primary Key) * EmployeeName * Role * Contact * Salary | 1. **Expense**  * ExpenseID (Primary Key) * Date * ExpenseType * Amount * Description |
| 1. **Payment**  * PaymentID – INT * PayeeType – ENUM('Farmer', 'Customer', 'Employee') * PayeeID – INT * Date – DATE * Amount – FLOAT * PaymentFor – VARCHAR(50) * PaymentStatus – ENUM('Paid', 'Pending') | 1. **Dairy**  * DairyID – INT * DairyName – VARCHAR(100) * Location – VARCHAR(255) * Contact – VARCHAR(15) |



**V. Normalization**

Perform normalization (1NF, 2NF, 3NF, BCNF) as applicable for the entire database.

**GENERAL RULES OF NORMALIZATION:**

| **Normal Form** | **Condition** |
| --- | --- |
| **1NF** | Atomic values only, no repeating groups |
| **2NF** | Already in 1NF + no partial dependencies (applies to composite keys only) |
| **3NF** | Already in 2NF + no transitive dependencies |
| **BCNF** | Already in 3NF + every determinant is a candidate key |

**1. Farmer**  
Farmer(FarmerID, FarmerName, Contact, Address)

* 1NF: Atomic values, no repeating groups – Satisfied
* 2NF: Only single primary key (FarmerID), no partial dependency – Satisfied
* 3NF: No transitive dependency – Satisfied
* BCNF: All determinants are candidate keys – Satisfied

**Already in BCNF**

**2. MilkCollection**  
MilkCollection(CollectionID, FarmerID, Date, Session, Quantity, FatPercentage, SNF, PricePerFatUnit)

* 1NF: Atomic values – Satisfied
* 2NF: Single primary key, no partial dependency – Satisfied
* 3NF: No transitive dependency – Satisfied
* BCNF: All determinants are candidate keys – Satisfied

**Already in BCNF**

**3. FarmerPayment**  
FarmerPayment(PaymentID, FarmerID, StartDate, EndDate, TotalMilkSupplied, PaymentStatus)

* 1NF: Atomic values – Satisfied
* 2NF: Single primary key, no partial dependency – Satisfied
* 3NF: No transitive dependency – Satisfied
* BCNF: All determinants are candidate keys – Satisfied

**Already in BCNF**

**4. MilkStorage**  
MilkStorage(StorageID, Date, MorningCollection, EveningCollection, TotalStored)

* 1NF: Atomic values – Satisfied
* 2NF: Single primary key – Satisfied
* 3NF: TotalStored is functionally dependent on MorningCollection and EveningCollection – violates 3NF
* BCNF: Not satisfied due to derived attribute

**Normalize into:**  
MilkStorage(StorageID, Date, MorningCollection, EveningCollection)  
StorageTotal(Date, TotalStored)

**5. Customer**  
Customer(CustomerID, CustomerName, Contact, Address)

* 1NF: Atomic values – Satisfied
* 2NF: Single primary key – Satisfied
* 3NF: No transitive dependency – Satisfied
* BCNF: All determinants are candidate keys – Satisfied

**Already in BCNF**

**6. MilkSales**  
MilkSales(SaleID, CustomerID, Date, QuantityPurchased, RatePerLiter, TotalAmount)

* 1NF: Atomic values – Satisfied
* 2NF: Single primary key – Satisfied
* 3NF: TotalAmount = QuantityPurchased × RatePerLiter – violates 3NF
* BCNF: Not satisfied due to derived attribute

**Normalize into:**  
MilkSales(SaleID, CustomerID, Date, QuantityPurchased, RatePerLiter)  
SalesAmount(SaleID, TotalAmount)

**7. CustomerPayment**  
CustomerPayment(PaymentID, CustomerID, Month, TotalAmount, PaymentStatus)

* 1NF: Atomic values – Satisfied
* 2NF: Single primary key – Satisfied
* 3NF: No transitive dependency – Satisfied
* BCNF: All determinants are candidate keys – Satisfied

**Already in BCNF**

**8. Delivery**  
Delivery(DeliveryID, CustomerID, SaleID, DeliveryDate, Status, EmployeeID)

* 1NF: Atomic values – Satisfied
* 2NF: Single primary key – Satisfied
* 3NF: No transitive dependency – Satisfied
* BCNF: All determinants are candidate keys – Satisfied

**Already in BCNF**

**9. Employee**  
Employee(EmployeeID, EmployeeName, Role, Contact, Salary)

* 1NF: Atomic values – Satisfied
* 2NF: Single primary key – Satisfied
* 3NF: No transitive dependency – Satisfied
* BCNF: All determinants are candidate keys – Satisfied

**Already in BCNF**

**10. Expense**  
Expense(ExpenseID, Date, ExpenseType, Amount, Description)

* 1NF: Atomic values – Satisfied
* 2NF: Single primary key – Satisfied
* 3NF: No transitive dependency – Satisfied
* BCNF: All determinants are candidate keys – Satisfied

**Already in BCNF**

**11. Payment**   
Payment(PaymentID, PayeeType, PayeeID, Date, Amount, PaymentFor, PaymentStatus)

* 1NF: Atomic values – Satisfied
* 2NF: Single primary key – Satisfied
* 3NF: PaymentFor may be context-dependent on PayeeType – Possible transitive dependency
* BCNF: If PaymentFor depends on PayeeType, then violates BCNF

**Normalize if needed into:**  
PaymentBasic(PaymentID, PayeeType, PayeeID, Date, Amount, PaymentStatus)  
PaymentDetail(PaymentID, PaymentFor)

**12. Dairy**  
Dairy(DairyID, DairyName, Location, Contact)

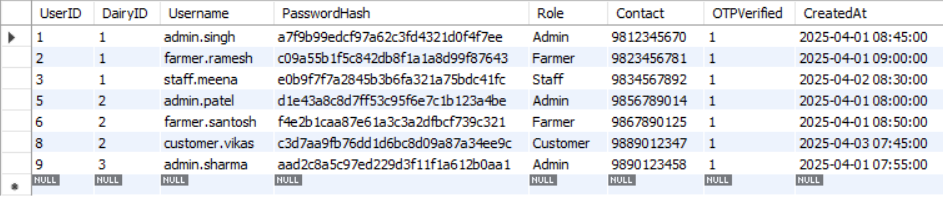
* 1NF: Atomic values – Satisfied
* 2NF: Single primary key – Satisfied
* 3NF: No transitive dependency – Satisfied
* BCNF: All determinants are candidate keys – Satisfied

**Already in BCNF**

**VI. SQL Queries**

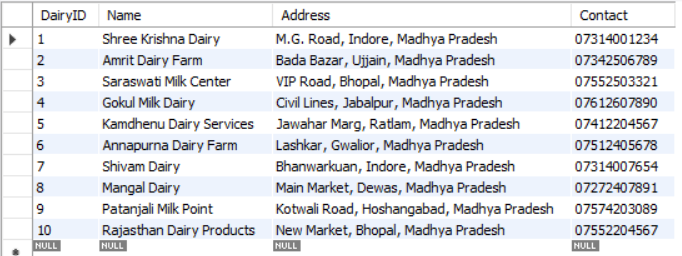
1. Get all users who are verified by OTP

SELECT \* FROM Users WHERE OTPVerified = TRUE;



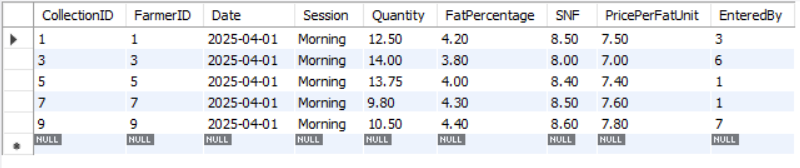
2.List all dairies in Madhya Pradesh:

SELECT \* FROM Dairies WHERE Address LIKE '%Madhya Pradesh%';



3.Show all milk collections done in the morning:

SELECT \* FROM MilkCollections WHERE Session = 'Morning';



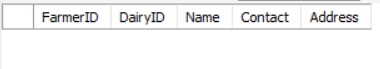
4.Find all farmers associated with "Shree Dairy - Indore":

SELECT f.\*

FROM Farmers f

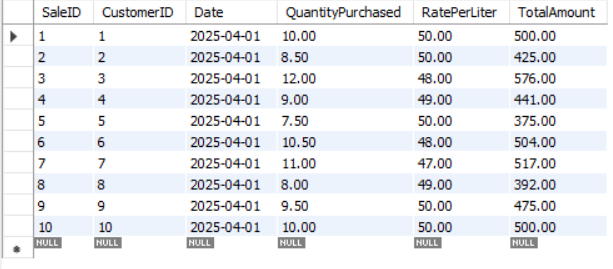
JOIN Dairies d ON f.DairyID = d.DairyID

WHERE d.Name = 'Shree Dairy - Indore';



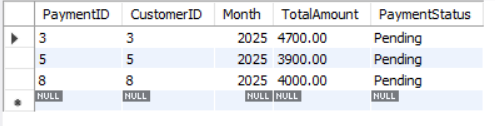
5. Retrieve milk sales over ₹100:

SELECT \* FROM MilkSales WHERE TotalAmount > 100;



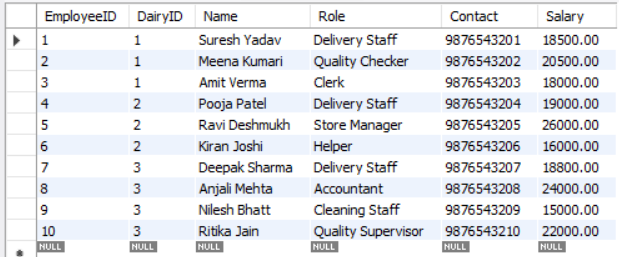
6. Get customers who haven’t paid yet:

SELECT \* FROM CustomerPayments WHERE PaymentStatus = 'Pending';



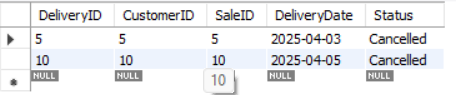
7. Get all employees with salary over ₹12,000:

SELECT \* FROM Employees WHERE Salary > 12000;



8. List milk deliveries that were cancelled:

SELECT \* FROM MilkDeliveries WHERE Status = 'Cancelled';

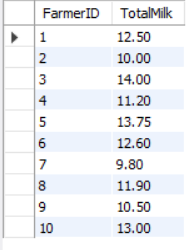


9. Total milk collected per farmer:

SELECT FarmerID, SUM(Quantity) AS TotalMilk

FROM MilkCollections

GROUP BY FarmerID;

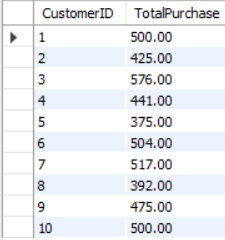


10. Total sales per customer:

SELECT CustomerID, SUM(TotalAmount) AS TotalPurchase

FROM MilkSales

GROUP BY CustomerID;

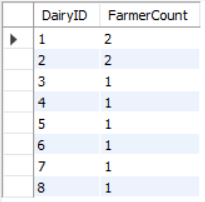


11. Number of farmers per dairy:

SELECT DairyID, COUNT(\*) AS FarmerCount

FROM Farmers

GROUP BY DairyID;

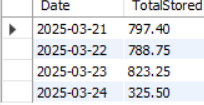


12. Daily milk storage summary:

SELECT Date, SUM(TotalStored) AS TotalStored

FROM MilkStorage

GROUP BY Date;



1. Top 3 farmers by total milk supplied:

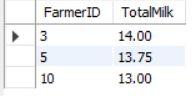
SELECT FarmerID, SUM(Quantity) AS TotalMilk

FROM MilkCollections

GROUP BY FarmerID

ORDER BY TotalMilk DESC

LIMIT 3;

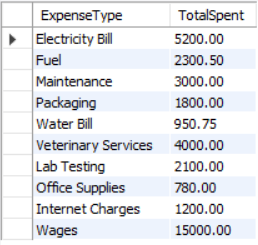


1. Expenses grouped by type:

SELECT ExpenseType, SUM(Amount) AS TotalSpent

FROM Expenses

GROUP BY ExpenseType;



1. Monthly income from customers:

SELECT Month, SUM(TotalAmount) AS Income

FROM CustomerPayments

GROUP BY Month;

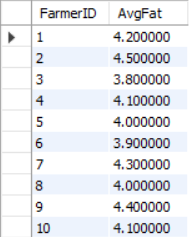


1. Farmer-wise average fat percentage:

SELECT FarmerID, AVG(FatPercentage) AS AvgFat

FROM MilkCollections

GROUP BY FarmerID;

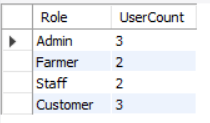


1. Count of users by role:

SELECT Role, COUNT(\*) AS UserCount

FROM Users

GROUP BY Role;

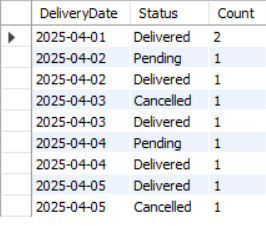


18.Daily milk delivery status count:

SELECT DeliveryDate, Status, COUNT(\*) AS Count

FROM MilkDeliveries

GROUP BY DeliveryDate, Status;

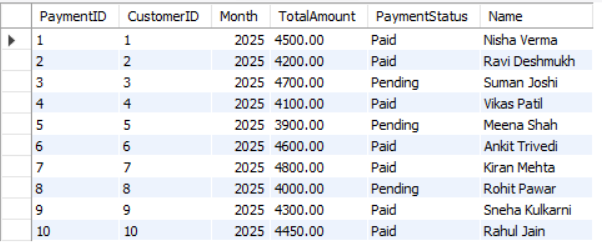


18.View customer payments with customer names:

SELECT cp.\*, c.Name

FROM CustomerPayments cp

JOIN Customers c ON cp.CustomerID = c.CustomerID;

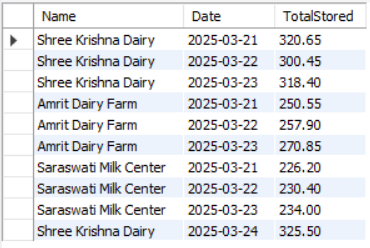


19. Total milk storage and dairy name:

SELECT d.Name, s.Date, s.TotalStored

FROM MilkStorage s

JOIN Dairies d ON s.DairyID = d.DairyID;



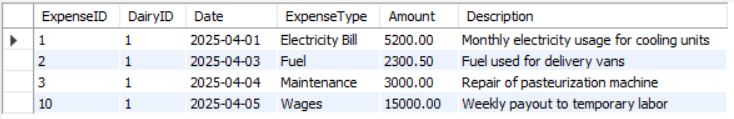
20. Expenses for 'Shree Krishna Dairy':

SELECT e.\*

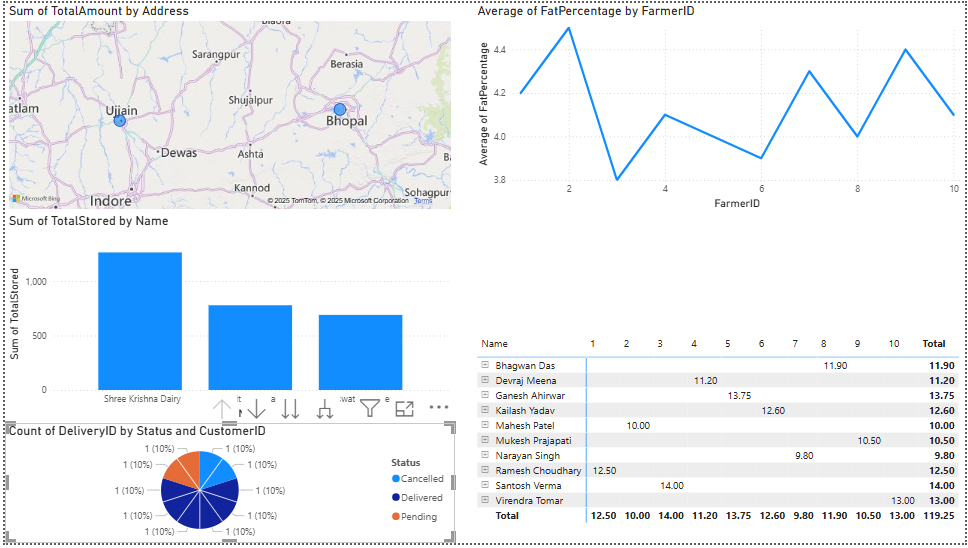
FROM Expenses e

JOIN Dairies d ON e.DairyID = d.DairyID

WHERE d.Name = 'Shree Krishna Dairy';



**VI. Project demonstration**



**VII. Self -Learning beyond classroom**

While working together on the Shree Dairy Management Database System, we collectively explored several advanced concepts and real-world applications that extended far beyond our classroom learning:

* **Real-World Data Modeling:** We gained a deeper understanding of how actual business operations—like milk collection, logistics, and billing—can be represented in database design. Together, we applied normalization techniques and carefully established relationships using foreign keys to maintain integrity across interconnected tables.
* **ENUMs and Business Rule Implementation:** As a team, we used ENUM data types to manage values like milk collection sessions (Morning/Evening) and payment statuses (Paid/Pending). This gave us practical insight into embedding real business logic directly into our schema.
* **Unified Payment System Design:** Designing a single payment table to manage multiple payee types (Farmers, Customers, Employees) introduced us to polymorphic relationships—something we had only briefly encountered in theory. Working through it together enhanced our collective problem-solving skills.
* **Efficient Table Structuring:** We collaborated on optimizing table structures by minimizing redundancy and following naming conventions. This helped ensure that the database was both scalable and easy to understand.
* **Advanced SQL Practice:** Throughout the project, we honed our SQL skills by using features like AUTO\_INCREMENT, ENUM, and FOREIGN KEY constraints. This hands-on practice gave us the confidence to independently design and build complex, relational databases.

**VIII. Learning from the Project**

This project offered us a holistic learning experience, especially in understanding how technology empowers traditional sectors like dairy farming. As a group, we took away several key lessons:

* **Comprehensive System Thinking:** We were able to visualize and implement an entire operational flow—from milk collection to final payments and deliveries. It taught us how small modules come together to create a unified system.
* **Focus on Data Accuracy:** Knowing the system handled financial transactions and inventory made us more attentive to the importance of accuracy. We became more deliberate in setting constraints, choosing field types, and designing validation logic.
* **Translating Real-Life Scenarios into Data Models:** We developed the ability to analyze and break down business activities into tables, fields, and relationships. This was one of the most insightful parts of the project for us.
* **Business-Oriented Thinking:** We started thinking like business analysts—understanding stakeholders’ needs and ensuring our system provided value through clarity, accountability, and usability.
* **Communication & Documentation:** Throughout the process, we learned how to divide tasks, document our progress, and present technical ideas in a way that was accessible to both tech-savvy and non-technical audiences.

**IX. Challenges Faced**

As a team working on the Shree Dairy Management Database System, we faced several challenges that required collaboration, critical thinking, and adaptability:

* **Designing Complex Relationships:** Structuring relationships between multiple entities such as Farmers, Customers, Employees, and various Payment types was one of the most demanding parts. We had to brainstorm and refine our ER diagrams multiple times to ensure all real-world operations were accurately represented.
* **Implementing a Unified Payment System:** Managing payments for different roles (farmers, customers, employees) in a single table was conceptually challenging. We worked together to implement a polymorphic-style structure using ENUMs and IDs effectively.
* **Data Consistency Across Modules:** Ensuring consistent and accurate data—especially in financial records tied to milk collection, storage, and sales—required careful testing and validation. It pushed us to focus on foreign key constraints, data types, and logical connections.
* **Mapping Real-World Operations:** Translating real dairy operations like morning/evening milk collection, door-to-door delivery, and monthly billing into a normalized database model was not easy. We held discussions to align technical design with operational workflows.
* **Collaborative Integration:** Coordinating different parts of the database development (ERD design, SQL scripts, relationship mappings, documentation) among group members taught us the importance of task distribution and communication.

**X. Conclusion**

This project was a significant learning journey for our group. We successfully collaborated to design and implement a robust database system that could support all operational aspects of a dairy business—from collection and storage to delivery and payments.

The process helped us apply theoretical concepts in a real-world scenario, enhanced our understanding of relational databases, and strengthened our teamwork and project management skills. By addressing practical challenges and ensuring that every detail—from data entry to billing—was accounted for, we’ve built a strong foundation in database development.

We are proud of what we accomplished as a group and are confident that this experience will help us tackle more complex systems and collaborative projects in the future.