Final Report: Inventory Management System for a Local Farmer's Market

Group No: 1
Course: CSCI 4710 - Databases
Group Members: Jeet R Patel, Shubh Vivek Patel, Omkar Datta, Jagruth Reddy, Bhanu Prakash Reddy, Manasa Tallapaka
Date: November 24, 2024

Abstract:

This report presents the design, development, and deployment of a robust **Inventory Management System** tailored to the needs of vendors at a local farmer's market. The system was created to simplify inventory tracking, optimize stock levels, monitor sales, and analyze seasonal trends. It provides real-time data updates and actionable insights to help vendors improve efficiency and customer satisfaction.

Problem Statement:

The **Inventory Management System** is designed to address the challenges faced by vendors in managing stock levels, sales, and seasonal trends at a farmer's market. Vendors often struggle with overstocking, stockouts, and tracking customer preferences. The system aims to provide a centralized solution for inventory management, sales tracking, and seasonal demand forecasting, improving vendor efficiency and customer satisfaction.

Requirements Gathering:

The system includes the following functional requirements:

1. **Vendor and Product Management**:
    - Maintain vendor profiles (e.g., name, contact information, stall location).
    - Manage product details such as name, price, category, and seasonal availability.

2. **Inventory Tracking**:
    - Monitor stock levels for each vendor's products.
    - Set restocking thresholds to alert vendors when inventory is low.
3. **Sales Tracking**:
    - Record sales transactions, including products sold, quantities, and total revenue.
    - Link sales to specific vendors and customers.
4. **Seasonal Analysis**:
    - Analyze past sales data to predict seasonal demand.
    - Provide actionable insights for stock optimization.
5. **Customer Preferences**:
    - Maintain customer profiles to track preferences and purchase history.
6. **Data Reporting**:
    - Generate reports summarizing vendor performance, best-selling products, and seasonal trends.

Introduction:

Managing inventory in a farmer's market is a challenging task due to the variety of products, seasonal demands, and customer preferences. Vendors often face overstocking or stockouts, resulting in wasted resources or missed sales opportunities. A comprehensive inventory management system is critical to addressing these challenges.

Objectives:

- Develop a centralized database system for tracking inventories, sales, and vendor information.
- Incorporate real-time updates for product availability, vendor sales, and customer preferences.
- Leverage historical sales data to predict demand trends and ensure optimal stock levels.
- Enhance vendor-customer relationships by providing insights into purchasing trends.

Methodology-Project Scope:

The project involved designing and deploying a relational database to manage:

1. Vendor profiles and contact information.
2. Product details and inventory levels.
3. Customer purchase history and preferences.
4. Sales transactions and seasonal data analysis.

Tools and Technologies:

- **SQL**: For database creation and management.
- **MySQL/PostgreSQL**: DBMS for implementation.
- **ERD Tools**: MySQL Workbench for ERD visualization.
- **Programming Languages**: Python Programming.

Steps for Implementation:

- **Database Design**: A normalized schema was designed using primary and foreign keys for relational integrity.
- **CRUD Operations**: Developed to manage product, vendor, and sales data.
- **Seasonal Data Integration**: Algorithms implemented to analyze and predict demand.
- **Security**: Ensured data validation and secure access.
- **Testing**: Conducted unit testing and stress testing with mock data.

System Design:

Entity-Relationship Diagram (ERD): The ERD represents the relationships between key entities in the system, including Vendor, Product, Inventory, Sale, Customer, and Seasonal Analysis.

The ERD captured **relationships** between **key entities**:

- **Vendor** ↔ **Product** ↔ **Inventory**
- **Product** ↔ **Sale** ↔ **Customer**
- **Product** ↔ **Seasonal Analysis**

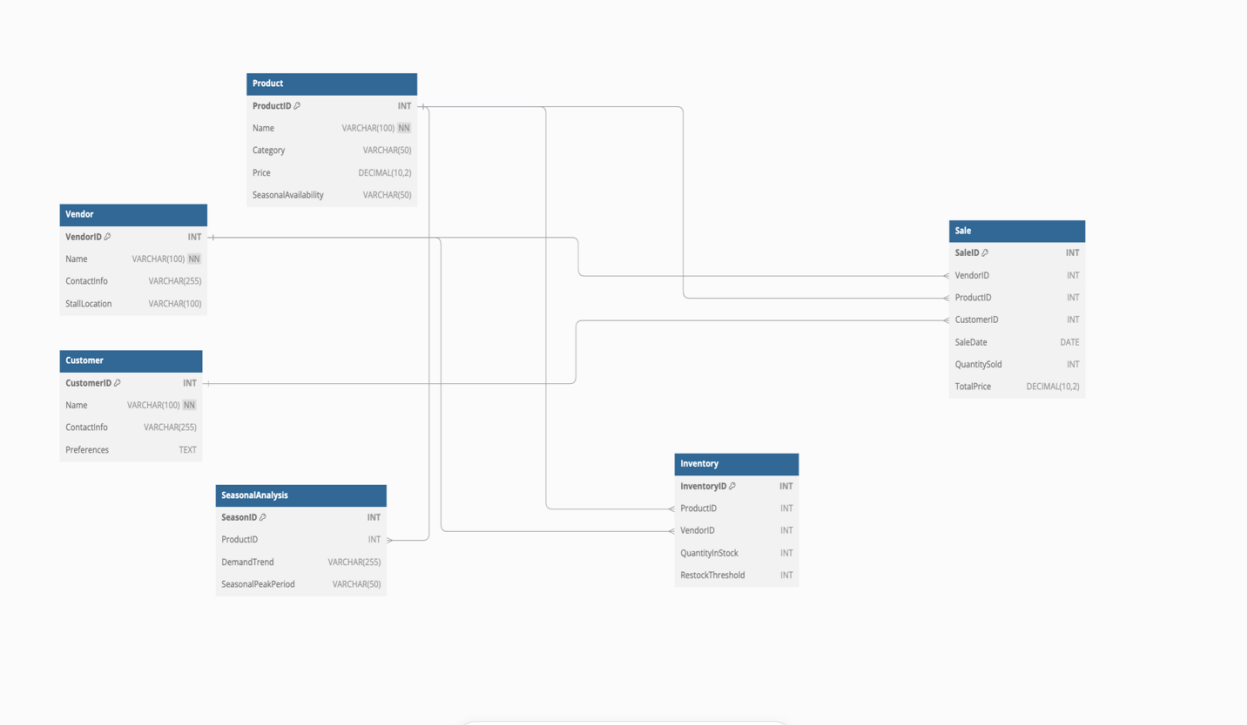
Database Schema:

- Vendor: Stores vendor profiles.
- Product: Maintains product details.
- Inventory: Tracks stock levels and restocking needs.
- Sale: Logs sales transactions.
- Customer: Records customer preferences.
- Seasonal Analysis: Tracks seasonal demand trends.

Relationships:

- Vendors manage multiple products (1: N).
- Products are linked to inventory and sales (N:M).
- Seasonal data enhances forecasting for products.

ERD Diagram:



ERD Explanation:

Entities and Attributes

1. **Vendor**:
    - **Attributes**:
        - VendorID: Primary Key, uniquely identifies a vendor.
        - Name: The vendor's name.
        - ContactInfo: Vendor's contact information, such as phone or email.
        - StallLocation: The location of the vendor's stall in the farmer's market.
    - **Purpose**: This table stores details about all vendors participating in the market.

2. **Product**:
    - **Attributes**:
        - ProductID: Primary Key, uniquely identifies a product.
        - Name: The name of the product (e.g., "Tomato", "Apple").
        - Category: The product category (e.g., "Vegetable", "Fruit").
        - Price: The price per unit of the product.
        - SeasonalAvailability: Indicates the seasons when the product is available (e.g., "Summer").
    - **Purpose**: This table captures information about the products sold by vendors.

3. **Inventory**:
    - **Attributes**:
        - InventoryID: Primary Key, uniquely identifies an inventory record.
        - ProductID: Foreign Key referencing Product.ProductID, links the inventory to a specific product.
        - VendorID: Foreign Key referencing Vendor.VendorID, links the inventory to a specific vendor.
        - QuantityInStock: The current quantity of the product in stock.
        - RestockThreshold: The minimum stock level that triggers restocking.
    - **Purpose**: This table manages the stock levels of products for each vendor, enabling real-time inventory tracking.
4. **Sale**:
    - **Attributes**:
        - SaleID: Primary Key, uniquely identifies a sales transaction.
        - VendorID: Foreign Key referencing Vendor.VendorID, links the sale to a specific vendor.
        - ProductID: Foreign Key referencing Product.ProductID, links the sale to a specific product.

        - CustomerID: Foreign Key referencing Customer.CustomerID, links the sale to a specific customer.
        - SaleDate: The date of the sale.
        - QuantitySold: The quantity of the product sold in the transaction.
        - TotalPrice: The total price for the transaction (calculated as QuantitySold * Product.Price).
    - **Purpose**: This table records all sales transactions for analysis of vendor performance and product popularity.

5. **Customer**:
    - **Attributes**:
        - CustomerID: Primary Key, uniquely identifies a customer.
        - Name: The customer's name.
        - ContactInfo: The customer's contact details (e.g., phone or email).
        - Preferences: Notes or preferences about the customer's purchasing habits.
    - **Purpose**: This table tracks customer information, enabling personalized insights and trend analysis.

6. **SeasonalAnalysis**:
    - **Attributes**:
        - SeasonID: Primary Key, uniquely identifies a seasonal analysis record.
        - ProductID: Foreign Key referencing Product.ProductID, links the analysis to a specific product.
        - DemandTrend: Describes the demand trend for the product (e.g., "High", "Low").
        - SeasonalPeakPeriod: Indicates the period of peak demand for the product (e.g., "June-August").
    - **Purpose**: This table helps vendors predict product demand based on historical data and seasonal trends.

Relationships:

1. **Vendor – Product (via Inventory)**:
    - A vendor can sell multiple products, and a product can be sold by multiple vendors. This is modeled through the **Inventory** table, which serves as a junction table in this many-to-many relationship.

    - **Key Relationship**: Vendor.VendorID ↔ Inventory.VendorID ↔ Inventory.ProductID ↔ Product.ProductID.

2. **Vendor – Sale**:
    - A vendor can have multiple sales, but each sale is associated with only one vendor.
    - **Key Relationship**: Vendor.VendorID ↔ Sale.VendorID.

3. **Product – Sale**:
    - A product can appear in multiple sales, and each sale includes one product.
    - **Key Relationship**: Product.ProductID ↔ Sale.ProductID.

4. **Customer – Sale**:
    - A customer can make multiple purchases (sales), but each sale is associated with one customer.
    - **Key Relationship**: Customer.CustomerID ↔ Sale.CustomerID.

5. **Product – SeasonalAnalysis**:
    - Each product can have multiple seasonal analyses, but each seasonal analysis is specific to one product.
    - Key Rel: Product.ProductID ↔ SeasonalAnalysis.ProductID.

Key Features Highlighted in the ERD:

- **Centralized Data**: The ERD shows how data is centralized around vendors, products, and sales, ensuring efficient management.

- **Scalable Design**: By using normalized tables and relationships, the system is scalable for handling additional vendors, products, and customers.

- **Predictive Analysis**: The inclusion of the Seasonal Analysis table demonstrates how the system leverages data for demand forecasting.

- **Real-Time Tracking**: The Inventory table ensures vendors can monitor and manage stock levels effectively.

Summary of ERD:

The ERD for the Inventory Management System effectively models the entities, and their relationships required for real-time inventory tracking, sales monitoring, and seasonal trend analysis. By capturing and linking vendor, product, and sales data, the system provides a comprehensive solution to optimize stock levels, enhance customer satisfaction, and improve vendor performance.

Normalization:

1. **First Normal Form (1NF)**:
    - Ensures atomicity of attributes.
    - Removed repeating groups by splitting composite attributes into individual columns.
2. **Second Normal Form (2NF)**:
    - Eliminated partial dependencies by separating data into related tables.
3. **Third Normal Form (3NF)**:
    - Removed transitive dependencies by ensuring all non-key attributes depend only on the primary key.

Implementation:

Database Table Creation

1. **Vendor Table:**

```
CREATE TABLE Vendor (
VendorID INT PRIMARY KEY,
Name VARCHAR (100),
Contact Info VARCHAR (255),
Stall Location VARCHAR (100));
```

2. **Product Table:**

```
CREATE TABLE Product (
  ProductID INT AUTO_INCREMENT PRIMARY KEY,
  Name VARCHAR (100) NOT NULL,
  Category VARCHAR (50),
  Price DECIMAL (10, 2),
  Seasonal Availability VARCHAR (50));
```

3. **Inventory Table:**

```
CREATE TABLE Inventory (
  InventoryID INT AUTO_INCREMENT PRIMARY KEY,
  ProductID INT,
  VendorID INT,
  QuantityInStock INT,
  RestockThreshold INT,
  FOREIGN KEY (ProductID) REFERENCES Product (ProductID),
  FOREIGN KEY (VendorID) REFERENCES Vendor (VendorID));
```

4. **Sale Table:**

```
    CREATE TABLE Sale (
      SaleID INT AUTO_INCREMENT PRIMARY KEY,
      VendorID INT,
      ProductID INT,
      CustomerID INT,
      SaleDate DATE,
      QuantitySold INT,
      TotalPrice DECIMAL(10, 2),
      FOREIGN KEY (VendorID) REFERENCES Vendor(VendorID),
      FOREIGN KEY (ProductID) REFERENCES Product(ProductID),
      FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
);
```

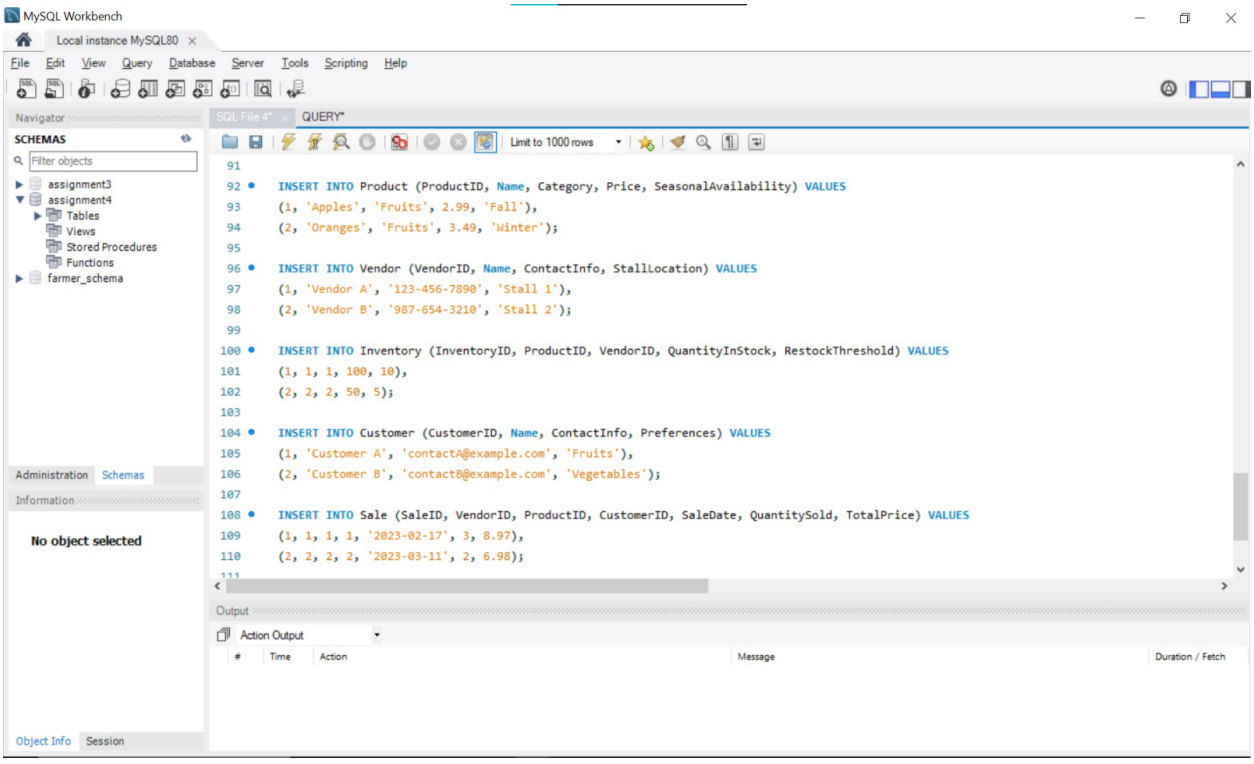
5. **Customer Table:**

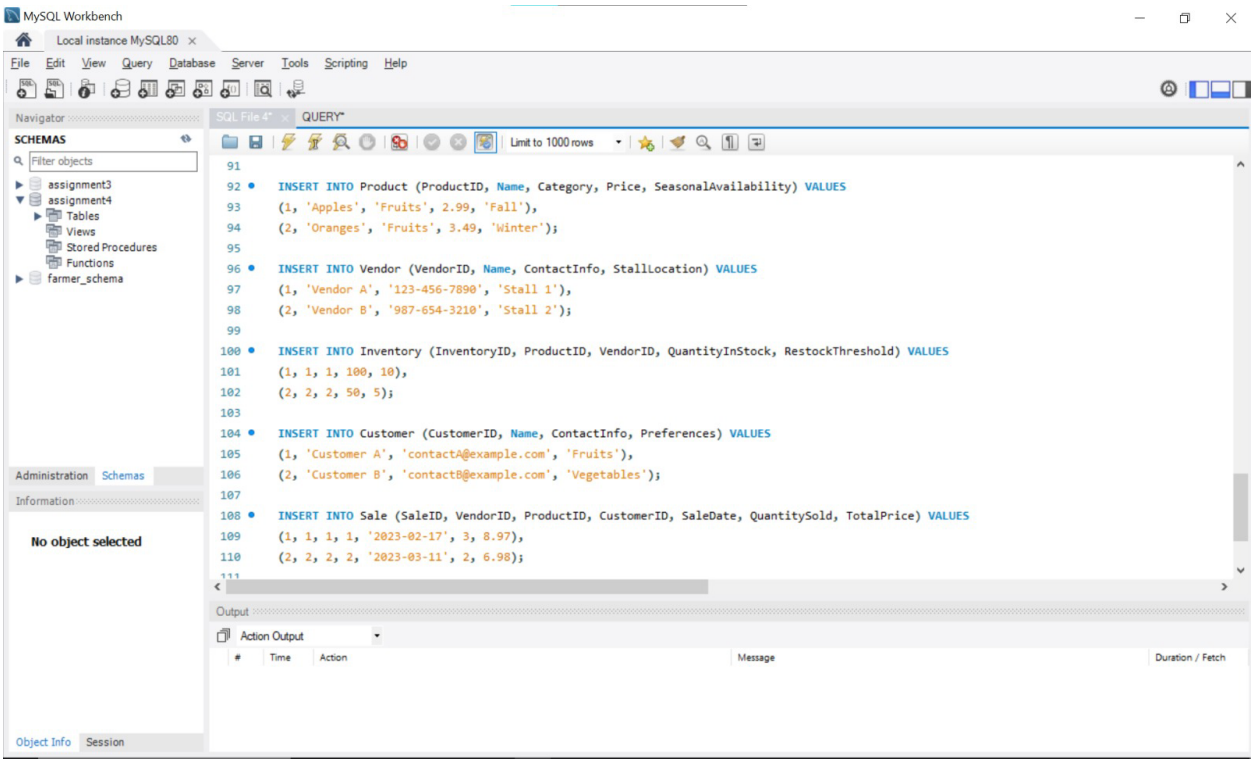
```
CREATE TABLE Customer (
   CustomerID INT AUTO_INCREMENT PRIMARY KEY,
   Name VARCHAR(100),
   ContactInfo VARCHAR(255),
   Preferences TEXT
);
```

6. **Seasonal Analysis Table:**

```
CREATE TABLE SeasonalAnalysis (
   SeasonID INT AUTO_INCREMENT PRIMARY KEY,
   ProductID INT,
   DemandTrend VARCHAR(255),
   SeasonalPeakPeriod VARCHAR(50),
   FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
);
```

Insertion of Data: Evidence





Summary of Primary Keys:

The **Primary Key (PK)** uniquely identifies each record in a table. Below are the primary keys for all tables in the system:

1. Vendor Table:
    - VendorID: Unique identifier for each vendor.
2. Product Table:
    - ProductID: Unique identifier for each product.
3. Inventory Table:
    - InventoryID: Unique identifier for each inventory record.
4. Sale Table:
    - SaleID: Unique identifier for each sales transaction.
5. Customer Table:
    - CustomerID: Unique identifier for each customer.
6. Seasonal Analysis Table:
    - SeasonID: Unique identifier for each seasonal analysis record.

Summary of Foreign Keys:

The **Foreign Key** establishes relationships between tables, ensuring referential integrity. Below are the foreign keys and their roles:

1. Inventory Table:
    - ProductID: References Product (ProductID) to associate inventory with specific products.
    - VendorID: References Vendor (VendorID) to associate inventory with specific vendors.
2. Sale Table:
    - VendorID: References Vendor (VendorID) to link sales to specific vendors.
    - ProductID: References Product (ProductID) to link sales to specific products.
    - CustomerID: References Customer (CustomerID) to link sales to specific customers.
3. Seasonal Analysis Table:
    - ProductID: References Product(ProductID) to link seasonal analysis to specific products.

How They Work Together:

- **Primary Keys** uniquely identify records in each table.
- **Foreign Keys** link these records across tables, ensuring consistent relationships and preventing orphaned data (e.g., a sale without a corresponding vendor or product).
- **Together**, they enable efficient querying and data integrity within the database. For example:
    - A query to fetch all sales of a product can use Sale.ProductID linked to Product.ProductID.

- An inventory record is associated with a specific vendor and product through Inventory.VendorID and Inventory.ProductID.

3. Implementation (30 points)
SQL Queries:

- Overview: The implementation involves designing SQL queries to handle data operations effectively. These include creating tables, inserting new records, updating existing data, retrieving information, and deleting records as necessary.

- Purpose: The queries ensure data consistency, support project functionalities, and demonstrate how the database interacts with the application.

- Focus Areas: Queries address CRUD (Create, Read, Update, Delete) operations to manage vaccination records, patient information, insurance details, and provider relationships.

CRUD Operations:

- **Create**: Insert new vendors, products, and sales records.
- **Read**: Retrieve inventory status, vendor performance, and customer preferences.
- **Update**: Modify stock levels or vendor information.
- **Delete**: Remove obsolete records.

Testing and Validation:
Mock Data

- Representative data was used to test the database functionality:
  - Vendor records with realistic names, contact information, and stall locations.
  - Product details with varying prices, categories, and seasonal availabilities.
  - Sales transactions linked to specific vendors, products, and customers.

Validation:

1. **Scenarios Tested**:
    - Valid operations: Adding, updating, retrieving, and deleting data across all tables.
    - Edge cases: Zero or negative stock levels, invalid dates, missing mandatory fields.
    - Constraint violations: Attempting to insert data that violates foreign key or check constraints.
2. **Cascading Updates and Deletions**:
    - Verified that:
        - Deleting a product automatically removed related inventory records without affecting unrelated data.
        - Updating VendorID in the **Vendor** table reflected changes in linked inventory and sales records.
3. **Query Testing**:
    - Ensured all SQL queries produced accurate and expected outputs:
        - Retrieval of top-selling products.
        - Identification of low-stock items triggering restock alerts.
        - Seasonal trend analysis based on historical sales data.

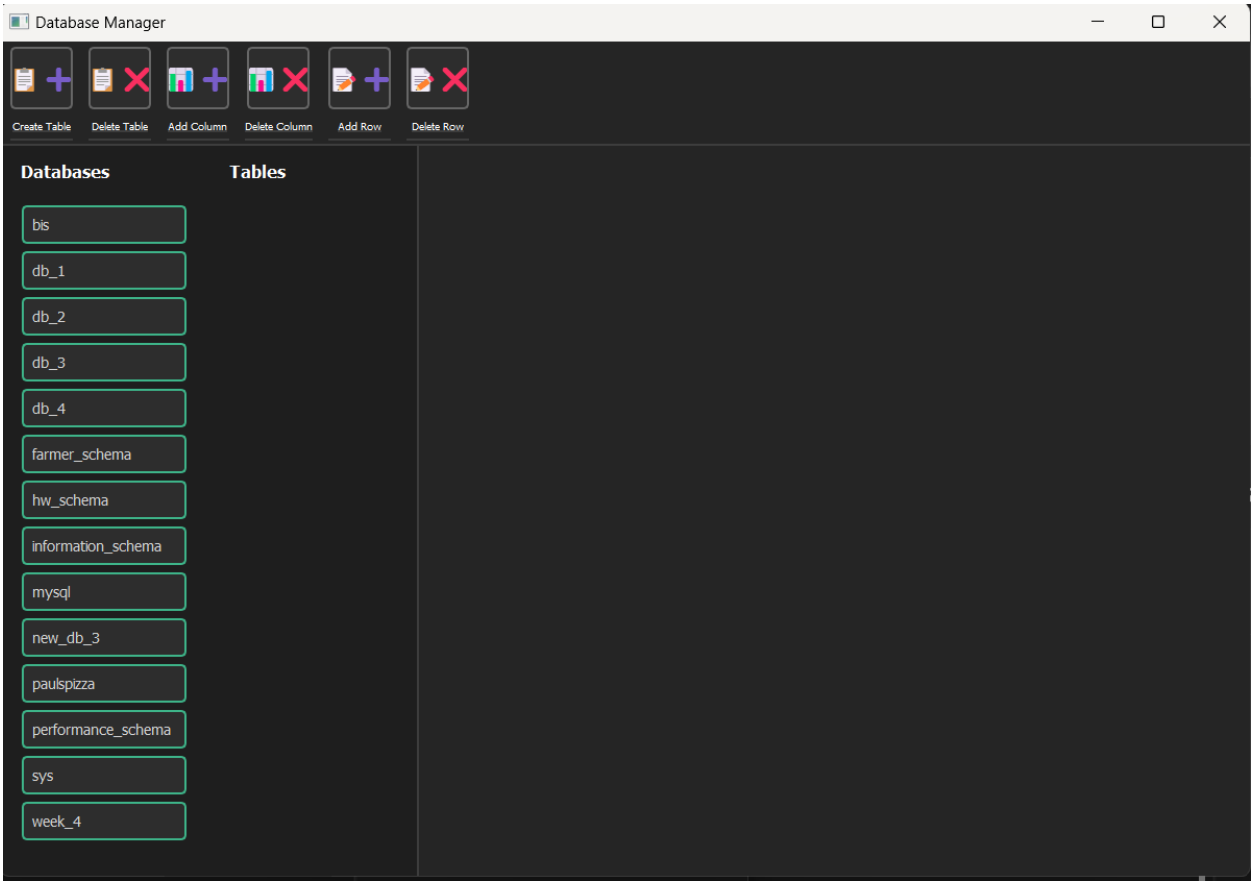
Screenshots and Evidence
Screenshots capture:

Screenshot 1: Database Manager Overview

This screenshot depicts the **Database Manager's main interface**, displaying a list of available databases on the left-hand panel. The interface includes options to create, delete, and manage tables within each database. Highlighted features:

- **Top Toolbar**: Contains options such as "Create Table," "Delete Table," "Add Column," "Delete Column," "Add Row," and "Delete Row."
-
- **Databases Panel**: Displays a list of databases, including farmer_schema, which is relevant to this project.

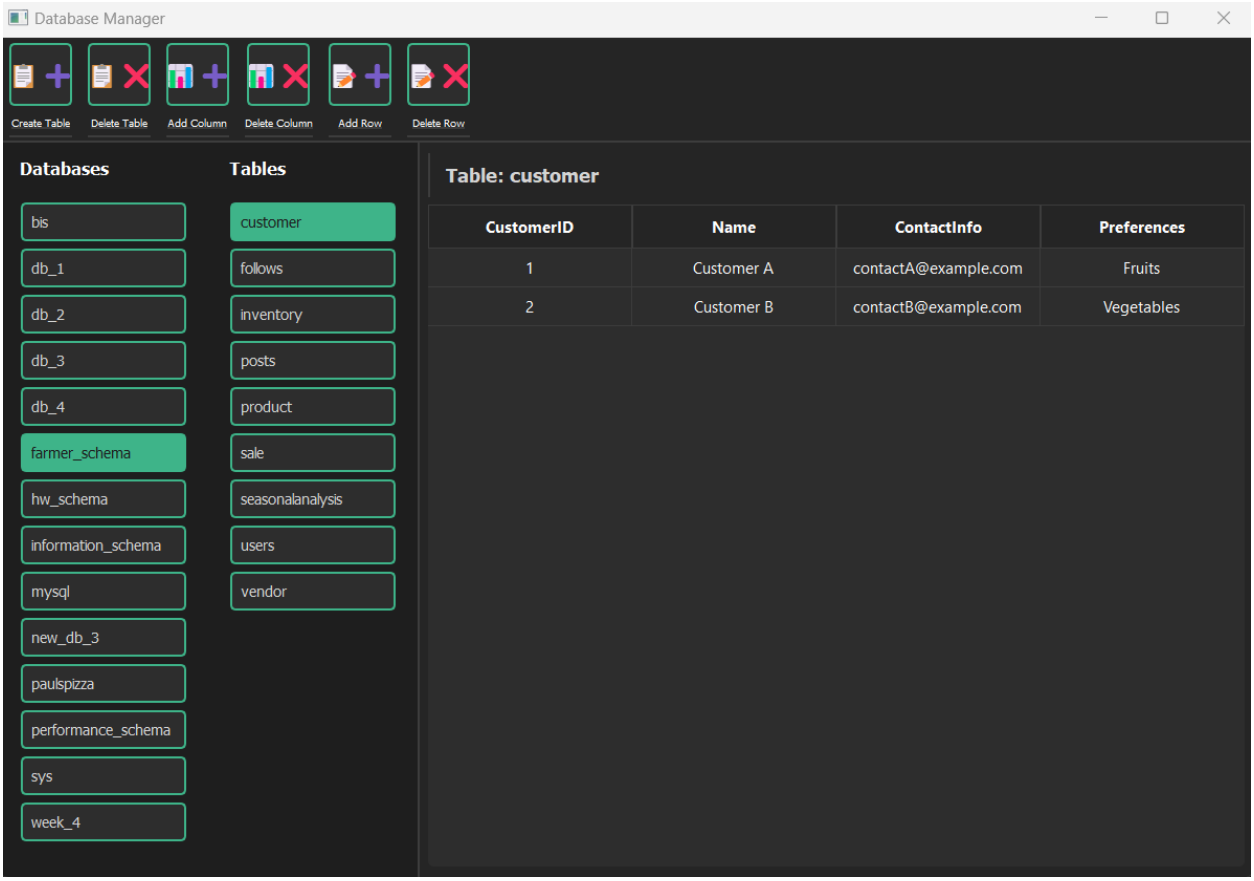
- **Tables Panel**: Lists tables within the selected database. Currently, no specific table is selected.



Screenshot 2: Customer Table Overview

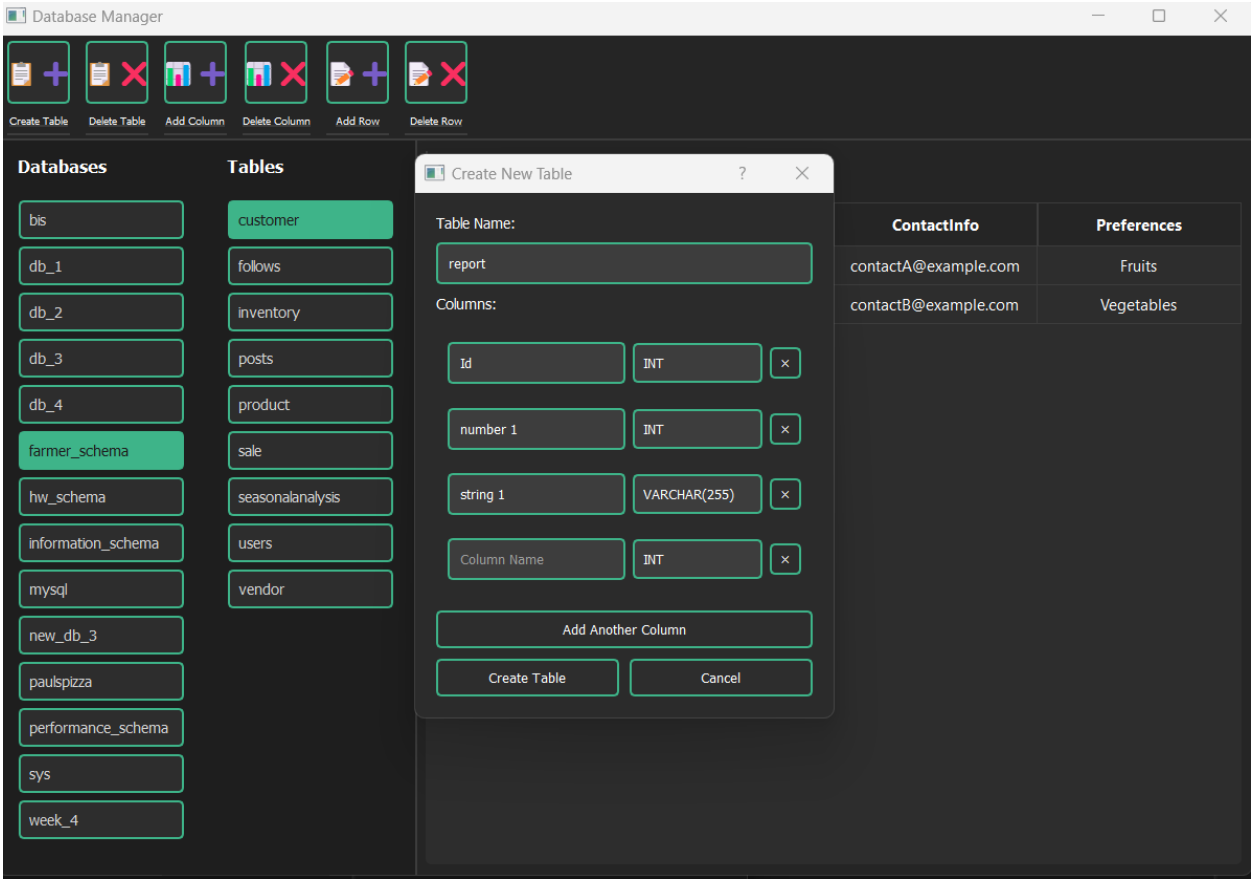
This screenshot shows the **Customer Table** under the farmer_schema database. The table stores customer-related data, including:

- **Columns**: CustomerID (primary key), Name, Contact Info, and Preferences.
- **Sample Data**: Includes two customers:
  - **Customer A** with a preference for "Fruits."
  - **Customer B** with a preference for "Vegetables." This table demonstrates the proper structure for managing customer information in the Inventory Management System.



Screenshot 3: Table Creation Dialog
This screenshot highlights the **"Create Table" dialog box**, allowing users to define a new table within the database. Key features:

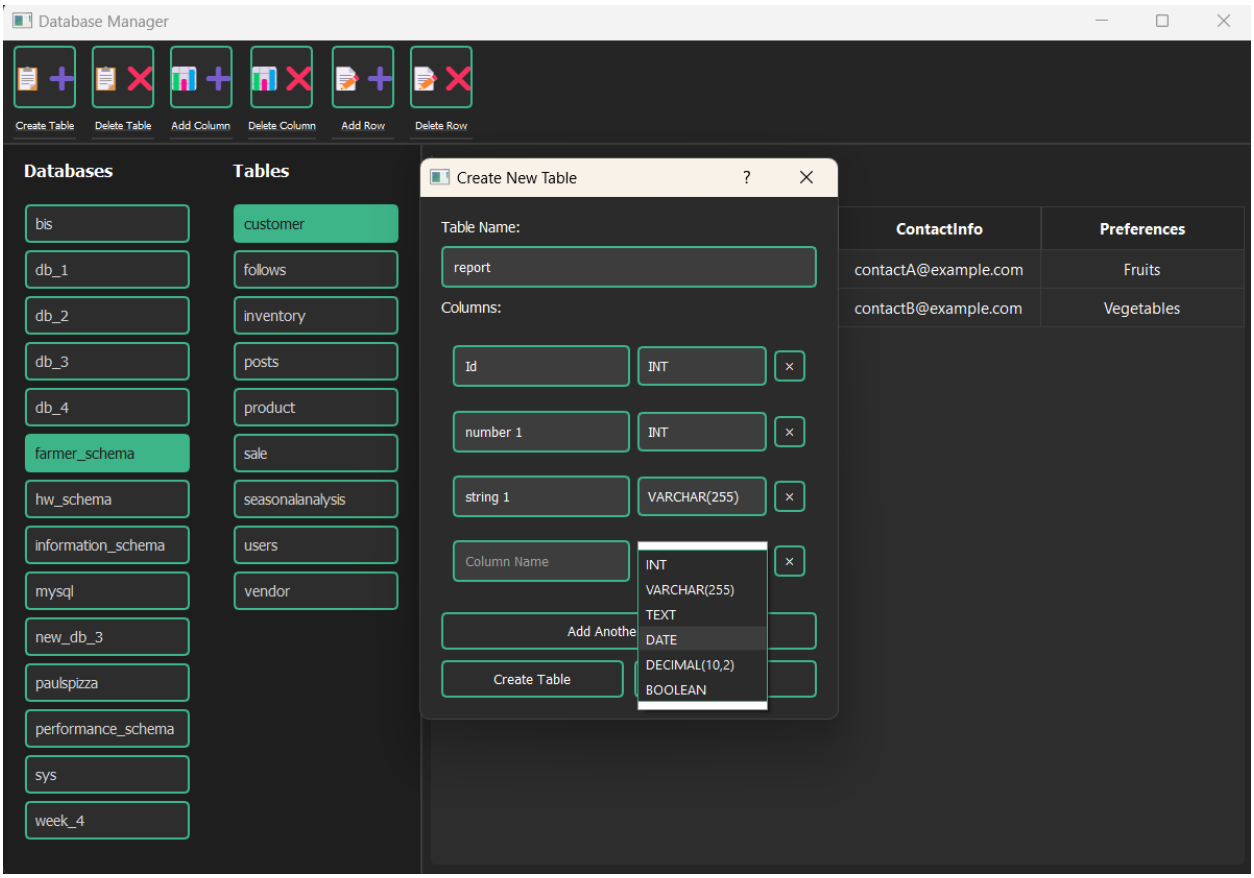
- **Table Name Field**: Input for specifying the table's name (e.g., report).
- **Columns Section**: Users can add columns with their names, data types (e.g., INT, VARCHAR), and constraints.
- **Buttons**: Options to "Add Another Column" or "Create Table." This functionality is essential for extending the database with new tables as requirements evolve.



Screenshot 4: Adding Data Types

This screenshot expands on the **Create Table dialog** by showing the dropdown menu for selecting column data types, including:

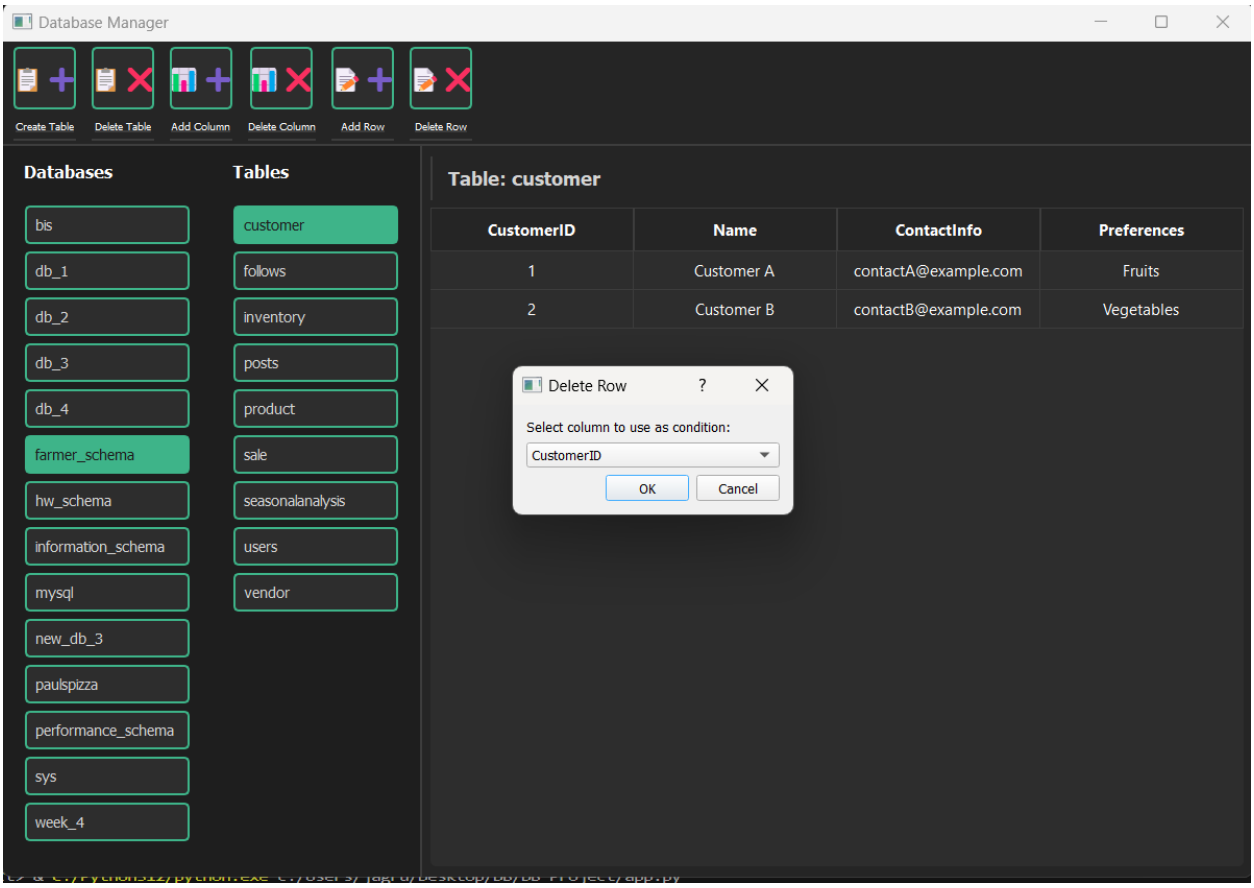
INT: For numeric values.
VARCHAR (255): For text fields.
DATE: For storing dates.
DECIMAL (10,2): For precise decimal numbers (e.g., prices).
This menu ensures proper data typing and validation while defining table columns.



Screenshot 5: Deleting a Row

This screenshot illustrates the **Delete Row functionality** for the customer table. Key highlights:

- **Selection Dropdown**: Users can select a column (e.g., CustomerID) to set the condition for deleting a row.
- **Confirmation Dialog**: Ensures that row deletion is intentional and based on the specified condition. This functionality allows administrators to maintain data accuracy by removing irrelevant or incorrect records.

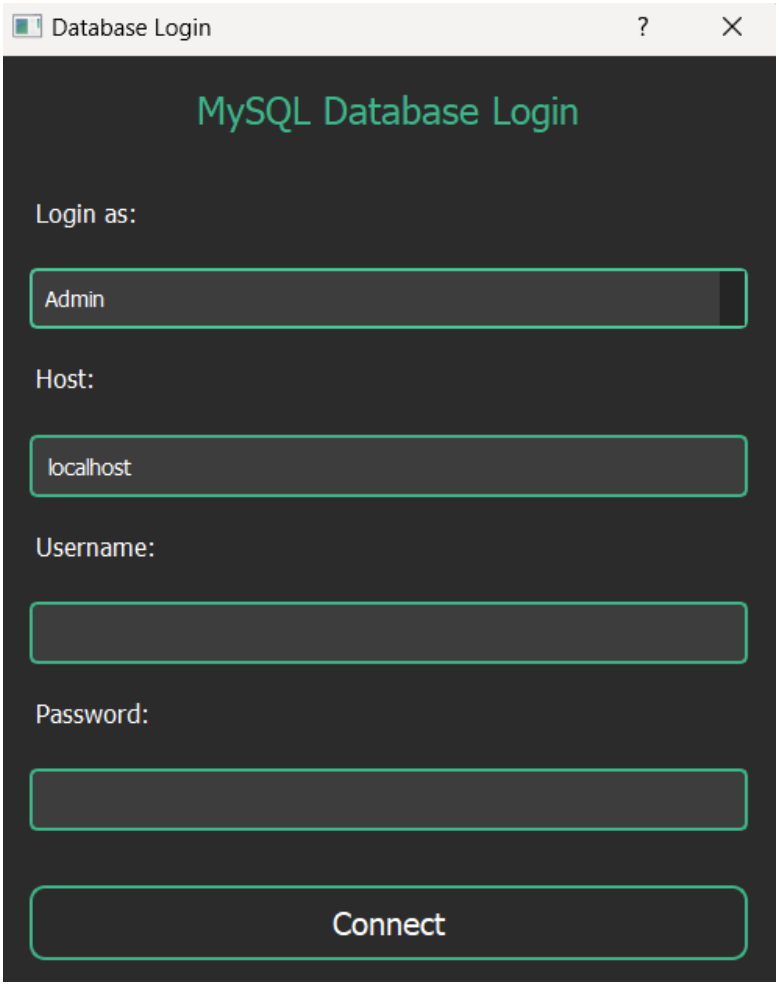


Screenshots
Include screenshots to demonstrate:
Login interface.

Screenshot 1: MySQL Database Login Interface (Admin View)

This screenshot depicts the MySQL Database Login Interface, designed to authenticate users before accessing the database. Key features include:

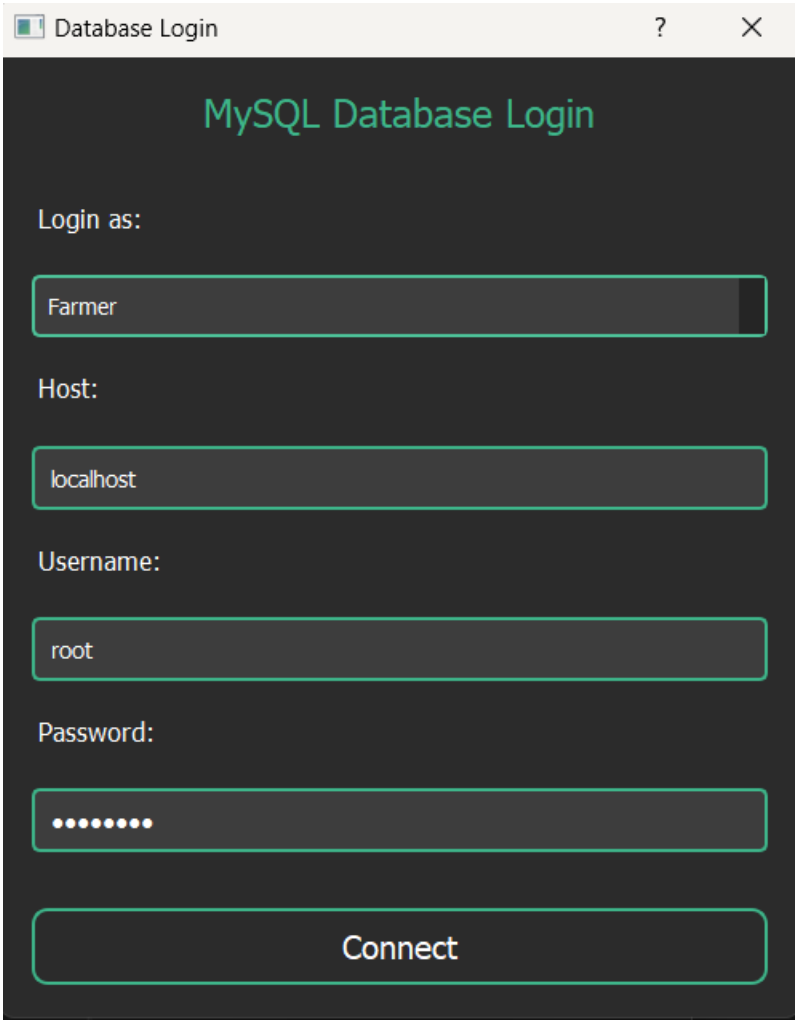
- Login as: The user role (e.g., Admin) is displayed, allowing for role-based access control.
- Host: The server hosting the database, shown here as localhost.
- Username and Password Fields: Inputs for the database administrator to securely log in.
- Connect Button: Triggers the authentication process to establish a connection with the database. This interface ensures secure access to the database, limiting unauthorized entry.



Screenshot 2: MySQL Database Login Interface (Farmer View)

This screenshot shows a similar MySQL Database Login Interface, tailored for a specific user role (Farmer). Key details:

- Login as: Displays Farmer, highlighting role-based authentication.
- Host: The database server remains localhost.
- Username and Password: Fields are pre-filled for demonstration, with the username root and a hidden password for security.
- Connect Button: Allows the user to establish a connection to the database. This view demonstrates the flexibility of the system to handle multiple roles while maintaining security and simplicity in login management.



Results:

1. **Improved Inventory Management**: Vendors could efficiently track stock levels and sales, reducing waste and improving availability.
2. **Sales Insights**: Reports provided valuable insights into top-selling items and profit margins.
3. **Seasonal Optimization**: Seasonal analysis ensured vendors were prepared for peak demand periods, minimizing stockouts.
4. **Customer Engagement**: Customer preferences helped vendors tailor their offerings.

Tools and Technologies
• SQL for database design and management.
• MySQL Workbench for schema creation and testing.

Teamwork & Collaboration (5 Points)
Team Contribution
Overview:
The development of the **Inventory Management System for a Local Farmer's Market** was successfully achieved through collaborative efforts. Each team member took on specific responsibilities to ensure the project was delivered on time and met all objectives.

Individual Contributions:

- **Shubh Patel**:
  - Designed and normalized the database schema.
  - Developed SQL queries for CRUD operations and ensured referential integrity.
  - Managed the integration of the database with the user interface.
- **JeetKumar Patel**:
  - Assisted in the creation of the ERD and schema design.
  - Implemented backend functionality to support inventory and sales tracking.
  - Tested and debugged database queries to ensure accuracy and performance.
  - Documented the development process and prepared the final project report.
- **Jagruth Reddy:**
  - Designed the user interface using HTML, CSS, and JavaScript.
  - Integrated responsive UI features and dynamic content rendering.
  - Conducted testing of UI components, ensuring real-time validation and usability.
- **Omkar Datta**:
  - Led the requirements gathering process and defined the problem statement.
  - Created mock data for testing database functionality and application workflows.

- **Manasa Tallapaka**:
  - Developed the seasonal analysis module to predict product demand using historical data.
  - Implemented advanced SQL queries for identifying seasonal trends and generating insights.
  - Conducted performance testing of seasonal analysis algorithms for accuracy and efficiency.

- **Bhanu Prakash Reddy:**
  - Designed and implemented user authentication and role-based access control for the system.
  - Built the login interface for secure access to the database.
  - Ensured encryption and security of sensitive data, including vendor and customer information.

Collaboration Process

Communication Tools:
- Weekly team meetings were conducted to discuss progress, share updates, and resolve challenges.
- Tools like Slack, Zoom, and Google Drive were used for effective communication and resource sharing.

Task Allocation:
- Tasks were divided based on individual expertise, ensuring efficient use of team resources.
- Progress tracking was maintained using a shared Google Sheet for transparency and accountability.

Peer Evaluation:

- **Feedback**: Team members provided constructive feedback during peer evaluations to recognize contributions and suggest improvements.
- **Evaluation Forms**: Each member completed a peer evaluation form, outlining their contributions and evaluating others' performance.

Challenges and Resolutions:

- **Challenge**: Synchronizing database updates with the user interface during integration.
    - **Resolution**: Conducted additional testing sessions and collaborated to debug integration issues.
- **Challenge**: Ensuring all team members were proficient in the technologies used for the project.
    - **Resolution**: Organized knowledge-sharing sessions and shared resources to bridge technical skill gaps.

Key Outcomes

- Collaborative efforts ensured the project milestones were completed on time.
- All team members made meaningful contributions, confirmed through peer evaluations.
- The project delivered a functional and scalable Inventory Management System that met the requirements and objectives.

References:

1. Database Design and Implementation Concepts (CSCI 4710 Course Material).
2. MySQL Documentation.
3. SQL Query Optimization Techniques.