Benjamin Mosley
CIDM-3350
Final Project

A) Introduction

As the importance of the grocery store has grown, the importance of responsible employees has grown alongside. One of the most important jobs for the store employees to perform is checking the “Best-By” dates. This task is performed to help ensure no items are expired or past their quality assurance date. Customers rely on the store and its employees to keep the shelves stocked with the best items, and to keep them from eating spoiled products. Not only could this affect the health of the customers, but the image of the business as a well-maintained and responsible company.

A grocery store's reputation is built on trust, and a key factor in maintaining that trust is the consistent delivery of fresh, high-quality products. Customers expect their shopping experience to be seamless, with the assurance that every item on the shelf meets strict quality standards. If this expectation is not met, not only could it lead to customer dissatisfaction, but it could also result in serious consequences such as foodborne illnesses or permanent damage to the business's reputation. Therefore, ensuring product freshness is not just a quality assurance task; it is a vital responsibility that underpins the overall success of the store.

B) Current Issues

Currently, the process of keeping track of dates is very tedious for employees to perform. As they must manually check for dates. This means physically sorting through items, remembering which items they have checked, as well as which they have not. They must perform this tedious task every week, if not every day, which can become exhausting after a while. Given the importance of this task, this is not ideal, as workers will end up missing expired items, causing the quality of items to decrease.

This manual process also opens the door to significant human error. Employees may unintentionally overlook certain items, especially in larger stores where inventory turnover is high. Additionally, the repetitive nature of the task leads to employee fatigue and a decrease in efficiency over time. The lack of a centralized system means that communication between employees is limited, and there is no reliable way to track which items have already been checked. This not only wastes time but also increases the likelihood of expired items being left on the shelves, negatively impacting both customer trust and the store's bottom line.

C) Solution

This is why I believe a centralized database will greatly enhance the performance of employees in managing inventory and maintaining product quality. By providing employees with an organized and centralized location to check which items must be removed and which ones are still safe for sale, this system will not only improve operational efficiency but also ensure higher customer satisfaction. Customers will feel more confident purchasing items from the store, knowing that expired or poor-quality products are no longer an issue.

The database will also assist the company by reducing data redundancy. Currently, employees rely on memory or inconsistent manual processes to track which items they have checked and which they haven't. This leads to inefficiencies such as rechecking already verified items or, worse, skipping items that need urgent attention. Implementing a database system that uniquely identifies each batch of items, their department, and their expiration date will eliminate this redundancy and improve accuracy.

Additional Benefits:

- *Reduced Data Redundancy*
    - By uniquely identifying each inventory batch with details such as the department, supplier, and expiration date, the database eliminates repetitive tasks and ensures no expired product is overlooked. Employees can focus on what needs immediate attention without guesswork or duplication of efforts.

- *Improved Efficiency and Time Management*
    - The system will allow employees to prioritize tasks effectively by providing a clear list of batches that are nearing expiration. This saves valuable time that would otherwise be spent physically sorting and manually checking items, allowing employees to focus on other critical tasks.

- *Better Product Quality Assurance*
    - A centralized database ensures that expired items are flagged and removed in a timely manner, keeping shelves stocked only with fresh and high-quality products. This will significantly improve the store's reputation as a responsible and customer-focused business.

- *Real-Time Inventory Tracking*
    - Employees and managers can access up-to-date information on inventory batches, including quantities and expiration dates. This enables better planning for restocking and prevents over-ordering or understocking of products.

- *Improved Employee Morale*
    - Automating tedious tasks like manual date checking reduces employee frustration and fatigue. A user-friendly system streamlines their workflow, allowing them to perform their jobs more effectively and with greater satisfaction.

- *Enhanced Customer Trust*
    - Customers rely on the store to provide safe, high-quality products. By consistently removing expired items and keeping shelves stocked with fresh inventory, the database ensures a safer shopping experience, fostering customer

loyalty and trust.

- *Scalability for Business Growth*
    - As the store grows, managing inventory manually will become increasingly challenging. A centralized database is scalable and can handle larger inventories, multiple departments, and more employees without compromising accuracy or efficiency.

How It Works:

1. Batch Identification: Each batch of inventory is assigned a unique identifier, linked to details like item type, department, supplier, and expiration date.
2. Automated Alerts: The system can generate automated alerts or reports for batches nearing their expiration, ensuring timely removal.
3. Accessible Interface: Employees can log in to the database and view an organized list of items requiring attention, reducing the need for physical checks.
4. Integration with Departments: Departments can access the database to view the status of their inventory, aiding in better coordination between employees and management.

D) Business Impact

This system will enhance your ability to maintain **high-quality inventory** with detailed batch and expiration tracking.Improve **accountability** through clear relationships between employees, departments, and managers.Ensure **accurate stock levels** and reduce losses due to spoilage or errors.Build stronger **supplier relationships** by keeping all communication and order details organized. Create an efficient **warehouse system**, reducing retrieval time and enhancing operational efficiency.

This system is designed to scale with your business, supporting growth while keeping operations streamlined and efficient.

Business Rules for Inventory Management System

The **Inventory Management System** is designed to help your business efficiently track employees, departments, inventory, suppliers, and warehouses while ensuring product quality through detailed batch tracking. Below, I'll explain the key business rules implemented in the system and how they align with your operations:

1. Employee Management and Hierarchies

- **Rule**: Each employee works in a department and may have a manager overseeing their tasks.

- **Why It Matters**: This structure helps maintain accountability and clear reporting lines. For example, managers can track employee performance and ensure tasks are completed on time.
- **Example in the System**: Employee "Bob Smith" works in the "Inventory Management" department and reports to "Alice Johnson," who is the department manager.

2. Department Specialization

- **Rule**: Departments are responsible for specific tasks, such as inventory management, safety training, or customer support.
- **Why It Matters**: Clearly defining departmental responsibilities ensures employees receive the proper training, fair wages, and assigned tasks that match their skill set.
- **Example in the System**: The "Safety Training" department oversees employee safety protocols, while "Inventory Management" focuses on managing stock levels.

3. Inventory Tracking by Department

- **Rule**: Each department manages its specific inventory, ensuring accountability for stock levels and the quality of products.
- **Why It Matters**: This system reduces errors and ensures departments can focus on the inventory that directly impacts their operations.
- **Example in the System**: The "Inventory Management" department oversees the various departments and their respective items. Such as milk in the dairy department, or carrots in the produce department.

4. Batch Tracking for Quality Control

- **Rule**: Inventory is tracked in batches, with each batch assigned an expiration date and quantity.
- **Why It Matters**: Tracking batches ensures that perishable or time-sensitive items are used before their expiration dates. This avoids losses due to spoilage or outdated products.
- **Example in the System**: A batch of groceries from "Barilla" has an expiration date of November 15, 2024, ensuring employees can prioritize its sale or use.

5. Supplier Relationships

- **Rule**: Each supplier provides inventory to the warehouses, and their details (e.g., name, location, and contact) are recorded.
- **Why It Matters**: Maintaining strong relationships with suppliers ensures a steady flow of inventory and improves communication for reorders or addressing issues.
- **Example in the System**: "Dave’s Family Farms" supplies produce items to the "Produce Inventory" warehouse in Kentucky.

6. Warehouse and Inventory Storage

- **Rule**: Warehouses store specific items and maintain clear relationships with suppliers and inventory.
- **Why It Matters**: Efficiently organizing inventory in warehouses ensures faster retrieval and better stock management, improving overall operational efficiency.
- **Example in the System**: The "Dry Goods Inventory" in Colorado stores dry food items like pasta from Barilla.

7. Employee-Department Assignments

- **Rule**: Employees are assigned to departments through a relationship table to ensure flexibility in assignments and updates.
- **Why It Matters**: This allows managers to move employees between departments as needed, ensuring adaptability to changing business needs.
- **Example in the System**: "Diana Green" works in "Customer Support," providing assistance to customers and resolving complaints.

8. Inventory Relationships with Suppliers and Warehouses

- **Rule**: Each item in the inventory is tied to a supplier, department, and warehouse for clear tracking.
- **Why It Matters**: This ensures that every item is accounted for and easily traced to its origin, reducing the risk of lost or misplaced inventory.
- **Example in the System**: A t-bone steak is linked to the "Signature Farms" supplier and stored in the "Meat Market Inventory."

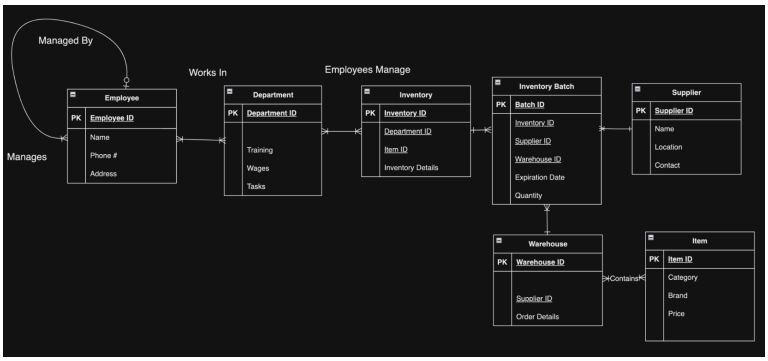
9. Warehouse-Item Relationship

- **Rule**: Warehouses store specific items, and the relationships are tracked in a "Contains" table.
- **Why It Matters**: This organization ensures warehouses can provide detailed reports about the items they store, streamlining inventory checks.
- **Example in the System**: The "Dairy Inventory" contains dairy products such as Gandy's milk.

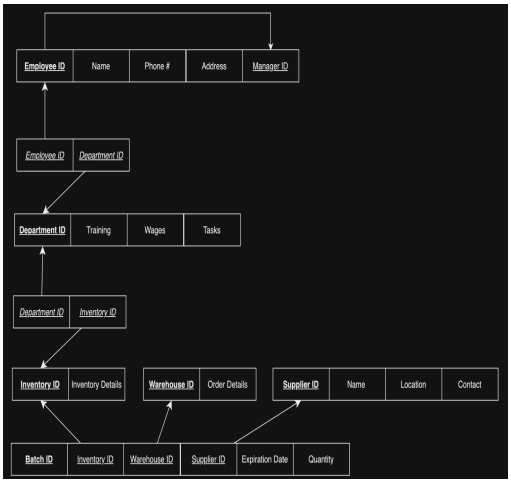
10. Comprehensive Reporting and Analysis

- **Rule**: The system allows for detailed reports, such as total inventory quantities, batch details, and department performance.
- **Why It Matters**: This enables business owners to make data-driven decisions about reordering inventory, optimizing employee performance, or negotiating supplier contracts.
- **Example in the System**: A report shows that Gandy's inventory needs replenishment as only fifty items remain in stock. Compared to the other items, which have stock-piles in the hundreds.

Entity Relationship Diagram



Entity Relationship Diagram in Third Normal Form (3NF)



SQL (Structured Query Language) Demonstration and Examples

1. **Table Creation**

```sql
CREATE TABLE IF NOT EXISTS Employee (
    EmployeeID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255) NOT NULL,
    PhoneNumber VARCHAR(15),
    Address VARCHAR(255),
    ManagerID INT,
    FOREIGN KEY (ManagerID) REFERENCES Employee(EmployeeID)
);
```

```sql
CREATE TABLE IF NOT EXISTS Department (
    DepartmentID INT PRIMARY KEY AUTO_INCREMENT,
    Training VARCHAR(255),
    Wages DECIMAL(10, 2),
    Tasks VARCHAR(255)
);
```

```sql
CREATE TABLE IF NOT EXISTS WorksIn (
    EmployeeID INT,
    DepartmentID INT,
    PRIMARY KEY (EmployeeID, DepartmentID),
    FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID),
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);
```

```sql
CREATE TABLE IF NOT EXISTS Inventory (
    InventoryID INT PRIMARY KEY AUTO_INCREMENT,
    DepartmentID INT,
    ItemID INT,
    InventoryDetails VARCHAR(255),
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID),
    FOREIGN KEY (ItemID) REFERENCES Item(ItemID)
);
```

```sql
CREATE TABLE IF NOT EXISTS Item (
    ItemID INT PRIMARY KEY AUTO_INCREMENT,
    Category VARCHAR(255),
    Brand VARCHAR(255),
    Price DECIMAL(10, 2)
);
```

```sql
CREATE TABLE IF NOT EXISTS InventoryBatch (
    BatchID INT PRIMARY KEY AUTO_INCREMENT,
    InventoryID INT,
    SupplierID INT,
    WarehouseID INT,
    ExpirationDate DATE,
    Quantity INT,
    FOREIGN KEY (InventoryID) REFERENCES Inventory(InventoryID),
    FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID),
    FOREIGN KEY (WarehouseID) REFERENCES Warehouse(WarehouseID)
);
```

```sql
CREATE TABLE IF NOT EXISTS Supplier (
    SupplierID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255),
    Location VARCHAR(255),
    Contact VARCHAR(255)
);
```

```sql
CREATE TABLE IF NOT EXISTS Warehouse (
    WarehouseID INT PRIMARY KEY AUTO_INCREMENT,
    SupplierID INT,
    OrderDetails VARCHAR(255),
    FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID)
);
```

```sql
CREATE TABLE IF NOT EXISTS Contains (
    WarehouseID INT,
    ItemID INT,
    PRIMARY KEY (WarehouseID, ItemID),
    FOREIGN KEY (WarehouseID) REFERENCES Warehouse(WarehouseID),
    FOREIGN KEY (ItemID) REFERENCES Item(ItemID)
);
```

2. Data Insertion

```sql
INSERT INTO Employee (Name, PhoneNumber, Address, ManagerID)
VALUES
    ('Alice Johnson', '555-1234', '123 Main St', NULL),
    ('Bob Smith', '555-5678', '456 Elm St', 1),
    ('Charlie Brown', '555-8765', '789 Oak St', 1),
    ('Diana Green', '555-4321', '321 Pine St', 2);
```

```sql
INSERT INTO WorksIn (EmployeeID, DepartmentID)
VALUES
    (1, 1),
    (2, 2),
    (3, 2),
    (4, 3);
```

```sql
INSERT INTO Department (Training, Wages, Tasks)
VALUES
    ('Safety Training', 20.50, 'Oversee safety'),
    ('Inventory Management', 18.75, 'Manage stock levels'),
    ('Customer Support', 17.00, 'Assist customers');
```

```sql
INSERT INTO Item (Category, Brand, Price)
VALUES
    ('Dairy', 'Milk', 299.99),
    ('Dry Goods', 'Pasta', 5.99),
    ('Produce', 'Carrots', 49.99),
    ('Meat', 'T-Bone Steak', 199.99);
```

```sql
INSERT INTO Inventory (DepartmentID, ItemID, InventoryDetails)
VALUES
    (2, 1, 'Dairy Inventory'),
    (2, 2, 'Dry Goods Inventory'),
    (2, 3, 'Produce Inventory'),
    (2, 4, 'Meat Market Inventory');
```

```sql
INSERT INTO Supplier (Name, Location, Contact)
VALUES
    ('Gandy', 'Texas', 'Gandy@gandysdairy.com'),
    ('Barilla', 'Colorado', 'Barilla@barilla.com'),
    ('Daves Family Farms', 'Kentucky', 'Dave@dff.com'),
    ('Signature Farms', 'California', 'Signature@signatureselect.com');
```

```sql
INSERT INTO Warehouse (SupplierID, OrderDetails)
VALUES
    (1, 'Dairy storage'),
    (2, 'Dry Goods storage'),
    (3, 'Produce storage facility'),
    (4, 'Meat storage facility');
```

```sql
INSERT INTO InventoryBatch (InventoryID, SupplierID, WarehouseID, ExpirationDate, Quantity)
VALUES
    (1, 1, 1, '2024-12-31', 50),
    (2, 2, 2, '2024-11-15', 200),
    (3, 3, 3, '2025-01-20', 100),
    (4, 4, 4, '2025-06-30', 30);
```

```sql
INSERT INTO Contains (WarehouseID, ItemID)
VALUES
    (1, 1),
    (2, 2),
    (3, 3),
    (4, 4);
```

SQL Queries and Outputs

1. **Retrieve all employees and their managers**

```sql
-- 1. Retrieve all employees and their managers
SELECT e1.Name AS Employee, e2.Name AS Manager
FROM Employee e1
LEFT JOIN Employee e2 ON e1.ManagerID = e2.EmployeeID;
```

Output

	Employee	Manager
	Alice Johnson	NULL
	Bob Smith	Alice Johnson
	Charlie Brown	Alice Johnson
	Diana Green	Bob Smith

2. **Retrieve all items with their prices**

```sql
-- 2. Retrieve all items with their prices
SELECT Category, Brand, Price FROM Item;
```

Output

	Category	Brand	Price
	Dairy	Milk	299.99
	Dry Goods	Pasta	5.99
	Produce	Carrots	49.99
	Meat	T-Bone Steak	199.99

3. **Retrieve inventory details**

```sql
-- 3. Retrieve inventory details
SELECT InventoryDetails, DepartmentID FROM Inventory;
```

Output

	InventoryDetails	DepartmentID
	Dairy Inventory	2
	Dry Goods Inventory	2
	Produce Inventory	2
	Meat Market Inventory	2

4. **Retrieve items stored in warehouses**

```sql
-- 4. Retrieve items stored in warehouses
SELECT w.OrderDetails, i.Category, i.Brand
FROM Warehouse w
JOIN Contains c ON w.WarehouseID = c.WarehouseID
JOIN Item i ON c.ItemID = i.ItemID;
```

Output

	OrderDetails	Category	Brand
	Dairy storage	Dairy	Milk
	Dry Goods storage	Dry Goods	Pasta
	Produce storage facility	Produce	Carrots
	Meat storage facility	Meat	T-Bone Steak

5. **Retrieve inventory batches and their expiration dates**

```sql
-- 5. Retrieve inventory batches with expiration dates
SELECT ib.BatchID, i.InventoryDetails, ib.ExpirationDate, ib.Quantity
FROM InventoryBatch ib
JOIN Inventory i ON ib.InventoryID = i.InventoryID;
```

Output

	BatchID	InventoryDetails	ExpirationDate	Quantity
	1	Dairy Inventory	2024-12-31	50
	2	Dry Goods Inventory	2024-11-15	200
	3	Produce Inventory	2025-01-20	100
	4	Meat Market Inventory	2025-06-30	30

6. **Retrieve all suppliers and their associated warehouses**

```sql
-- 6. Retrieve all suppliers and their associated warehouses
SELECT s.Name AS Supplier, w.OrderDetails AS Warehouse
FROM Supplier s
JOIN Warehouse w ON s.SupplierID = w.SupplierID;
```

Output

Supplier	Warehouse
Gandy	Dairy storage
Barilla	Dry Goods storage
Daves Family Farms	Produce storage facility
Signature Farms	Meat storage facility

7. **Count the total number of employees**

```
-- 7. Count the total number of employees
SELECT COUNT(*) AS TotalEmployees FROM Employee;
```

Output

TotalEmployees
4

8. **Create a sum of all inventory quantities**

```
-- 8. Sum of all inventory quantities
SELECT SUM(ib.Quantity) AS TotalQuantity FROM InventoryBatch ib;
```

Output

TotalQuantity
380

9. **Calculate average wages in each department**

```
-- 9. Average wages in each department
SELECT d.Training, AVG(d.Wages) AS AvgWages
FROM Department d
GROUP BY d.Training;
```

Output

	Training	AvgWages
	Safety Training	20.500000
	Inventory Management	18.750000
	Customer Support	17.000000

10. **Sum of the total value of each item in the inventory**

```sql
-- 10. Total value of items in each category
SELECT i.Category, SUM(i.Price * ib.Quantity) AS TotalValue
FROM Item i
JOIN InventoryBatch ib ON i.ItemID = ib.InventoryID
GROUP BY i.Category;
```

Output

	Category	TotalValue
	Dairy	14999.50
	Dry Goods	1198.00
	Produce	4999.00
	Meat	5999.70