ADDIS ABABA UNIVERSITY (AAU)

COLLEGE OF NATURAL AND COMPUTATIONAL SCIENCE
SCHOOL OF INFORMATION SCIENCE
FINAL PROJECT DOCUMENTATION

Inventory Management System For Arada Mart Supermarket (4 Kilo)

***Academic Year**: 2024 G.C.*

Course Title: *FUNDAMENTALS OF DATABASE SYSTEMS*
***Course Code:** INSY 2031*

SECTION 3
Done by:
MELORI ALEMU (UGR/4962/15)
MICHAEL BEHAILU (UGR/4067/15)
REBECCA SINTAYEHU (UGR/6532/15)
YONATAN GENENE (UGR/2866/15)
YORDANOS TAMRAT (UGR/6532/15)

SUBMITTED TO :
Instructor Adey E.

Table of content

1. Introduction

1.2. Background

Supermarkets are facing more and more difficulties in successfully and efficiently managing their inventories in the fast-paced retail conditions of today. Supermarkets need a strong inventory management system to remain profitable and competitive due to their large product selection, shifting demand patterns, and requirement to balance stock levels to suit consumer needs while minimizing costs as much possible. This supermarket inventory management system we made for Arada mart supermarket was developed in response to these challenges, aiming to streamline inventory processes, optimize stock levels, and enhance overall operational efficiency. Our project aimed to create a centralized database system that would store and manage critical information related to the supermarket's inventory, suppliers, sales data, and customer information levitating the experiences of all the parties involving.

1.3. Purpose of the system

The purpose of the database system designed for the supermarket is to effectively handle and arrange vital information about inventory, vendors, sales, clients, and other operational aspects of the supermarket. the system provides us with basic information maintenance function so that managers may go through the function to add, remove, and amend the basic information of membership and goods. The supermarket management system makes it very easy to handle data input, output, and retrieval, allowing for the specific, visual, and rationalization of disorganized supermarket data.
All things considered, the main goals of the supermarket database system are to effectively consolidate and manage vital data, boost customer satisfaction, streamline operations, maximize inventory control, and increase corporate profitability.

1.4. Statements of the problem

Even though the supermarket currently has a database system some pieces of information (for instance products expire date) was missing here and there. With our new version of database it was tried to cover these information for a better experience.

1.5. The scope of the project

The scope of this project considers the primary and most essential parts of the inventory management system.

- **Inventory items or Products:** The database would contain details about each item in the stock including its name, brand, category, unit price, Quantity and Expire date.
- **Purchase orders:** the purchase order aspects of the data base will include order id, order date, ordered quantity, unit-price and total amount.
- **Employee:** our database will efficiently organize human resources and maintain track of each person throughout their involvement with the system.
- **Supplier:** the supplier's name, contact person, phone number, email are the main things to be considered in the suppliers data aspects.
- **Customers:** the supermarket's customers name, contact and address will be included here.
- **Invoice/sales**: The database would include information about sales of inventory items, including the date of the sale's id, the sales description, the quantity sold and if there is any discount.
- **Payment detail:** the payment method and Id, amount paid and payment date are included.
- **Loyalty**: if there are points earned and membership status are considered here.

1.6. Objective of this project

1.6.2 General objective

The main goal of a supermarket database system is to offer a consolidated, well-organized platform for handling, storing, and retrieving all pertinent information about the store's activities.
With the help of the inventory management system it is possible organize internal activities, control the flow of supplies, make efficient use of personnel and equipment, and communicate with customers

1.6.3 Specific goals

Efficient data Management: The primary objective of a database system is to efficiently store and manage vast amounts of data related to inventory, sales, customers, suppliers, employees, and transactions in a structured and organized manner.
Better Inventory control: To facilitate efficient inventory management, the database system track product movements, determine reorder points, and optimize stock replenishment procedures to avert stock outs and overstock scenarios. It also provide insight into stock levels.

Data Integrity and accuracy: The database system ensures data integrity by maintaining accurate and consistent information across all records, preventing duplication, errors, and inconsistencies in data entry and retrieval.
Enhanced consumer Service: The database system should assist personalize marketing activities, track consumer preferences, and improve customer service by providing customized promotions, loyalty programs, and personalized suggestions by collecting and analyzing customer data.
Scalability and Flexibility: The database system will be scalable to accommodate the supermarket's growing data needs and flexible enough to adapt to changing business requirements, technological advancements, and industry trends.
Streamlining operations: the inventory management system can help streamline operations by automating inventory tracking and ordering, reducing manual labor and errors.

1.7. Database development methodology

1.7.2 Data sources and collection methods

In the process of collection of data for the supermarket's database system historical data from previous systems or records was used to populate the database. This data includes sales records, customer information, inventory data, and financial records and the Employee Records like employee data, including names, contact information, roles, and schedules was obtained from HR of the supermarket.

1.7.3 Database analysis and design Methods

The data analysis and design portion of the database will involve the following steps:

- **Entity-relationship(E-R) modeling:** Identifying the entities, attributes, and relationships between them.
- **Normalization**: this deals with ensuring that the database is in a normalized form to minimize redundancy and improve data integrity.
- **Shema designing** : designing the database schema based on the ER model, including tables, relationships, and constraints.

1.8. Deliverables of the project

A. **Database design:** a carefully thought-out and recorded database schema that describes the characteristics, connections, and organization of the data elements in the system.
B. **Friendly Interfaces for data Entry:** Interfaces or forms that are easy to use for entering and maintaining data in a database system.
C. **Data Validation and Quality control:** by implementing data validation rules and mechanisms to ensure the accuracy, consistency, and integrity of data entered into the database including checks for data types, constraints, and business rules.
D. **Reporting and analytics:** this involves creating and producing reports, dashboards, and data visualizations that assists in decision-making and performance analysis.
E. **User documentation:** a documentation that provides guidance on how to use the database system, including instructions for data entry, report generation, and system administration tasks. That helps supermarket staff understand and effectively utilize the system.
F. **Project documentation**: information that includes the specifications for the project, design choices, implementation specifics, and any database system setups or customizations. This documentation can be referred to for maintenance, updates, and additions.
G. **System Maintenance and Support:** Establishing processes and support mechanisms for ongoing system maintenance, including database backups, system upgrades, and troubleshooting. This ensures the long-term stability and reliability of the database system.

1.9. Development tools, platforms, and technologies

Relational database management system specifically Microsoft SQL server for database storage and management was used.

1.10. Project time plan

The project time plan prepared using gantt chart

	Week1	Week2	Week3	Week4	Week5	Week6
Task 1	Planning phase					
Task 2		Analysis phase				
Task 3			Design phase			
Task 4				implementation		
Task 5					Testing phase	

2. Requirement specification

2.2. Data requirements

- **Product data:** detailed and up to date information about the supermarket's items included in the database. This comprises data such as product names, descriptions, categories, pricing, barcodes, suppliers, and stock levels.
- **Inventory data**: accurate and current information regarding the supermarket's inventory must be maintained in the database. This entails keeping track of the number of each product in stock, the locations of the stocks, the expiration dates of perishables, and any movements or modifications to the stock.
- **Sales and Transaction data:** Information on items purchased, their quantities, pricing, discounts, payment methods, and client details should all be tracked and recorded in the database.
- **customer Information:** contact information, past purchases, involvement in loyalty programs, and other behavioral or demographic data should all be kept in the database. This makes it possible to segment customers, target promotions, and use tailored marketing.
- **Supplier data:** The database should maintain details about the supermarket's suppliers, including contact information, pricing agreements, delivery schedules,

and product sources. This data facilitates effective supplier management, order tracking, and maintaining strong supplier relationships.

- **Employee data:** Information on employees, including as schedules, personal information, work roles, and performance reviews, must be kept in the database. Performance review, payroll processing, and labor management are all aided by this data.

2.3 Transaction requirements

2.3.1 Data entry requirements

Data entry requirements for the supermarket database system includes the following:

- If the supermarket maintains a customer database, the system should provide fields for entering customer information, such as names, contact details, loyalty program identifiers, and purchase history.
- The system should have features to manage user access and security, including assigning user roles and permissions to control data entry and modification privileges.
- Specify appropriate data types and constraints for each database field to ensure accurate storage of data and enforce validation rules.
- Implementing indexing on key fields to optimize data retrieval performance, especially for frequently accessed data.
- Establish backup and recovery procedures to safeguard against data loss and ensure the availability.
- Entering information about the goods that are sold in the store, including as names, prices, categories, suppliers, barcodes or id numbers, and stock levels, is part of this process.
- Data entry for inventory management should be possible through the system. This includes tracking expiry dates for perishable commodities, logging stock movements (including receiving new shipments or moving things between locations), and updating stock levels.
- The system need to facilitate data input for sales transactions, encompassing the recording of particulars such the date, time, items acquired, their quantities, costs, rebates, modes of payment, and, if relevant, customer information.
- The system may include fields for entering supplier details, such as supplier names, contact information, payment terms, and any contractual agreements or special terms.

- The system should allow for data entry related to promotional campaigns, discounts, and special offers. This may include entering details about the promotion, applicable products, discount percentages or amounts, and validity dates.

2.3.2 Data retrieval requirements

Supermarket database system's data retrieval requirements cover the procedures and factors to be taken into account when properly and effectively extracting certain data from the database. Here are a few points to remember:

- Data retrieval processes should be scalable and flexible, accommodating the increasing volume of data as the supermarket grows.
- Optimize database queries to ensure fast retrieval of product information, employee's information, transaction history, customer data, supplier's data and financial details, especially for large datasets.
- The database should store and allow retrieval of transaction records, including details such as date, time, purchased products, quantities, prices, and payment information.
- The database should support data retrieval for generating reports and performing analytics.

2.3.3 Data updating requirements

The procedures and factors involved in updating and changing correct and current data inside a supermarket database system are known as data update requirements. Some of them are listed below

- Manage frequent product additions, changes, and discontinuations to provide accurate and up-to-date data for inventory control and customer satisfaction.
- Define and enforce appropriate update permissions to modify inventory details, sales, employee's data, and financial information.
- Adherence to relevant data protection regulations, privacy laws, and industry-specific compliance requirements during data updates.
- Implement data validation checks during the update process to ensure that only accurate and valid information is entered into the database.
- Proper handling of errors and exceptions during data updates, including informative error messages and appropriate error recovery mechanisms.
- Implementation of appropriate access controls to ensure that only authorized personnel can perform data updates.

2.3.4. Data removal requirements

Data removal requirements in the context of a supermarket's database system would include the procedures and factors unique to maintaining and erasing data inside that system. Here are some things to think about:

- Privacy concerns regarding customer data: it includes complying with applicable data protection rules and regulations, such as getting consent for data collecting and providing channels for consumers to request the deletion of their data.
- Transactional data: anonymizing or erasing transactional data after a certain amount of time, taking into account things like business demands for analytical or auditing reasons and regulatory obligations.
- Loyalty Programs: This entails giving clients the option to withdraw from the program and having their data deleted upon request.
- Employee Information: Managing personnel records, abiding with privacy laws, and safely erasing employee data upon termination or as mandated by data retention policy.
- Backup and archiving: taking legal obligations and retention rules into account when deleting or removing data from backups and archives.
- Storage cost Optimization: Removing outdated or redundant data can help optimize storage costs and streamline database operations.

In general, it is crucial to adhere to best practices for data deletion when implementing data removal requirements in supermarket database systems. These best practices include making sure data is safely removed from the database, keeping track of data removal activities through audit trails, and recording the justifications for data removal in order to prove compliance with internal and regulatory policies.

3. Database Design of the Inventory Management System

3.2. Conceptual Database Design

3.2.1 Entities Identification and Description

1. Supplier: Represents the entities that supply products to the supermarket.

2. Employee: Represents individuals employed by the supermarket, including their roles and responsibilities.

3. Customer: Represents individuals or entities that purchase products from the supermarket.

4. Sale: Represents transactions involving the sale of products to customers.

5. Product: Represents the various products available for sale in the supermarket.

6. Loyalty: Represents loyalty programs that may be associated with customers.

7. PurchaseOrder: Represents orders placed by the supermarket to suppliers for the purchase of products.

8. Store: Represents physical store locations operated by the supermarket.

3.2.2. Relationships of Entities

1. Supplier Entity:

- One-to-Many with PurchaseOrder: One supplier can supply multiple purchase orders.
- Many-to-Many with Product: A supplier supplies multiple products, and a product can be supplied by many suppliers.

2. Employee Entity:

- One-to -many with Purchase Order: One Employee can make multiple purchase Orders
- One-to-Many with Sale: One Employee can make multiple sale

- Many-to-One with store: Many Employee can work for one store

3. Customer Entity:

- One-to-Many with Sale: One customer can make multiple purchases.

4. Sale Entity:

- Many-to-One with Customer: Many sales can be associated with one customer.
- Many-to-One with Employee: Many sales can be processed by one employee.
- Many-to-Many with Product: A sale involves multiple products, and a product can be part of multiple sales.
- Many-to-One with store: Many sales are performed in a store

5. Product Entity:

- Many-to-Many with Sale: A product can be sold in multiple sales, and a sale can include multiple products.
- Many-to-Many with Store: A product can be sold in multiple stores, and a store can include multiple products
- One-to-Many with PurchaseOrder: One product can be part of multiple purchase orders.
- Many-to-Many with Supplier: A supplier supplies multiple products, and a product can be supplied by many suppliers.

6. PurchaseOrder Entity:

- One-to-Many with Product: One purchase order can include multiple products.
- Many-to-One with Supplier: Many purchase orders can be associated with one supplier.
- Many-to-One with Employee: Many purchase orders can be created by one employee.

7. Store Entity:

- One-to-Many with Employee: One store can have multiple employees.
- One-to-Many with Sale: One store can have multiple sales transactions.
- Many-to-Many with Product: A product can be sold in multiple stores, and a store can include multiple products

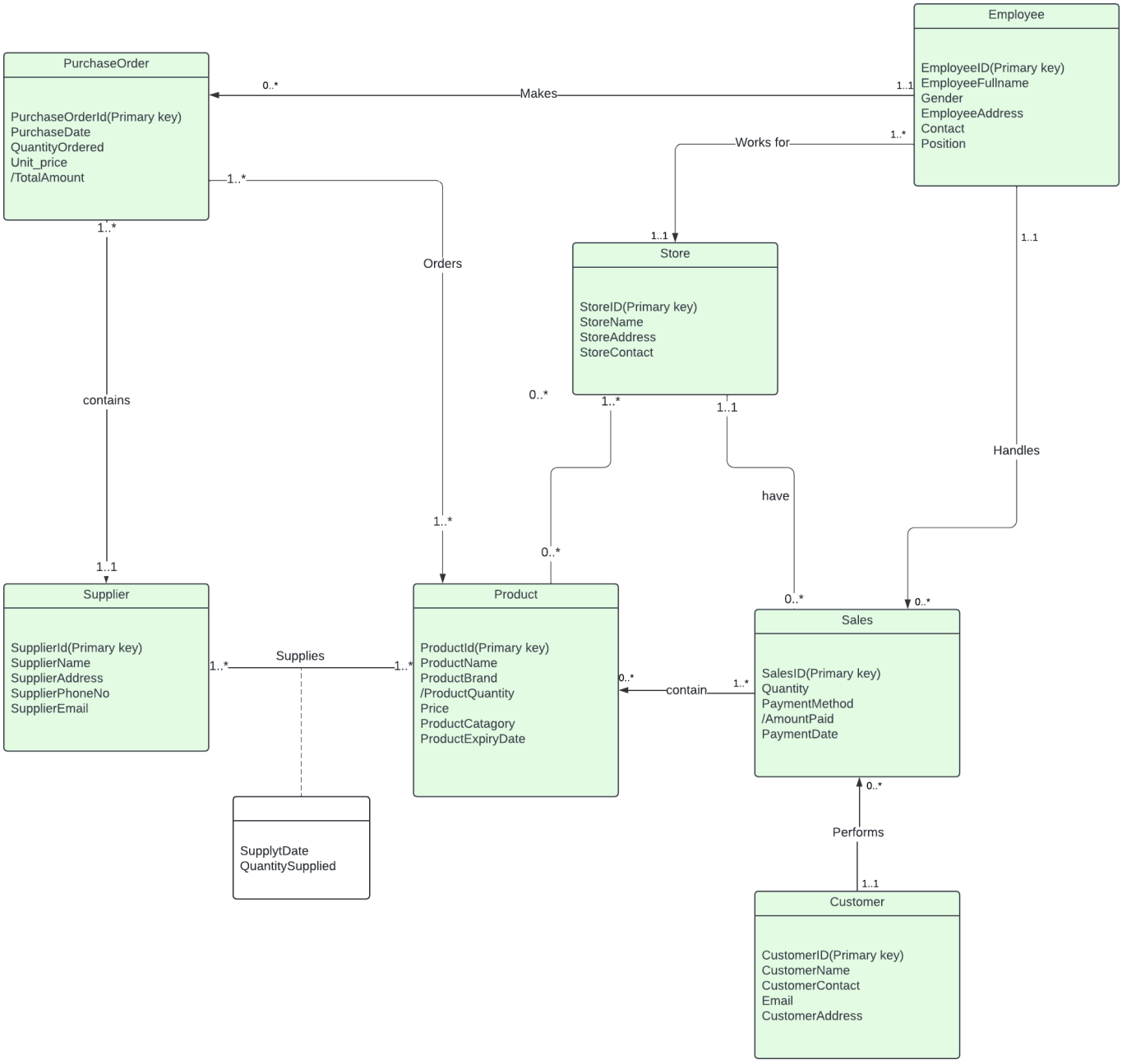
3.2.3. Attributes Identification and Description

1. **Supplier**:
    - **SupplierID**: A unique identifier for each supplier.
    - **SupplierName**: The name of the supplier.
    - **SupplierEmail**: The email address of the supplier.
    - **Address**: The physical address where the supplier operates from.
    - **SupplierContact**: Contact details (phone number, etc.) for the supplier.
2. **Employee**:
    - **EmployeeID**: A unique identifier for each employee.
    - **EmployeeName**: The name of the employee.
    - **Gender**: The gender of the employee (e.g., male, female, non-binary).
    - **Position**: The job position or role of the employee.
    - **Address**: The employee's address.
    - **EmployeeContact**: Contact details for the employee.
3. **Customer**:
    - **CustomerID**: A unique identifier for each customer.
    - **CustomerName**: The name of the customer.
    - **Email**: The email address of the customer.
    - **Address**: The customer's address.
    - **CustomerContact**: Contact details for the customer.
4. **Sale**:
    - **SalesID**: A unique identifier for each sale.
    - **Quantity**: The quantity of products sold.
    - **PaymentMethod**: The method used for payment (e.g., cash, credit card).
    - **AmountPaid**: The total amount paid by the customer.
    - **PaymentDate**: The date when the payment was made.
5. **Product**:
    - **ProductID**: A unique identifier for each product.
    - **ProductName**: The name of the product.
    - **ProductBrand**: The brand associated with the product.

- **ProductQuantity**: The available quantity of the product.
- **Unit_price**: The price per unit of the product.
- **ProductCatagory**: The category or type of the product.
- **ProductExpiryDate**: The expiration date (if applicable) for perishable products.
- **PointsRedeemed**: The loyalty points redeemed by customers.

6. **Purchase Order**:
   - **PurchaseOrderID**: A unique identifier for each purchase order.
   - **OrderDate**: The date when the purchase order was placed.
   - **QuantityOrdered**: The quantity of products ordered.
   - **Unit_price**: The price per unit in the purchase order.
   - **TotalAmount**: The total amount for the entire purchase order.
7. **Store**:
   - **StoreID**: A unique identifier for each store.
   - **StoreName**: The name of the store.
   - **Address**: The physical address of the store.
   - **StoreContact**: Contact details for the store.

3.2.4. ER Diagram



3.3 Logical Database Design

3.3.1 ER-Relation Mapping

Store(StoreID,StoreName,City,subcity,street) **Primary key**: StoreID	**Supplier**(SupplierID, SupplierName, SupplierEmail, City,subcity,street) **Primary key**: SupplierID
Employee(EmployeeID, EmployeeName, Gender, Position, StoreID ,City,subcity,houseNo) **Primary key**: EmployeeID **Foreign key**: StoreID references Store(StoreID)	**Customer**(CustomerID,CustomerName, Email,City,subcity,houseNo) **Primary key**: CustomerID
Sales (SalesID,Quantity, PaymentMethod, AmountPaid, PaymentDate, EmployeeID, CustomerID) **Primary key**: SalesID **Foreign key**: EmployeeID references Employee(EmployeeID) **Foreign key**: CustomerID references Customer(CustomerID)	**Product** (ProductID, ProductName, ProductBrand, ProductQuantity, Unit_price, ProductCatagory, ProductExpiryDate) **Primary key**: ProductID
StoreProduct(StoreID, ProductID,Quantity**)** **Primary key** : StoreID, ProductID **Foreign key**: StoreID references Store(StoreID) **Foreign key**: ProductID references Product(ProductID)	**PurchaseOrder** (PurchaseOrderID, PurchaseDate, QuantityOrdered, Unit_price,TotalAmount, EmployeeID, SupplierID) **Primary key**: PurchaseOrderID **Foreign key**: EmployeeID references Employee(EmployeeID) **Foreign key**: SupplierID references Supply(Supplier)
ProductPurchaseOrder (PurchaseOrderID, ProductID,Quantity**)** **Primary key** : PurchaseOrderID, ProductID **Foreign key**: PurchaseOrderID references PurchaseOrder(PurchaseOrderID) **Foreign key**: ProductID references Product(ProductID)	**ProductSupply**(SupplierID, ProductID, SupplyDate,QuantitySupplied) **Primary key** : SupplierID, ProductID **Foreign key**: SupplierID references Supplier (SupplierID) **Foreign key**: ProductID references Product(ProductID)

StoreContact (StoreID , phone_number) **Primary key** : StoreID **Foreign key**: StoreID references Store(StoreID)	**ProductSales (**SalesID, ProductID,Quantity**)** **Primary key** : SalesID, ProductID **Foreign key**: SalesID references Sales(SalesID) **Foreign key**: ProductID references Product(ProductID)
CustomerContact(CustomerID,phone_number) **Primary key** : CustomerID **Foreign key**: CustomerID references Customer(CustomerID)	**EmployeeContact** (EmployeeID , phone_number) **Primary key** : EmployeeID **Foreign key**: EmployeeID references Employee(EmployeeID)
SupplierContact (SupplierID, phone_number) **Primary key** : SupplierID **Foreign key**: SupplierID references Supplier (SupplierID)	

3.4. The Normalization Process

3.4.1. First Normal Form

The first normal form requires no repeating groups in the table, and
that every attribute value should be atomic. Based on these conditions, we will check to see if the tables satisfy the 1NF.

1. Store Table (Satisfies 1NF):
- *Store*(StoreID, StoreName, City,subcity,street)
- Each attribute contains atomic values. No repeating groups.

2. Employee Table (Satisfies 1NF):
- *Employee*(EmployeeID, EmployeeName, Gender, Position, StoreID ,City,subcity,houseNo)
- Each attribute contains atomic values. No repeating groups.

3. **Customer** Table (Satisfies 1NF):
- Customer(CustomerID, CustomerName, Email, Address)
- Each attribute contains atomic values. No repeating groups.

4. **Supplier** Table (Satisfies 1NF):
- Supplier(SupplierID, SupplierName, SupplierEmail, City,subcity,street)
- Each attribute contains atomic values. No repeating groups.

5. **Product** Table (Satisfies 1NF):
- Product(ProductID, ProductName, ProductBrand, ProductQuantity, Unit_price, ProductCatagory, ProductExpiryDate)
- Each attribute contains atomic values. No repeating groups.

6. **Sales** Table (Satisfies 1NF):
- Sales (SalesID,Quantity, PaymentMethod, AmountPaid, PaymentDate, EmployeeID, CustomerID)
- Each attribute contains atomic values. No repeating groups.

7. **PurchaseOrder** Table (Satisfies 1NF):
   - PurchaseOrder (PurchaseOrderID, PurchaseDate, QuantityOrdered, Unit_price,TotalAmount, EmployeeID, SupplierID)
   - Each attribute contains atomic values. No repeating groups.

8. **StoreProduct** Table (Satisfies 1NF):
   - storeproduct (StoreID, ProductID, Quantity)
   - Each attribute contains atomic values. No repeating groups.

9. **ProductSupply** Table (Satisfies 1NF):
   - ProductSupply(SupplierID, ProductID, SupplyDate, QuantitySupplied)
   - Each attribute contains atomic values. No repeating groups.

10. **ProductPurchaseOrder** Table (Satisfies 1NF):
   - ProductPurchaseOrder(PurchaseOrderID, ProductID, Quantity)
   - Each attribute contains atomic values. No repeating groups.

11. **ProductSales** Table (Satisfies 1NF):
   - ProductSales(SalesID, ProductID, Quantity)
   - Each attribute contains atomic values. No repeating groups.

12. **StoreContact** Table (Satisfies 1NF)
   - StoreContact(StoreID, phone_number)
   - Each attribute contains atomic values. No repeating groups.

13. **EmployeeContact** Table (Satisfies 1NF):
   - EmployeeContact(EmployeeID, phone_number)
   - Each attribute contains atomic values. No repeating groups.

14. **CustomerContact** Table (Satisfies 1NF):
   - CustomerContact(CustomerID, phone_number)
   - Each attribute contains atomic values. No repeating groups.

15. **SupplierContact** Table (Satisfies 1NF):
   - SupplierContact(SupplierID, phone_number)
   - Each attribute contains atomic values. No repeating groups.

3.4.2. Second Normal Form

Requires for all non-key attributes to be fully dependent on the primary key; no partial dependencies and has to be on the first normal form.

1. Store:
- Satisfies 2NF. The table has no partial dependencies, and all non-prime attributes are fully functionally dependent on the primary key (StoreID).

2. Employee:
- Satisfies 2NF. The table has no partial dependencies, and all non-prime attributes are fully functionally dependent on the primary key (EmployeeID).

3. Customer:
- Satisfies 2NF. The table has no partial dependencies, and all non-prime attributes are fully functionally dependent on the primary key (CustomerID).

4. Supplier:
- Satisfies 2NF. The table has no partial dependencies, and all non-prime attributes are fully functionally dependent on the primary key (SupplierID).

5. Product:
- Satisfies 2NF. The table has no partial dependencies, and all non-prime attributes are fully functionally dependent on the primary key (ProductID).

6. Sales:
- Satisfies 2NF. The table has no partial dependencies, and all non-prime attributes are fully functionally dependent on the primary key (SalesID).

7. PurchaseOrder:
- Satisfies 2NF. The table has no partial dependencies, and all non-prime attributes are fully functionally dependent on the primary key (PurchaseOrderID).

8.StoreProduct:
- Satisfies 2NF. The table has no partial dependencies, and all non-prime attributes are fully functionally dependent on the composite primary key (StoreID, ProductID).

9. ProductSupply:
- Satisfies 2NF. The table has no partial dependencies, and all non-prime attributes are fully functionally dependent on the composite primary key (SupplierID, ProductID).

10. ProductPurchaseOrder:

- Satisfies 2NF. The table has no partial dependencies, and all non-prime attributes are fully functionally dependent on the composite primary key (PurchaseOrderID, ProductID).

11. ProductSales:
- Satisfies 2NF. The table has no partial dependencies, and all non-prime attributes are fully functionally dependent on the composite primary key (SalesID, ProductID).

12. StoreContact:
- Satisfies 2NF. The table has no partial dependencies, and all non-prime attributes are fully functionally dependent on the primary key (StoreID).

13. EmployeeContact:
- Satisfies 2NF. The table has no partial dependencies, and all non-prime attributes are fully functionally dependent on the primary key (EmployeeID).

14. CustomerContact:
- Satisfies 2NF. The table has no partial dependencies, and all non-prime attributes are fully functionally dependent on the primary key (CustomerID).

15. SupplierContact:
- Satisfies 2NF. The table has no partial dependencies, and all non-prime attributes are fully functionally dependent on the primary key (SupplierID).

Summary:
All the provided tables satisfy the 2nd Normal Form (2NF). They have no partial dependencies, and all non-prime attributes are fully functionally dependent on their respective primary keys. The structure of the tables supports data integrity and avoids redundancy.

3.4.3. Third Normal Form

The third normal form requires for there to not be any transitive dependencies between non-key attributes and must be on the first and second normal form.

A transitive dependency exists when a non-key attribute depends on another non-key attribute.

1. Store:

Store (StoreID, StoreName, City, Subcity, Street)

- Satisfies 3NF as there are no transitive dependencies, and all non-prime attributes are fully functionally dependent on the primary key.

3. Employee:

Employee (EmployeeID, EmployeeName, Gender, Position, City, Subcity, HouseNo, StoreID)

- Satisfies 3NF as there are no transitive dependencies, and all non-prime attributes are fully functionally dependent on the primary key.

4. Customer:

Customer (CustomerID, CustomerName, Email, City, Subcity, HouseNo)

- Satisfies 3NF as there are no transitive dependencies, and all non-prime attributes are fully functionally dependent on the primary key.

5. Supplier:

Supplier (SupplierID, SupplierName, SupplierEmail, City, Subcity, Street)

- Satisfies 3NF as there are no transitive dependencies, and all non-prime attributes are fully functionally dependent on the primary key.

6. Product:

Product (ProductID, ProductName, ProductBrand, ProductQuantity, Unit_price, ProductCatagory, ProductExpiryDate)

- Satisfies 3NF as there are no transitive dependencies, and all non-prime attributes are fully functionally dependent on the primary key.

7. Sales:

Sales (SalesID, Quantity, PaymentMethod, AmountPaid, PaymentDate, EmployeeID, CustomerID)

- Satisfies 3NF as there are no transitive dependencies, and all non-prime attributes are fully functionally dependent on the primary key.

8. PurchaseOrder:

PurchaseOrder (PurchaseOrderID, OrderDate, QuantityOrdered, Unit_price, TotalAmount, EmployeeID, SupplierID)

- Satisfies 3NF as there are no transitive dependencies, and all non-prime attributes are fully functionally dependent on the primary key.

9.StoreProduct:

storeproduct (**StoreID**, ProductID, Quantity)

- Satisfies 3NF as there are no transitive dependencies, and all non-prime attributes are fully functionally dependent on the primary key.

10. ProductSupply:

ProductSupply (**SupplierID, ProductID**, SupplyDate, QuantitySupplied)

- Satisfies 3NF as there are no transitive dependencies, and all non-prime attributes are fully functionally dependent on the primary key.

11. ProductPurchaseOrder:

ProductPurchaseOrder (**PurchaseOrderID, ProducID,** Quantity)

- Satisfies 3NF as there are no transitive dependencies, and all non-prime attributes are fully functionally dependent on the primary key.

12. ProductSales:

ProductSales (**SalesID, ProductID,** Quantity)

- Satisfies 3NF as there are no transitive dependencies, and all non-prime attributes are fully functionally dependent on the primary key.

13. StoreContact:

StoreContact (**StoreID,** Phone_number)

- Satisfies 3NF as there are no transitive dependencies, and all non-prime attributes are fully functionally dependent on the primary key.

14. EmployeeContact:

EmployeeContact (**EmployeeID**, Phone_number)

- Satisfies 3NF as there are no transitive dependencies, and all non-prime attributes are fully functionally dependent on the primary key.

15. CustomerContact:

CustomerContact (**CustomerID**, Phone_number)

- Satisfies 3NF as there are no transitive dependencies, and all non-prime attributes are fully functionally dependent on the primary key.

16. SupplierContact:

SupplierContact (**SupplierID**, Phone_number)

- Satisfies 3NF as there are no transitive dependencies, and all non-prime attributes are fully functionally dependent on the primary key.

In summary, all the provided tables appear to satisfy the 3rd Normal Form (3NF) as there are no transitive dependencies, and all non-prime attributes are fully functionally dependent on the primary key in each table.

3.5. Referential integrity

Referential integrity refers to the accuracy and consistency of data when it is linked between two or more tables in a database. It ensures that data remains intact across related tables.

PurchaseOrder						
PurchaseOrderID	OrderDate	QuantityOrdered	UnitPrice	TotalAmount	EmployeeID	SupplierID

Supplier					
SupplierID	SupplierName	SupplierEmail	City	Subcity	Street

Loyalty				
LoyaltyID	LoyaltyType	PointsEarned	PointsRedeemed	CustomerID

Employee							
EmployeeID	EmployeeName	Gender	Position	City	Subcity	HouseNo	StoreID

Customer					
CustomerID	CustomerName	Email	City	Subcity	HouseNo

Sales						
SalesID	Quantity	PaymentMethod	AmountPaid	PaymentDate	EmployeeID	CustomerID

Product						
ProductID	ProductName	ProductBrand	ProductQty	UnitPrice	ProductCat	ProductExprDate

Store				
StoreID	StoreName	City	SubCity	Street

StoreProduct		
StoreID	**ProductID**	Quantity

ProductSupply			
SupplierID	**ProductID**	SupplyDate	QuantitySupplied

ProductSales		
SalesID	**ProductID**	Quantity

ProductPurchaseOrder		
SalesID	**ProductID**	Quantity

StoreContact	
StoreID	Phone_number

EmployeeContact	
StoreID	Phone_number

CustomerContact	
CustomerID	Phone_number

SupplierContact	
SupplierID	Phone_number

4. Physical database design of the inventory management system

4.2 Physical Design Strategy

In the final phase of the database design, the designer must decide how to translate the logical database design (that is, the entities, attributes, relationships, and constraints) into a physical database design that can be implemented using the target DBMS. When designing the physical aspect of a database, we need to consider factors like storage, performance, scalability, and security.

There are some key points we included in our physical design strategy:
Data Types: Choose appropriate data types for each column based on the nature of the data it will store. Common data types include integers, strings, dates, and decimals.
Constraints: Define constraints to enforce data integrity and maintain consistency in the database.
Indexing Strategy: Decide which columns to index for faster data retrieval and optimize query performance.
Partitioning: Consider partitioning large tables to improve query performance and manage data more efficiently.
Storage Allocation: Find out what hardware requirements are required to support the database. In order to achieve the required performance and storage capacity.
Backup and Recovery: Implement a backup and recovery strategy to ensure data integrity and availability in case of system failures.

4.3. Database deployment details

Database deployment involves the process of deploying the database schema and data to a production environment. This include also choosing the database management system, servers, storage devices, and networking equipment, needed to support the database.
There are some key details we included in our database deployment plan:
Deployment Environment: Specify the target environment where the database will be deployed.
Database software: Choose a DBMS that aligns with the requirements of our application and supports the desired features and functionalities. Popular DBMS options include MySQL, PostgreSQL, Oracle Database, SQL Server. For this particular project we used Microsoft SQL Server for the physical implementation.
Server: A server is needed to host the database and provide access to the system. The server should have enough storage capacity to store the patient data and be powerful enough to handle the queries and requests from the users.
Network: Networking infrastructure can support the communication between the database server and client applications. This could be a local area network (LAN) or a wide area network (WAN).

Access control: restrict access to authorized users.
Monitoring and Maintenance: Establish monitoring mechanisms to track database performance, usage, and potential issues post-deployment. Configure alerts for critical events (e.g disk space, failed backups).

4.4. Implementation and testing

SQL script for creating the database

```
CREATE DATABASE inventory_management_system;
USE inventory_management_system;
```

SQL Scripts for creating the tables, view, indexes.

```
CREATE TABLE Customer (
CustomerID varchar (10) primary key,
CustomerName varchar (40),
Email varchar (40),
City varchar (30),
Subcity varchar (30),
houseNo varchar (10)
)
CREATE TABLE Store (
StoreID varchar (10) primary key,
StoreName varchar (40),
City varchar (30),
Subcity varchar (30),
Street varchar (30)
)

CREATE TABLE Employee(
EmployeeID varchar(10) primary key,
EmployeeName varchar(40) NOT NULL,
Gender char(1) NOT NULL,
Position varchar(20) NOT NULL,
City  varchar(30) NOT NULL,
Subcity  varchar(30) NOT NULL,
houseNo  varchar(10),
StoreID varchar(10) NOT NULL,
FOREIGN KEY (StoreID) REFERENCES Store(StoreID)
)
CREATE TABLE Product(
ProductId varchar(10) primary key,
ProductName varchar(40) UNIQUE ,
productBrand varchar(25),
```

```
productQuantity int,
ProductCatagory varchar(25),
ProductExpireDate date NOT NULL,
Unit_price decimal(10,2) NOT NULL
)
CREATE TABLE Supplier(
SupplierId varchar(10) primary key,
SupplierName varchar(40),
SupplierEmail varchar(40),
City  varchar(30),
Subcity varchar(30),
Street varchar(30)
)
CREATE TABLE Sales (
SalesID varchar(10) primary key,
Quantity int,
PaymentMethod varchar(25),
AmountPaid DECIMAL(10, 2) NOT NULL,
PaymentDate DATETIME NOT NULL,
EmployeeID varchar(10),
CustomerID varchar(10),
FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID),
FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
)
CREATE TABLE PurchaseOrder(
PurchaseOrderID varchar(10) primary key,
PurchaseDate date,
QuantityOrdered int,
Unit_price decimal(10,2) NOT NULL,
TotalAmount decimal(10,2) NOT NULL,
EmployeeID varchar(10),
SupplierID varchar(10) NOT NULL,
FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID),
FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID)
)
CREATE TABLE StoreProduct(
StoreID VARCHAR(10) NOT NULL,
ProductID VARCHAR(10) NOT NULL,
Quantity INT,
PRIMARY KEY (StoreID, ProductID),
FOREIGN KEY (StoreID) REFERENCES Store(StoreID),
FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
)
CREATE TABLE ProductSupply (
SupplierID VARCHAR(10) NOT NULL,
ProductID VARCHAR(10) NOT NULL,
```

```
SupplyDate DATE,
QuantitySupplied INT,
PRIMARY KEY (SupplierID, ProductID),
FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID),
FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
)
CREATE TABLE ProductPurchaseOrder (
PurchaseOrderID VARCHAR(10) NOT NULL,
ProductID VARCHAR(10) NOT NULL,
Quantity INT,
PRIMARY KEY (PurchaseOrderID, ProductID),
FOREIGN KEY (PurchaseOrderID) REFERENCES PurchaseOrder(PurchaseOrderID),
FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
)
CREATE TABLE ProductSales (
SalesID VARCHAR(10) NOT NULL,
ProductID VARCHAR(10) NOT NULL,
Quantity INT,
PRIMARY KEY (SalesID, ProductID),
FOREIGN KEY (SalesID) REFERENCES Sales(SalesID),
FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
)
CREATE TABLE StoreContact (
StoreID VARCHAR(10) NOT NULL,
Phone_number VARCHAR(15) NOT NULL,
PRIMARY KEY (StoreID),
FOREIGN KEY (StoreID) REFERENCES Store(StoreID)
)
CREATE TABLE EmployeeContact (
EmployeeID VARCHAR(10) NOT NULL,
Phone_number VARCHAR(15) NOT NULL,
PRIMARY KEY (EmployeeID),
FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
)
CREATE TABLE CustomerContact (
CustomerID VARCHAR(10),
Phone_number VARCHAR(15),
PRIMARY KEY (CustomerID),
FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
)
CREATE TABLE SupplierContact (
SupplierID VARCHAR(10) NOT NULL,
Phone_number VARCHAR(15) NOT NULL,
PRIMARY KEY (SupplierID),
FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID)
)
```

```
------ Index for enhancing data retrieval
CREATE INDEX IX_Customer_CustomerName
ON Customer(CustomerName);


CREATE INDEX IX_Product_ProductName
ON Product(ProductName);
CREATE INDEX IX_Sales_SalesId
ON Sales(SalesId);
CREATE INDEX IX_PurchaseOrder_PurchaseDate
ON PurchaseOrder(PurchaseDate);

--------VIEWS
CREATE VIEW Cashier_View AS
SELECT Customer.CustomerID, Customer.CustomerName, Sales.SalesID, Sales.Quantity,
Sales.AmountPaid, Sales.PaymentMethod, Sales.PaymentDate , Product.ProductName
FROM Customer
INNER JOIN Sales ON Customer.CustomerID = Sales.CustomerID
CROSS JOIN Product

CREATE VIEW Manager_View AS
SELECT Employee.EmployeeID, Employee.EmployeeName, Employee.Position,
EmployeeContact.Phone_number AS EmployeePhone , Customer.CustomerID,
Customer.CustomerName, Product.ProductName,
Product.Unit_price, Supplier.SupplierName, SupplierContact.Phone_number AS
SupplierPhone , Sales.AmountPaid
FROM Sales
INNER JOIN Employee
INNER JOIN EmployeeContact ON Employee.EmployeeID = EmployeeContact.EmployeeID
INNER JOIN Store ON Employee.StoreID = Store.StoreID ON Sales.EmployeeID =
Employee.EmployeeID
INNER JOIN Customer ON Sales.CustomerID = Customer.CustomerID
CROSS JOIN Product
CROSS JOIN SupplierContact
INNER JOIN Supplier ON SupplierContact.SupplierID = Supplier.SupplierId
Testing (with sample data)

INSERT INTO Customer (CustomerID, CustomerName, Email, City, Subcity, houseNo)
VALUES
('C001', 'Dereje melaku', 'Derejemelaku@gmail.com', 'addis ababa', 'arada', '3124');

INSERT INTO Store (StoreID, StoreName, City, Subcity, Street)
VALUES
('S001', 'Arada Mart', 'addis ababa', 'arada', 'Adwa street');
```

```
INSERT INTO Employee (EmployeeID, EmployeeName, Gender, Position, City, Subcity,
houseNo, StoreID)
VALUES
('E001', 'Tamrat Zewedu', 'M', ' Cashier', 'addis ababa', 'bole', '1013', 'S001');

INSERT INTO Product (ProductID, ProductName, productBrand, productQuantity,
ProductCatagory, ProductExpireDate, Unit_price)
VALUES
('P001', 'lotion', 'nivea', 150, 'skin care', '2026-01-17', 173.00);

INSERT INTO Supplier (SupplierId, SupplierName, SupplierEmail, City, Subcity, Street)
VALUES
('SUP001', 'Belete Haile', 'Beletehaile@gmail.com', 'addis ababa', 'kirkos', 'Gabon street');

INSERT INTO Sales (SalesID, Quantity, PaymentMethod, AmountPaid, PaymentDate,
EmployeeID, CustomerID)
VALUES
('SA001', 5, 'Credit Card', 865.00, '2024-02-01 08:15:23', 'E001', 'C001');

INSERT INTO PurchaseOrder (PurchaseOrderID, PurchaseDate, QuantityOrdered,
Unit_price, TotalAmount, EmployeeID, SupplierID)
VALUES
('PO001', '2024-01-15', 20, 9.99, 199.80, 'E001', 'SUP001');

INSERT INTO StoreContact (StoreID, Phone_number)
VALUES
('S001', '+251965410283');

INSERT INTO EmployeeContact (EmployeeID, Phone_number)
VALUES
('E001', '+251936252113');

INSERT INTO CustomerContact (CustomerID, Phone_number)
VALUES
('C001', '+251925320132');

INSERT INTO SupplierContact (SupplierID, Phone_number)
VALUES
('SUP001', '+251946383215');
```