Minor Project Report

on

Project Title: - Digital Record Integration

Course: Advanced Database Techniques

Course Code: CAP570

Submitted by

1. Piyush Raj (12405997), Group: - 10

Submitted to

Dr. Heena Khanna
UID: 31463
Assistant Professor, SCA, LPU

Lovely Faculty of Technology & Sciences

School of Computer Applications

Lovely Professional University

Punjab

Acknowledgment

We would like to express our sincere gratitude to **Dr. Heena Khanna (31463), Assistant Professor, School of Computer Applications, Lovely Professional University**, for her guidance and continuous supervision, as well as for providing essential information and unwavering support throughout the project. Her consistent direction and willingness to share her extensive knowledge enabled us to gain a deep understanding of the project, which greatly assisted us in completing our tasks on time.

We would also like to extend our heartfelt thanks to all the individuals who graciously participated in the interviews and surveys conducted during the project. Their experiences and insights provided a practical understanding of how database systems are used in real-world applications, which greatly enriched the quality of this project.

Finally, we would like to thank our families and friends for their constant encouragement and support during this journey. This project would not have been possible without the contribution of everyone involved.

Thank you all.

Table of Contents

1. Introduction

➔ Overview of the Digital Record Integration:

A Digital Record Integration (DRI) project in the context of Database Management Systems (DBMS) typically involves the consolidation and integration of various digital records or data sources into a unified database. This project aims to streamline data management processes, enhance data accessibility, and improve decision-making within an organization.

The DRI initiative aims to create a robust and efficient relational database for supply chain and inventory operations. This project focuses on integrating critical data elements, such as user profiles, sales transactions, and product inventory, to streamline data management and analysis. By leveraging a well-structured relational schema and applying normalization techniques, the database ensures data integrity, minimizes redundancy, and supports real-time decision-making. The ultimate goal is to enhance operational efficiency and enable data-driven strategies for inventory optimization and supply chain management.

➔ Importance of Digital Record Integration

- **Improved Efficiency**: Digital integration of records ensures that data flows seamlessly across systems, eliminating the need for manual data entry, reducing errors, and speeding up processes. This is especially valuable in industries like supply chain management, healthcare, finance, and education, where accurate, timely information is critical.
- **Real-Time Data Access**: Digital records can be accessed in real-time, allowing for faster decision-making. For example, inventory management can be improved by integrating sales data and stock levels, providing a real-time view of available products and enabling quicker restocking.
- **Data Accuracy and Consistency**: When records are integrated across digital platforms, it ensures that the same data is reflected consistently across all systems. This minimizes discrepancies and errors, which is vital in areas like finance, legal documentation, and supply chain tracking.
- **Better Collaboration**: Integration allows different departments and teams to access the same records, improving collaboration. For instance, the inventory team can work seamlessly with sales and procurement departments, leading to more coordinated efforts and better outcomes.
- **Cost Reduction**: By automating data integration, businesses can reduce operational costs associated with manual data entry, physical records management, and slow communication between departments. This also leads to less redundancy in storing information.
- **Scalability**: As businesses grow, managing records manually or in silos becomes increasingly difficult. Digital integration allows companies to scale their operations while maintaining data integrity, ensuring the systems can handle increased volumes of data without significant inefficiencies.

2. Project Overview

The core product of the Digital Record Integration is a sophisticated relational database designed to optimize and streamline the management of supply chain and inventory data for Adventure Works. This database serves as a centralized repository, integrating essential components such as user information, sales records, and detailed product data.

Key features of the database include:

- **System Integration**: Connects multiple data sources (CRM, ERP, inventory systems) into a unified platform, eliminating inefficiencies and data silos.

- **Digitization**: Converts paper records into searchable digital formats, reducing physical storage needs and improving data accessibility.

- **Centralized Platform**: Establishes a single database with standardized formats, making it easier to search, retrieve, and analyse information.

- **Real-Time Updates**: Implements data synchronization, ensuring all systems reflect the latest information instantly for accurate decision-making.

- **Security & Compliance**: Enhances data protection through encryption and access controls, meeting industry regulations and reducing risk.

- **Workflow Automation**: Automates approval processes and notifications, minimizing manual effort and errors.

- **Advanced Search**: Offers full-text search and metadata tagging, providing a fast, user-friendly way to locate records.

- **User-Friendly Interface**: Modern design with intuitive navigation, increasing adoption and productivity. Overall, the product is designed to improve data accuracy, facilitate quick decision-making, and support the company's operational growth by providing a scalable and efficient data management solution.

3. Project Objectives

1. **Centralize Data Management**

- **Goal**: Consolidate all records into a single, cohesive system that integrates various data sources, including customer relationship management (CRM) tools, enterprise resource planning (ERP) systems, inventory management platforms, and any other relevant databases.
- **Details**: Currently, organizations often face challenges due to records being stored in multiple, disconnected systems. The DRI project will create a centralized infrastructure that unifies data, making management simpler and more effective.
- **Impact**: This centralization will eliminate inefficiencies, reduce redundancy, and provide a comprehensive view of all organizational data, making it easier for departments to work together.

2. **Enhance Data Accuracy**

- **Goal**: Implement mechanisms to ensure that records are accurate, consistent, and up-to-date across all integrated systems.
- **Details**: Disconnected systems often result in data inconsistencies, such as incorrect or outdated records. The DRI project will synchronize data in real-time, ensuring that updates in one system are automatically reflected across others.
- **Impact**: By maintaining high data integrity, the organization will reduce errors that can lead to costly mistakes, improve reporting accuracy, and enable more reliable analytics.

3. **Improve Accessibility**

- **Goal**: Ensure that authorized personnel can quickly and securely access records from anywhere, using a streamlined interface that supports efficient data retrieval.
- **Details**: Traditional record management systems often make data difficult to find, especially when stored in various formats or locations. The DRI project will standardize data formats and implement advanced search features like metadata tagging and full-text search.
- **Impact**: Enhanced accessibility will significantly reduce the time employees spend searching for information, leading to greater productivity and faster decision-making.

.

4. **Facilitate Collaboration**

- **Goal**: Promote seamless information sharing and collaboration across different teams and departments.

- **Details**: A centralized record system will break down communication barriers between teams, enabling easy and secure data sharing. Features like shared dashboards, notifications, and collaborative workflows will further enhance team coordination.
- **Impact**: Improved collaboration will lead to faster project completion, better alignment on strategic goals, and a more cohesive work environment.

5. **Reduce Costs**

- **Goal**: Minimize expenses related to physical storage, paper-based processes, and inefficient data management systems.
- **Details**: The DRI project will digitize paper records, eliminating the need for physical storage space and associated costs (e.g., paper, printing, and filing cabinets). Additionally, operational efficiencies gained from automation will reduce labour costs.
- **Impact**: Cost savings from reduced paper usage and improved process efficiency will enhance the organization's bottom line, making it more financially sustainable.

6. **Support Scalability**

- **Goal**: Build a flexible and scalable system that can accommodate the organization's growth and increasing data demands.
- **Details**: As the organization expands, the volume of records and the number of users will increase. The DRI project will ensure that the system is built with scalability in mind, using technologies that allow for easy scaling without performance degradation.
- **Impact**: A scalable system will future-proof the organization's record management infrastructure, making it adaptable to evolving business needs without requiring significant additional investment or rework.

4. Problem Statement

Organizations today face significant challenges in managing, accessing, and protecting their records due to outdated and fragmented systems. These challenges lead to inefficiencies, data inaccuracies, and increased operational costs. Below are the key problems that the Digital Record Integration project aims to address:

1. **Data Silos and Fragmented Systems**
    - **Issue**: Records are often stored across multiple, disconnected systems (e.g., CRM, ERP, and document repositories), making it difficult for departments to share or access information seamlessly. This lack of integration creates silos that hinder collaboration and make it challenging to have a comprehensive view of organizational data.
    - **Impact**: Poor communication between departments, inefficiencies in workflow management, and delays in decision-making due to a lack of accessible, consolidated information.
2. **Inefficient Manual Processes**
    - **Issue**: Many record management tasks, such as data entry, record retrieval, and approval workflows, are performed manually. These processes are time-consuming, labour-intensive, and prone to human error.
    - **Impact**: Reduced productivity, increased likelihood of errors, and higher labour costs due to repetitive, inefficient tasks.
3. **Dependence on Paper-Based Records**
    - **Issue**: A significant portion of organizational records is still maintained in paper form. This not only occupies physical space but also makes it difficult to search for and retrieve documents quickly.
    - **Impact**: Slow information retrieval, increased storage costs, and higher risk of data loss or damage due to physical factors like fire, water, or human mishandling.
4. **Inconsistent and Outdated Data**
    - **Issue**: Disconnected systems often lead to data inconsistencies, where updates made in one system are not reflected in others. This results in outdated or inaccurate records, affecting reporting, analysis, and operational decisions.
    - **Impact**: Inaccurate data leads to poor decision-making, compliance risks, and a loss of confidence among stakeholders who rely on timely and correct information.
5. **Security and Compliance Risks**
    - **Issue**: Traditional record-keeping systems may not provide adequate security features to protect sensitive information from unauthorized access or data breaches. Additionally, meeting compliance requirements (e.g., GDPR, HIPAA) is difficult with unstructured or poorly managed records.
    - **Impact**: Higher risk of data breaches, legal and regulatory non-compliance, and potential financial and reputational damage to the organization.
6. **Slow and Complex Data Retrieval**
    - **Issue**: Searching for specific records is often a slow and complex process, especially when data is stored in multiple formats or scattered across different locations. Legacy systems often lack advanced search functionality, making it cumbersome to find needed information.
    - **Impact**: Delayed decision-making, inefficiencies in responding to time-sensitive requests, and a general decrease in employee productivity.

5. Requirement Gathering :-

Requirement gathering is an essential phase that involves collecting detailed information about the organization's needs, current challenges, and expectations for the new digital record system. This phase ensures that the DRI project is tailored to meet user needs and resolve existing inefficiencies.

5.1 Interviews and Surveys

- **Interviews -**

- **Purpose**:
  - To gather qualitative, in-depth insights from key stakeholders who directly interact with record management systems.
  - To understand the processes, bottlenecks, and security concerns from a user's perspective.
- **Participants**:
  - **Records Managers**: Responsible for overseeing the storage, access, and security of organizational records.
  - **IT Administrators**: Experts who manage the technical infrastructure, providing insights into the technical limitations and integration needs.
  - **Department Heads and End Users**: Employees who frequently access records, providing feedback on their day-to-day challenges.
- **Method**:
  - **One-on-One Interviews**: Conducted with participants from different departments. Each session lasted 30-45 minutes, with prepared questions but room for follow-up queries based on responses.
  - **Interview Questions**:
    - **Current System**: "Can you describe how you currently store and access records in your department?"
    - **Challenges**: "What are the main frustrations or inefficiencies you experience with the current record management process?"
    - **Security**: "Do you have concerns about the security of your records? If so, what are they?"
    - **Desired Features**: "What improvements or new features would you like to see in a new digital record system?"
- **Key Findings from Interviews**:
  - **Data Retrieval Issues**: Many interviewees reported that searching for records was cumbersome and time-consuming, particularly when data was spread across different systems or formats.
  - **Manual Processes**: Participants highlighted the extensive time and effort needed for manual data entry and updates, which often resulted in errors and duplicated work.
  - **Lack of Integration**: There was a common complaint about disconnected systems, which made data sharing between departments inefficient and slowed down decision-making.
  - **Security Concerns**: IT administrators emphasized the risk of unauthorized access and outdated security protocols in the current setup, stressing the need for encryption and role-based access controls in the new system.
  - **User-Friendliness**: Employees expressed a desire for a more intutive and easy-to-use interface, as current systems often required extensive training and technical knowledge.

➢ **Surveys -**

- **Purpose**:
  - To collect quantitative data and gather broader feedback from a larger group of employees across different departments.
  - To understand common issues and prioritize features based on user preferences.
- **Participants**:
  - A diverse group of employees, including those who use the current record management systems frequently and those who use them occasionally. The goal was to ensure representation from various levels of the organization.
- **Method**:
  - **Survey Design**: The survey included a mix of multiple-choice questions, rating scales, and open-ended questions to capture both structured and qualitative data.
  - **Sample Survey Questions**:
    - **Satisfaction Levels**: "On a scale from 1 to 10, how satisfied are you with the current record management system?"
    - **Frequency of Issues**: "How often do you experience delays or difficulties in accessing records (e.g., daily, weekly, monthly)?"
    - **Preferred Features**: "Which of the following features would improve your efficiency? (Select all that apply: advanced search, automation, real-time updates, secure access)."
    - **Security and Compliance**: "Do you feel the current system adequately protects sensitive information?"
    - **Open Feedback**: "Please share any additional thoughts or suggestions for a new record management system."
- **Key Findings from Surveys**:
  - **Low Satisfaction Rates**: Many respondents rated their satisfaction with the current system as 5 or below, citing inefficiency and difficulty in navigating multiple platforms.
  - **High Frequency of Issues**: Over 60% of respondents indicated they encountered difficulties accessing records at least once a week, leading to frustration and delays in their work.
  - **Feature Preferences**: The most desired features were advanced search capabilities (selected by 80% of respondents), automation of routine tasks (70%), and real-time data synchronization (65%).
  - **Security Concerns**: A significant portion of respondents expressed concerns about data security, emphasizing the need for better compliance measures and access controls.

5.2 Website Exploration

Objective :-

The primary objective of creating a database with these tables is to facilitate efficient and scalable management of products, shopkeepers, customers, and sales transactions for a web-based application. The database allows for:

- **Inventory Management**: Real-time monitoring and updating of product quantities and availability.
- **Sales Tracking**: Recording and analyzing sales data for various shopkeepers and product categories.
- **Customer Management**: Storing and managing customer contact details, preferences, and transaction history.
- **Shopkeeper Information**: Managing shopkeeper profiles, including their shop details and products.

Website Exploration :-

The website would use these database tables to create different functional sections that support the operations of the e-commerce platform. Here's how each table can be integrated into the site:

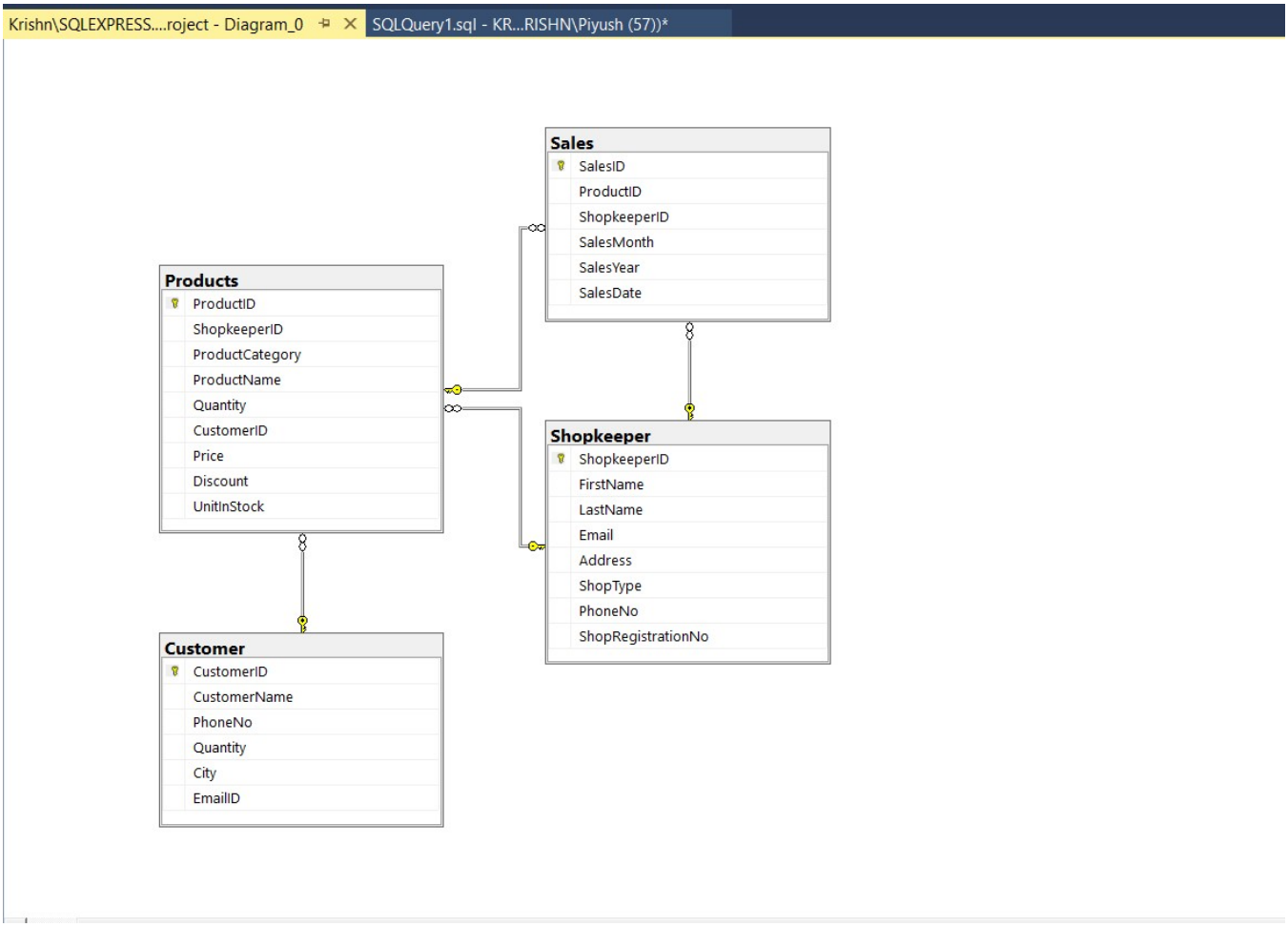
1. **Shopkeeper Management Interface**
    - **Features**: View, add, update, and delete shopkeeper details. Shopkeepers can log in and manage their profiles and the products they sell.
2. **Product Catalog and Management**
    - **Features**: Customers can browse products by category, see details like price, discounts, and stock availability. Shopkeepers can add or update product listings.
3. **Sales Dashboard**
    - **Features**: Shopkeepers and admin users can view and analyze sales trends over time. Information about sales transactions, including dates, customers, and products sold, is shown.
4. **Customer Account and Shopping Experience**
    - **Features**: Customers can create accounts, browse and purchase products, view their purchase history, and update contact details.

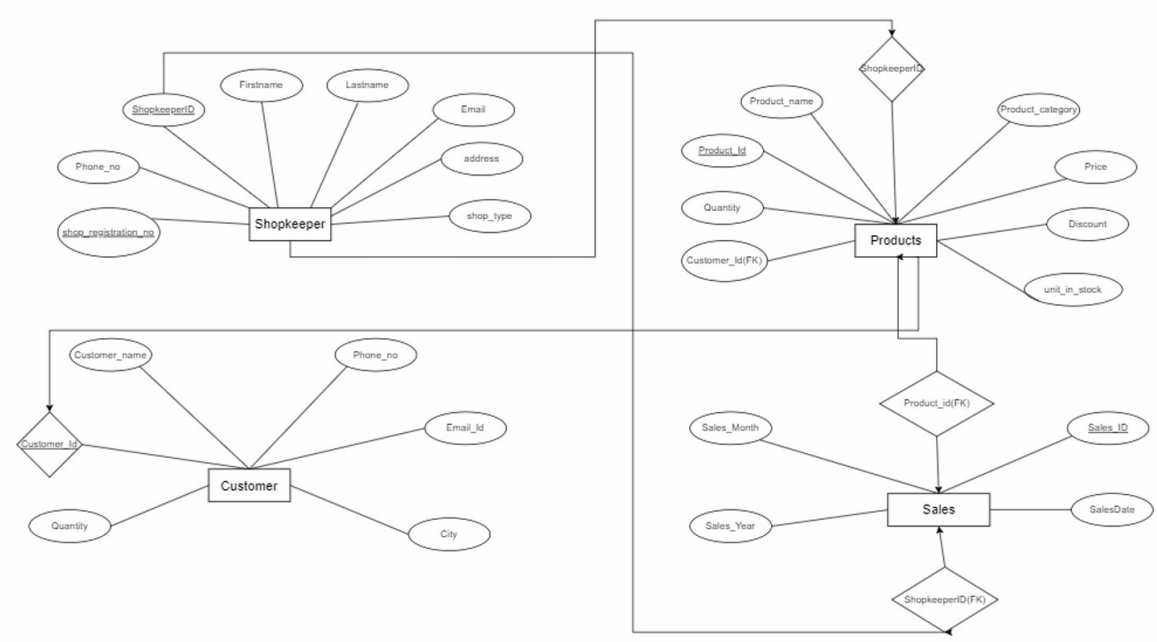
Observations :-

1. **Data Integration**: The integration of these tables ensures a seamless flow of data between various components of the website. For instance, sales updates automatically adjust product quantities.
2. **Performance and Scalability**: Using a relational database structure supports efficient querying, which is crucial for a website handling large numbers of transactions.
3. **User Experience**: The well-structured data allows the website to deliver personalized experiences, such as suggesting products based on past purchases or providing shopkeepers with tailored sales reports.
4. **Data Security**: Ensuring unique email addresses and phone numbers for customers and shopkeepers adds a layer of data integrity and security.

6. ER Diagram and Relational Schema

- **ER Diagram:** Include the Entity-Relationship Diagram that represents all entities (users, content, subscriptions, etc.) and relationships between them.
- **Explanation of ER Diagram:** Describe how entities are connected and what attributes are part of each entity.





Relational Schema:

The relationship schema effectively captures how customers, shopkeepers, products, and sales interact within the database. The use of primary and foreign keys establishes clear connections between the entities, facilitating data integrity and relational operations

1. **Customer**
    - **Primary Key:** CustomerID
    - **Attributes:**
        - CustomerName
        - PhoneNo
        - Quantity
        - City
        - EmailID
2. **Shopkeeper**
    - **Primary Key:** ShopkeeperID
    - **Attributes:**
        - FirstName
        - LastName
        - Email
        - Address
        - ShopType
        - PhoneNo
        - ShopRegistrationNo
3. **Products**
    - **Primary Key:** ProductID
    - **Foreign Key:** ShopkeeperID (references ShopkeeperID in Shopkeeper)

- **Attributes:**
  - ProductCategory
  - ProductName
  - Quantity
  - CustomerID (nullable, references CustomerID in Customer)
  - Price
  - Discount
  - UnitInStock

4. **Sales**
   - **Primary Key:** SalesID
   - **Foreign Keys:**
     - ProductID (references ProductID in Products)
     - ShopkeeperID (references ShopkeeperID in Shopkeeper)
   - **Attributes:**
     - SalesMonth
     - SalesYear
     - SalesDate

Relationships

- **Customer to Products**:
  - **Relationship Type:** Many-to-Many
  - **Description:** A customer can purchase multiple products, and a product can be purchased by multiple customers. This relationship is represented through the CustomerID in the **Products** table.
- **Shopkeeper to Products**:
  - **Relationship Type:** One-to-Many
  - **Description:** A shopkeeper can offer multiple products, but each product is associated with only one shopkeeper.
- **Shopkeeper to Sales**:
  - **Relationship Type:** One-to-Many
  - **Description:** A shopkeeper can have multiple sales transactions, but each sale is linked to only one shopkeeper.
- **Products to Sales**:
  - **Relationship Type:** One-to-Many
  - **Description:** A product can be sold in multiple sales transactions, but each sale corresponds to only one product.

7. Database Design and Normalization

7.1 Table Descriptions

1. **Products**
    - **Attributes:**
        - **ProductID** (Primary Key): Unique identifier for each product.
        - **ShopkeeperID** (Foreign Key): References the Shopkeeper entity, indicating which shopkeeper offers the product.
        - **ProductCategory**: The category or type of the product.
        - **ProductName**: Name of the product.
        - **Quantity**: Number of items available.
        - **CustomerID** (Foreign Key): Indicates which customer purchased the product (nullable).
        - **Price**: Cost of the product.
        - **Discount**: Any price reductions applicable to the product.
        - **UnitInStock**: Current stock level.
    - **Primary key:**
        - **ProductID**
    - **Foreign key:**
        - **ShopkeeperID, CustomerID.**

2. **Sales**
    - **Attributes:**
        - **SalesID** (Primary Key): Unique identifier for each sale transaction.
        - **ProductID** (Foreign Key): References the ProductID in the Products entity, linking each sale to a specific product.
        - **ShopkeeperID** (Foreign Key): References the ShopkeeperID in the Shopkeeper entity, linking each sale to the shopkeeper who sold the product.
        - **SalesMonth**: Month in which the sale occurred.
        - **SalesYear**: Year of the sale.
        - **SalesDate**: Exact date of the transaction.
3. **Shopkeeper**
    - **Attributes:**
        - **ShopkeeperID** (Primary Key): Unique identifier for each shopkeeper.
        - **FirstName**: First name of the shopkeeper.
        - **LastName**: Last name of the shopkeeper.
        - **Email**: Contact email for the shopkeeper.
        - **Address**: Physical address of the shopkeeper.
        - **ShopType**: Type of shop (e.g., grocery, electronics).
        - **PhoneNo**: Contact number for the shopkeeper.
        - **ShopRegistrationNo**: Unique registration number for the shop.
4. **Customer**

- **Attributes:**
  - **CustomerID** (Primary Key): Unique identifier for each customer.
  - **CustomerName**: The name of the customer.
  - **PhoneNo**: Contact number for the customer.
  - **Quantity**: Number of products purchased by the customer.
  - **City**: City where the customer resides.
  - **EmailID**: Contact email for the customer.

Relationships

1. **Products to Shopkeeper**:
   - **Type**: One-to-Many
   - **Description**: A single shopkeeper can offer multiple products (one shopkeeper can sell many products). The connection is made through ShopkeeperID.
2. **Sales to Products**:
   - **Type**: One-to-Many
   - **Description**: A product can be involved in multiple sales transactions, but each sale refers to only one product. This relationship is established via ProductID.
3. **Sales to Shopkeeper**:
   - **Type**: One-to-Many
   - **Description**: Each sale is associated with only one shopkeeper, but a shopkeeper can have many sales transactions. This is represented by the ShopkeeperID.
4. **Products to Customer**:
   - **Type**: Many-to-One
   - **Description**: While this relationship is represented with CustomerID in the Products table, it indicates that a product can be purchased by many customers, but each product in a specific context of sale (in the Sales table) is linked to one customer. However, CustomerID can sometimes be nullable, implying not all products are linked directly to a customer during sale (this may depend on how transactions are recorded).

7.2 Normalization Process

- **Customer to Products**:
  - **Relationship Type:** Many-to-Many
  - **Description:** A customer can purchase multiple products, and a product can be purchased by multiple customers. This relationship is represented through the CustomerID in the **Products** table.
- **Shopkeeper to Products**:
  - **Relationship Type:** One-to-Many
  - **Description:** A shopkeeper can offer multiple products, but each product is associated with only one shopkeeper.
- **Shopkeeper to Sales**:
  - **Relationship Type:** One-to-Many

    - **Description:** A shopkeeper can have multiple sales transactions, but each sale is linked to only one shopkeeper.
- **Products to Sales**:
    - **Relationship Type:** One-to-Many
    - **Description:** A product can be sold in multiple sales transactions, but each sale corresponds to only one product.

8. SQL Queries and Database Operations

8.1 Create Tables

Provide SQL scripts for creating all tables.

Table 01 – Shopkeeper

```
CREATE TABLE Shopkeeper (
    ShopkeeperID INT PRIMARY KEY IDENTITY(1,1),
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) NOT NULL UNIQUE,
    Address VARCHAR(255),
    ShopType VARCHAR(50),
    PhoneNo NVARCHAR(15) NOT NULL UNIQUE,
    ShopRegistrationNo NVARCHAR(50) UNIQUE
);
```

Table 02 – Products

```
CREATE TABLE Products (
    ProductID INT PRIMARY KEY IDENTITY(1,1),
    ShopkeeperID INT NOT NULL,
    ProductCategory NVARCHAR(50),
    ProductName NVARCHAR(100) NOT NULL,
    Quantity INT NOT NULL,
    CustomerID INT,
    Price DECIMAL(10, 2) NOT NULL,
    Discount DECIMAL(5, 2),
    UnitInStock INT NOT NULL,
    CONSTRAINT FK_Shopkeeper FOREIGN KEY (ShopkeeperID) REFERENCES Shopkeeper(ShopkeeperID),
    CONSTRAINT FK_Customer FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
);
```

Table 03 – Sales

```
CREATE TABLE Sales (
    SalesID INT PRIMARY KEY IDENTITY(1,1),
    ProductID INT NOT NULL,
    ShopkeeperID INT NOT NULL,
    SalesMonth INT CHECK (SalesMonth >= 1 AND SalesMonth <= 12),
    SalesYear INT,
    SalesDate DATE DEFAULT GETDATE(),
    CONSTRAINT FK_Product FOREIGN KEY (ProductID) REFERENCES Products(ProductID),
    CONSTRAINT FK_SalesShopkeeper FOREIGN KEY (ShopkeeperID) REFERENCES Shopkeeper(ShopkeeperID)
);
```

Table 04 – Customer

```
CREATE TABLE Customer (
    CustomerID INT PRIMARY KEY IDENTITY(1,1),
    CustomerName NVARCHAR(100) NOT NULL,
    PhoneNo NVARCHAR(15) NOT NULL UNIQUE,
    Quantity INT,
    City NVARCHAR(50),
    EmailID NVARCHAR(100) UNIQUE
);
```

8.2 Insert Data

Show SQL commands for inserting data into each table.

Insert data into table Shopkeeper

```
INSERT INTO Shopkeeper (FirstName, LastName, Email, Address, ShopType, PhoneNo, ShopRegistrationNo)
VALUES
('John', 'Doe', 'john.doe@example.com', '123 Elm St, Springfield', 'Grocery', '1234567890', 'REG123456'),
('Jane', 'Smith', 'jane.smith@example.com', '456 Maple Ave, Riverdale', 'Clothing', '0987654321', 'REG987654'),
('Michael', 'Johnson', 'michael.johnson@example.com', '789 Oak Dr, Gotham', 'Electronics', '2345678901', 'REG234567'),
('Emily', 'Williams', 'emily.williams@example.com', '135 Pine St, Metropolis', 'Bakery', '3456789012', 'REG345678'),
('David', 'Brown', 'david.brown@example.com', '246 Birch Rd, Star City', 'Furniture', '4567890123', 'REG456789'),
('Sophia', 'Jones', 'sophia.jones@example.com', '357 Cedar Ln, Central City', 'Pharmacy', '5678901234', 'REG567890'),
('Daniel', 'Garcia', 'daniel.garcia@example.com', '468 Willow St, Coast City', 'Hardware', '6789012345', 'REG678901'),
('Isabella', 'Martinez', 'isabella.martinez@example.com', '579 Ash Blvd, Bludhaven', 'Bookstore', '7890123456', 'REG789012'),
('James', 'Lopez', 'james.lopez@example.com', '680 Oak Ave, Smallville', 'Jewelry', '8901234567', 'REG890123'),
('Olivia', 'Gonzalez', 'olivia.gonzalez@example.com', '791 Sycamore St, Keystone City', 'Footwear', '9012345678', 'REG901234'),
('Matthew', 'Lee', 'matthew.lee@example.com', '902 Poplar Ln, National City', 'Grocery', '0123456789', 'REG012345'),
('Charlotte', 'Clark', 'charlotte.clark@example.com', '123 Chestnut St, Hub City', 'Clothing', '1238904567', 'REG135790'),
('Lucas', 'Kim', 'lucas.kim@example.com', '456 Walnut St, Hill Valley', 'Electronics', '7892340123', 'REG975321'),
('Amelia', 'Davis', 'amelia.davis@example.com', '789 Redwood Dr, Greenfield', 'Furniture', '6543210987', 'REG246801'),
('Ethan', 'Morris', 'ethan.morris@example.com', '135 Cedar Dr, River City', 'Bakery', '3450126789', 'REG112233');
```

Insert data into table Products

```
INSERT INTO Products (ShopkeeperID, ProductCategory, ProductName, Quantity, CustomerID, Price, Discount, UnitInStock)
VALUES
(1, 'Grocery', 'Organic Apple', 50,1, 2.50, 0.00, 500),
(2, 'Clothing', 'Men's Cotton T-shirt', 100,1, 15.99, 5.00, 150),
(3, 'Electronics', 'Wireless Headphones', 30,2, 45.99, 10.00, 70),
(4, 'Bakery', 'Chocolate Cake', 10,3, 12.00, 0.50, 40),
(5, 'Furniture', 'Wooden Coffee Table', 5,5, 120.00, 15.00, 10),
(6, 'Pharmacy', 'Vitamin D Supplements', 200,8, 9.99, 1.00, 300),
(7, 'Hardware', 'Hammer', 50,6, 14.99, 2.00, 100),
(8, 'Bookstore', 'Fiction Novel', 70,7, 18.50, 0.00, 120),
(9, 'Jewelry', 'Gold Necklace', 5,9, 499.99, 25.00, 7),
(10, 'Footwear', 'Running Shoes', 40,4, 60.00, 10.00, 90),
(1, 'Grocery', 'Almond Milk', 100,3, 3.99, 0.00, 250),
(2, 'Clothing', 'Women's Dress', 50,5, 29.99, 7.00, 80),
(3, 'Electronics', 'Smartphone', 20,7, 599.99, 50.00, 30),
(4, 'Bakery', 'Gluten-Free Bread', 25,8, 4.50, 0.20, 60),
(5, 'Furniture', 'Office Chair', 15,10, 75.00, 5.00, 25);
```

Insert data into table Sales

```sql
INSERT INTO Sales (ProductID, ShopkeeperID, SalesMonth, SalesYear, SalesDate)
VALUES
(1, 1, 1, 2024, '2024-01-15'),
(2, 2, 2, 2024, '2024-02-20'),
(3, 3, 3, 2024, '2024-03-10'),
(4, 4, 4, 2024, '2024-04-05'),
(5, 5, 5, 2024, '2024-05-25'),
(6, 6, 6, 2024, '2024-06-12'),
(7, 7, 7, 2024, '2024-07-18'),
(8, 8, 8, 2024, '2024-08-22'),
(9, 9, 9, 2024, '2024-09-30'),
(10, 10, 10, 2024, '2024-10-15'),
(11, 1, 11, 2024, '2024-11-05'),
(12, 2, 12, 2024, '2024-12-19'),
(13, 3, 1, 2025, '2025-01-11'),
(14, 4, 2, 2025, '2025-02-07'),
(15, 5, 3, 2025, '2025-03-03');
```

Insert data into table Customer

```sql
INSERT INTO Customer (CustomerName, PhoneNo, Quantity, City, EmailID)
VALUES
('Alice Johnson', '9876543210', 3, 'Springfield', 'alice.johnson@example.com'),
('Bob Smith', '8765432109', 5, 'Riverdale', 'bob.smith@example.com'),
('Charlie Brown', '7654321098', 2, 'Gotham', 'charlie.brown@example.com'),
('Diana Prince', '6543210987', 1, 'Metropolis', 'diana.prince@example.com'),
('Ethan Hunt', '5432109876', 4, 'Star City', 'ethan.hunt@example.com'),
('Fiona Davis', '4321098765', 6, 'Central City', 'fiona.davis@example.com'),
('George Miller', '3210987654', 2, 'Coast City', 'george.miller@example.com'),
('Hannah Lee', '2109876543', 3, 'Bludhaven', 'hannah.lee@example.com'),
('Ian Thompson', '1098765432', 1, 'Smallville', 'ian.thompson@example.com'),
('Jessica Williams', '9087654321', 5, 'Keystone City', 'jessica.williams@example.com'),
('Kyle Reed', '8098765432', 4, 'National City', 'kyle.reed@example.com'),
('Lily Clark', '7098765431', 2, 'Hub City', 'lily.clark@example.com'),
('Mason Evans', '6098765430', 7, 'Hill Valley', 'mason.evans@example.com'),
('Natalie Green', '5098765429', 3, 'Greenfield', 'natalie.green@example.com'),
('Owen Harris', '4098765428', 2, 'River City', 'owen.harris@example.com'),
('Paige Martin', '3098765427', 5, 'Springfield', 'paige.martin@example.com'),
('Quinn Turner', '2098765426', 1, 'Riverdale', 'quinn.turner@example.com'),
('Ryan Scott', '1098765425', 6, 'Gotham', 'ryan.scott@example.com'),
('Sophia Carter', '9087654320', 4, 'Metropolis', 'sophia.carter@example.com'),
('Tyler Adams', '8087654321', 2, 'Star City', 'tyler.adams@example.com');
```

8.3 Queries for Data Retrieval

Provide queries to demonstrate CRUD operations.

Query for Maximum Price from Products table and then group by its category name.

```
select ProductCategory , MAX(PRICE) AS TotalPrice
FROM Products
Group by ProductCategory
```

118 %

Results | Messages

	ProductCategory	TotalPrice
1	Bakery	12.00
2	Bookstore	18.50
3	Clothing	29.99
4	Electronics	599.99
5	Footwear	60.00
6	Furniture	120.00
7	Grocery	3.99
8	Hardware	14.99
9	Jewelry	499.99
10	Pharmacy	9.99

Query for join three tables and then display some particular values –

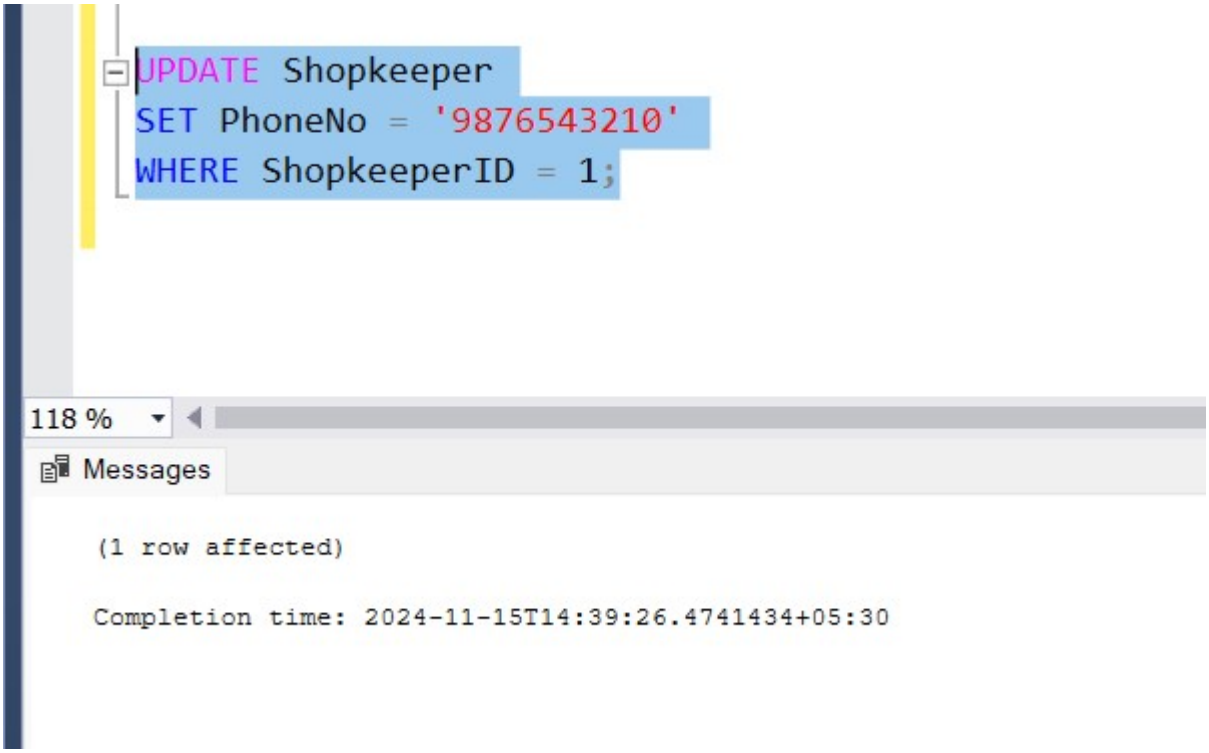
```
select Shopkeeper.FirstName as Info, Shopkeeper.Email, Products.ProductName, Sales.SalesDate
FROM Shopkeeper
INNER JOIN Products ON Shopkeeper.ShopkeeperID = Products.ShopkeeperID
INNER JOIN Sales ON Products.ProductID = Sales.ProductID
```

118 %

Results | Messages

	Info	Email	ProductName	SalesDate
1	John	john.doe@example.com	Organic Apple	2024-01-15
2	Jane	jane.smith@example.com	Men's Cotton T-shirt	2024-02-20
3	Michael	michael.johnson@example.com	Wireless Headphones	2024-03-10
4	Emily	emily.williams@example.com	Chocolate Cake	2024-04-05
5	David	david.brown@example.com	Wooden Coffee Table	2024-05-25
6	Sophia	sophia.jones@example.com	Vitamin D Supplements	2024-06-12
7	Daniel	daniel.garcia@example.com	Hammer	2024-07-18
8	Isabella	isabella.martinez@example.com	Fiction Novel	2024-08-22
9	James	james.lopez@example.com	Gold Necklace	2024-09-30
10	Olivia	olivia.gonzalez@example.com	Running Shoes	2024-10-15
11	John	john.doe@example.com	Almond Milk	2024-11-05
12	Jane	jane.smith@example.com	Women's Dress	2024-12-19
13	Michael	michael.johnson@example.com	Smartphone	2025-01-11
14	Emily	emily.williams@example.com	Gluten-Free Bread	2025-02-07
15	David	david.brown@example.com	Office Chair	2025-03-03

Query for update the value from an existing table



Function for Filtering Data-

--To find Maximum Quantity from the Products Table

```sql
--Function Based Query
CREATE FUNCTION dbo.record(@MaxQuantity DECIMAL(10,2))
RETURNS TABLE
AS
RETURN
(
    SELECT ProductID, ProductName, Quantity, Price
    FROM Products
    WHERE Quantity > @MaxQuantity
)

--RUN
SELECT * FROM dbo.record(50);
```

column Quantity(int, not null)

118 %

Results | Messages

	ProductID	ProductName	Quantity	Price
1	2	Men's Cotton T-shirt	100	15.99
2	6	Vitamin D Supplements	200	9.99
3	8	Fiction Novel	70	18.50
4	11	Almond Milk	100	3.99

--To add the first and last name from the shopkeeper table

```
--To Add the firstname & lastname from the Shopkeeper table
CREATE FUNCTION dbo.value(@Value CHAR)
RETURNS TABLE
AS
RETURN
(
    SELECT CONCAT(FirstName, ' ', LastName) AS Full_name
    FROM Shopkeeper
);
--Run
SELECT * FROM dbo.Value(5);
```

118 %

Results | Messages

	Full_name
1	John Doe
2	Jane Smith
3	Michael Johnson
4	Emily Williams
5	David Brown
6	Sophia Jones
7	Daniel Garcia
8	Isabella Martinez
9	James Lopez
10	Olivia Gonzalez
11	Matthew Lee
12	Charlotte Clark
13	Lucas Kim
14	Amelia Davis
15	Ethan Morris

--Add various tables and then show the particular values

```
CREATE FUNCTION dbo.combine(@CombineTable NVARCHAR)
RETURNS TABLE
AS
RETURN
(
    SELECT DISTINCT P.ProductID, P.ProductName ,S.FirstName, S.LastName
    FROM Shopkeeper S
    JOIN Products P ON S.ShopkeeperID = P.ShopkeeperID
    JOIN Sales Sa ON P.ProductID = Sa.ProductID
    WHERE SalesMonth BETWEEN 1 AND 5
);
--RUN
SELECT * FROM dbo.combine(10);
```

118 %

Results | Messages

	ProductID	ProductName	FirstName	LastName
1	1	Organic Apple	John	Doe
2	2	Men's Cotton T-shirt	Jane	Smith
3	3	Wireless Headphones	Michael	Johnson
4	4	Chocolate Cake	Emily	Williams
5	5	Wooden Coffee Table	David	Brown
6	13	Smartphone	Michael	Johnson
7	14	Gluten-Free Bread	Emily	Williams
8	15	Office Chair	David	Brown

--Stored Procedure—

1. Non paremeterised Stored Procedure

```sql
--Basic Stored Procedure:
--One non paremeterised SP
CREATE PROCEDURE GetAllData
AS
BEGIN
    SELECT * FROM Shopkeeper;
    SELECT * FROM Products;
    SELECT * FROM Sales;
    SELECT * FROM Customer;
END;
--RUN
EXEC GetAllData
```

118 %

Results Messages

	ShopkeeperID	FirstName	LastName	Email	Address	ShopType	PhoneNo	ShopRegistrationNo
1	1	John	Doe	john.doe@example.com	123 Elm St, Springfield	Grocery	9876543210	REG123456
2	2	Jane	Smith	jane.smith@example.com	456 Maple Ave, Riverdale	Clothing	0987654321	REG987654
3	3	Michael	Johnson	michael.johnson@example.com	789 Oak Dr, Gotham	Electronics	2345678901	REG234567
4	4	Emily	Williams	emily.williams@example.com	135 Pine St, Metropolis	Bakery	3456789012	REG345678
5	5	David	Brown	david.brown@example.com	246 Birch Rd, Star City	Furniture	4567890123	REG456789

	ProductID	ShopkeeperID	ProductCategory	ProductName	Quantity	CustomerID	Price	Discount	UnitInStock
1	1	1	Grocery	Organic Apple	50	1	2.50	0.00	500
2	2	2	Clothing	Men's Cotton T-shirt	100	1	15.99	5.00	150
3	3	3	Electronics	Wireless Headphones	30	2	45.99	10.00	70
4	4	4	Bakery	Chocolate Cake	10	3	12.00	0.50	40
5	5	5	Furniture	Wooden Coffee Table	5	5	120.00	15.00	10

	SalesID	ProductID	ShopkeeperID	SalesMonth	SalesYear	SalesDate
1	1	1	1	1	2024	2024-01-15
2	2	2	2	2	2024	2024-02-20
3	3	3	3	3	2024	2024-03-10
4	4	4	4	4	2024	2024-04-05

	CustomerID	CustomerName	PhoneNo	Quantity	City	EmailID
1	1	Alice Johnson	9876543210	3	Springfield	alice.johnson@example.com
2	2	Bob Smith	8765432109	5	Riverdale	bob.smith@example.com
3	3	Charlie Brown	7654321098	2	Gotham	charlie.brown@example.com

2. Parameterised Function

```sql
--Paremeterised SP
CREATE PROCEDURE GetProductsByCategory
    @Category NVARCHAR(50)
AS
BEGIN
    SELECT ProductID, ProductName, Price, Quantity, UnitInStock
    FROM Products
    WHERE ProductCategory = @Category;
END;
--Run
EXEC GetProductsByCategory @Category = 'Grocery';
```

118 %

Results Messages

	ProductID	ProductName	Price	Quantity	UnitInStock
1	1	Organic Apple	2.50	50	500
2	11	Almond Milk	3.99	100	250

3 .Using Try & Catch

Results Messages

	ProductID	ShopkeeperID	ProductCategory	ProductName	Quantity	CustomerID	Price	Discount	UnitInStock
1	1	1	Grocery	Organic Apple	50	1	2.50	0.00	500
2	2	2	Clothing	Men's Cotton T-shirt	100	1	15.99	5.00	150
3	3	3	Electronics	Wireless Headphones	30	2	45.99	10.00	70
4	4	4	Bakery	Chocolate Cake	10	3	12.00	0.50	40
5	5	5	Furniture	Wooden Coffee Table	5	5	120.00	15.00	10
6	6	6	Pharmacy	Vitamin D Supplements	200	8	9.99	1.00	300
7	7	7	Hardware	Hammer	50	6	14.99	2.00	100
8	8	8	Bookstore	Fiction Novel	70	7	18.50	0.00	120
9	9	9	Jewelry	Gold Necklace	5	9	499.99	25.00	7
10	10	10	Footwear	Running Shoes	40	4	60.00	10.00	90
11	11	1	Grocery	Almond Milk	100	3	3.99	0.00	250
12	12	2	Clothing	Women's Dress	50	5	29.99	7.00	80
13	13	3	Electronics	Smartphone	20	7	599.99	50.00	30
14	14	4	Bakery	Gluten-Free Bread	25	8	4.50	0.20	60
15	15	5	Furniture	Office Chair	15	10	75.00	5.00	25
16	16	1	Electronics	Bluetooth Speaker	20	NULL	150.00	NULL	50

```sql
--Try & Catch Stored Procedure
CREATE PROCEDURE UpdateDetails
@ShopkeeperID INT,
@ProductName CHAR(50),
@ProductCategory NVARCHAR(50),
@Quantity INT,
@Price INT,
@UnitInStock INT
AS
BEGIN
BEGIN TRY
INSERT INTO Products (ShopkeeperID, ProductName, ProductCategory, Quantity, Price, UnitInStock)
        VALUES (@ShopkeeperID, @ProductName, @ProductCategory, @Quantity, @Price, @UnitInStock);
PRINT 'Order updated successfully';
END TRY
BEGIN CATCH
PRINT 'An error occurred while updating the salary';
END CATCH
END
--RUN
EXEC UpdateDetails
    @ShopkeeperID = 1,
    @ProductCategory = 'Electronics',
    @ProductName = 'Bluetooth Speaker',
    @Quantity = 20,
    @Price = 150,
    @UnitInStock = 50;

SELECT * FROM Products
```

CURSOR Based Query

QUERY 01-

```
--CURSOR Based Query.
DECLARE @ShopkeeperID INT, @FirstName NVARCHAR(50), @LastName NVARCHAR(50), @Address NVARCHAR(50);

DECLARE RecordCursor CURSOR FOR
SELECT ShopkeeperID, FirstName, LastName, Address
FROM Shopkeeper

OPEN RecordCursor
FETCH NEXT FROM RecordCursor INTO @ShopkeeperID, @FirstName, @LastName , @Address

WHILE @@FETCH_STATUS=0

BEGIN
    PRINT 'ShopkeeperID: ' +CAST(@ShopkeeperID AS NVARCHAR(10)) +
    ',Name: '+@FirstName+' '+@LastName + ', Address: '+@Address;

    FETCH NEXT FROM RecordCursor INTO @ShopkeeperID, @FirstName,
    @LastName , @Address;
END
CLOSE RecordCursor
DEALLOCATE RecordCursor
```

Messages

```
ShopkeeperID: 1,Name: John Doe, Address: 123 Elm St, Springfield
ShopkeeperID: 2,Name: Jane Smith, Address: 456 Maple Ave, Riverdale
ShopkeeperID: 3,Name: Michael Johnson, Address: 789 Oak Dr, Gotham
ShopkeeperID: 4,Name: Emily Williams, Address: 135 Pine St, Metropolis
ShopkeeperID: 5,Name: David Brown, Address: 246 Birch Rd, Star City
ShopkeeperID: 6,Name: Sophia Jones, Address: 357 Cedar Ln, Central City
ShopkeeperID: 7,Name: Daniel Garcia, Address: 468 Willow St, Coast City
ShopkeeperID: 8,Name: Isabella Martinez, Address: 579 Ash Blvd, Bludhaven
ShopkeeperID: 9,Name: James Lopez, Address: 680 Oak Ave, Smallville
ShopkeeperID: 10,Name: Olivia Gonzalez, Address: 791 Sycamore St, Keystone City
ShopkeeperID: 11,Name: Matthew Lee, Address: 902 Poplar Ln, National City
ShopkeeperID: 12,Name: Charlotte Clark, Address: 123 Chestnut St, Hub City
ShopkeeperID: 13,Name: Lucas Kim, Address: 456 Walnut St, Hill Valley
ShopkeeperID: 14,Name: Amelia Davis, Address: 789 Redwood Dr, Greenfield
ShopkeeperID: 15,Name: Ethan Morris, Address: 135 Cedar Dr, River City

Completion time: 2024-11-15T21:43:43.4517037+05:30
```

QUERY 02 –

```
DECLARE @ProductID INT, @ProductName NVARCHAR(50), @ProductCategory NVARCHAR(50),
@FirstName NVARCHAR(50), @LastName NVARCHAR(50), @SalesDate DATE;

DECLARE DetailCursor CURSOR FOR
SELECT DISTINCT P.ProductID, P.ProductName ,P.ProductCategory,S.FirstName, S.LastName, Sa.SalesDate
FROM Shopkeeper S
JOIN Products P ON S.ShopkeeperID = P.ShopkeeperID
JOIN Sales Sa ON P.ProductID = Sa.ProductID;

OPEN DetailCursor
FETCH NEXT FROM DetailCursor INTO @ProductID, @ProductName, @ProductCategory,@FirstName, @LastName , @SalesDate

WHILE @@FETCH_STATUS=0

BEGIN
     PRINT 'ProductID: ' + CAST(@ProductID AS NVARCHAR(10)) +
          ', Name: ' + @FirstName + ' ' + @LastName +
          ', ProductName: ' + @ProductName +
          ', Category: ' + @ProductCategory +
          ', SalesDate: ' + CAST(@SalesDate AS NVARCHAR(20));

    FETCH NEXT FROM DetailCursor INTO  @ProductID, @ProductName, @ProductCategory,@FirstName, @LastName , @SalesDate
END
CLOSE DetailCursor
DEALLOCATE DetailCursor
```

Messages

```
ProductID: 1, Name: John Doe, ProductName: Organic Apple, Category: Grocery, SalesDate: 2024-01-15
ProductID: 2, Name: Jane Smith, ProductName: Men's Cotton T-shirt, Category: Clothing, SalesDate: 2024-02-20
ProductID: 3, Name: Michael Johnson, ProductName: Wireless Headphones, Category: Electronics, SalesDate: 2024-03-10
ProductID: 4, Name: Emily Williams, ProductName: Chocolate Cake, Category: Bakery, SalesDate: 2024-04-05
ProductID: 5, Name: David Brown, ProductName: Wooden Coffee Table, Category: Furniture, SalesDate: 2024-05-25
ProductID: 6, Name: Sophia Jones, ProductName: Vitamin D Supplements, Category: Pharmacy, SalesDate: 2024-06-12
ProductID: 7, Name: Daniel Garcia, ProductName: Hammer, Category: Hardware, SalesDate: 2024-07-18
ProductID: 8, Name: Isabella Martinez, ProductName: Fiction Novel, Category: Bookstore, SalesDate: 2024-08-22
ProductID: 9, Name: James Lopez, ProductName: Gold Necklace, Category: Jewelry, SalesDate: 2024-09-30
ProductID: 10, Name: Olivia Gonzalez, ProductName: Running Shoes, Category: Footwear, SalesDate: 2024-10-15
ProductID: 11, Name: John Doe, ProductName: Almond Milk, Category: Grocery, SalesDate: 2024-11-05
ProductID: 12, Name: Jane Smith, ProductName: Women's Dress, Category: Clothing, SalesDate: 2024-12-19
ProductID: 13, Name: Michael Johnson, ProductName: Smartphone, Category: Electronics, SalesDate: 2025-01-11
ProductID: 14, Name: Emily Williams, ProductName: Gluten-Free Bread, Category: Bakery, SalesDate: 2025-02-07
ProductID: 15, Name: David Brown, ProductName: Office Chair, Category: Furniture, SalesDate: 2025-03-03

Completion time: 2024-11-15T22:04:50.4948489+05:30
```

10.Conclusion :-

The **Digital Record Integration** Project successfully achieved its objectives by developing a centralized system for managing shopkeeper data, product inventory, and sales transactions. It created a robust database for tracking shopkeeper information, integrating real-time inventory and sales management, and ensuring data accuracy through validation rules and foreign key constraints. This system automates stock updates and sales tracking, enhancing operational efficiency. The user interface is intuitive, making it easy for shopkeepers to use, even with limited technical skills. The system also links customer data to sales, offering valuable insights for personalized services. Designed for scalability, it supports future growth, making it a reliable and efficient tool for shopkeepers, improving business operations and decision-making.