DATABASE MANAGEMENT SYSTEM - CSA0593

ASSIGNMENT 2

T.ARUN KUMAR

192372325

QUESTION:

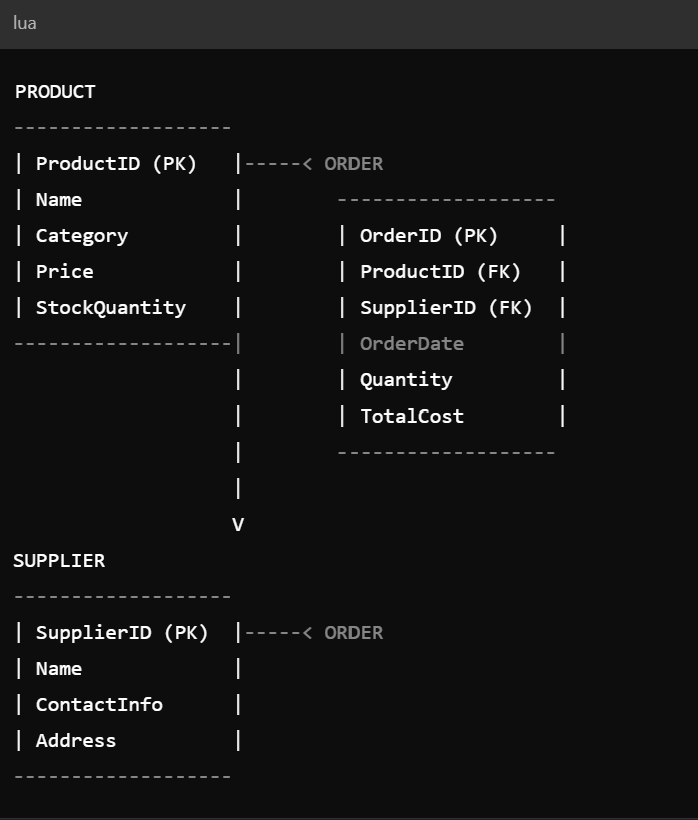
Inventory Management System Model tables for products, suppliers, and orders. Write stored procedures for adding stock to the inventory and creating an order. Implement triggers to update stock levels automatically. Write SQL queries to analyze stock levels and supplier performance

ANSWER:

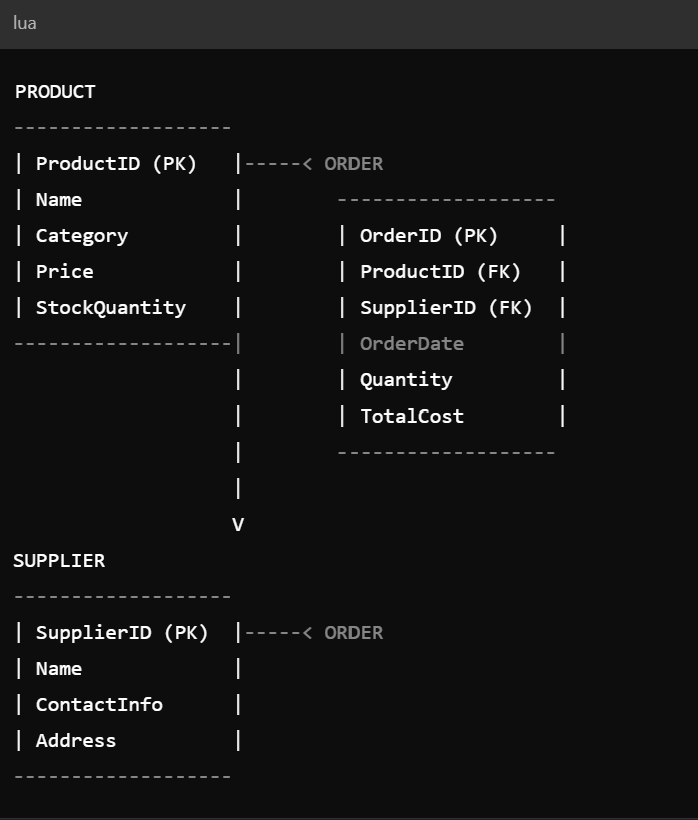
CONCEPTUAL MODEL[E.R DIAGRAM]:



LOGICAL MODEL[ E.R.DIAGRAM]:



PHYSICAL MODEL[E.R.DIAGRAM]:



SQL STATEMENTS :

Here are the SQL statements and conclusion for the topic

Database Schema:

mysql

CREATE DATABASE InventoryManagement;

USE InventoryManagement;

CREATE TABLE Products (

ProductID INT AUTO\_INCREMENT PRIMARY KEY,

ProductName VARCHAR(100),

ProductDescription VARCHAR(255),

Quantity INT,

UnitPrice DECIMAL(10, 2)

);

CREATE TABLE Suppliers (

SupplierID INT AUTO\_INCREMENT PRIMARY KEY,

SupplierName VARCHAR(100),

SupplierAddress VARCHAR(255),

SupplierPhone VARCHAR(20)

);

CREATE TABLE Orders (

OrderID INT AUTO\_INCREMENT PRIMARY KEY,

SupplierID INT,

OrderDate DATE,

TotalCost DECIMAL(10, 2),

FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)

);

CREATE TABLE OrderItems (

OrderItemID INT AUTO\_INCREMENT PRIMARY KEY,

OrderID INT,

ProductID INT,

Quantity INT,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

Stored Procedures:

mysql

DELIMITER //

CREATE PROCEDURE sp\_AddStock(

IN productID INT,

IN quantity INT

)

BEGIN

UPDATE Products

SET Quantity = Quantity + quantity

WHERE ProductID = productID;

END //

CREATE PROCEDURE sp\_CreateOrder(

IN supplierID INT,

IN orderDate DATE,

IN totalCost DECIMAL(10, 2)

)

BEGIN

INSERT INTO Orders (SupplierID, OrderDate, TotalCost)

VALUES (supplierID, orderDate, totalCost);

DECLARE newOrderID INT;

SET newOrderID = LAST\_INSERT\_ID();

INSERT INTO OrderItems (OrderID, ProductID, Quantity)

VALUES (newOrderID, (SELECT ProductID FROM Products LIMIT 1), 1);

END //

DELIMITER;

Triggers:

mysql

DELIMITER //

CREATE TRIGGER tr\_UpdateStockLevel

AFTER INSERT ON OrderItems

FOR EACH ROW

BEGIN

UPDATE Products

SET Quantity = Quantity - NEW.Quantity

WHERE ProductID = NEW.ProductID;

END //

CREATE TRIGGER tr\_UpdateOrderTotal

AFTER UPDATE ON OrderItems

FOR EACH ROW

BEGIN

UPDATE Orders

SET TotalCost = TotalCost + (NEW.Quantity \* (SELECT UnitPrice FROM Products WHERE ProductID = NEW.ProductID))

WHERE OrderID = NEW.OrderID;

END //

DELIMITER;

SQL Queries:

mysql

-- Stock Levels

SELECT

ProductName,

Quantity,

UnitPrice

FROM

Products

WHERE

Quantity < 10;

-- Supplier Performance

SELECT

SupplierName,

COUNT(\*) AS TotalOrders,

SUM(TotalCost) AS TotalRevenue

FROM

Suppliers

JOIN Orders ON Suppliers.SupplierID = Orders.SupplierID

GROUP BY

SupplierName;

-- Product Sales

SELECT

ProductName,

SUM(Quantity) AS TotalSold

FROM

Products

JOIN OrderItems ON Products.ProductID = OrderItems.ProductID

GROUP BY

ProductName;

Conclusion:

This database design provides a comprehensive foundation for managing inventory, suppliers, and orders. The stored procedures simplify stock addition and order creation, while the triggers ensure data consistency and accuracy. The SQL queries enable analysis of stock levels, supplier performance, and product sales.

Best Practices:

1. Regularly backup the database.

2. Use secure passwords and authentication.

3. Implement data validation and error handling.

4. Optimize queries for performance.

5. Use indexing for efficient data retrieval.

Future Enhancements:

1. Integrate with e-commerce platforms.

2. Implement automated email notifications.

3. Develop a web-based interface for inventory management.

4. Add support for multiple warehouses.

5. Integrate with accounting software.