DATABASE MANAGEMENT SYSTEM - CSA0593 ASSIGNMENT 5 B.LAKSHMI ANJALI 192311344

QUESTION:

Model tables for products, suppliers, sales, and customer orders.

- Write stored procedures to manage product stock, record sales transactions, and handle order fulfillment.
- Implement triggers to update inventory levels and reorder points when items are sold.
- Write SQL queries to generate sales reports, top-selling products, and low-stock alerts.

ANSWER:

CONCEPTUAL E.R.DIAGRAM:

```
PRODUCT
| ProductID (PK)
| Description
| Price
| StockQuantity
ReorderLevel
        Т
        ----- SUPPLIER
                         | SupplierID (PK) |
                         Name
                                          ı
                         | ContactInfo
                                          ı
CUSTOMER_ORDER
OrderID (PK)
CustomerName
                  Т
| OrderDate
| TotalAmount
        Т
            ----< ORDER_ITEM
                         | OrderItemID (PK)|
                         | OrderID (FK) |
                         | ProductID (FK) |
                        | Quantity
                         | Subtotal
                                          ı
        Т
SALES
| SalesID (PK)
| ProductID (FK)
| SaleDate
| QuantitySold
| TotalAmount
```

LOGICAL E.R DIAGRAM:

```
PRODUCT
| ProductID (PK) | ----< ORDER_ITEM
Name
Description
             | OrderItemID (PK)|
                      OrderID (FK)
Price
| StockQuantity |
                      | ProductID (FK) |
| ReorderLevel |
                      | Quantity
                      | Subtotal
               П
SUPPLIER
| SupplierID (PK) |----< PRODUCT_SUPPLIER
| ContactInfo
                      | ProductSupplierID (PK) |
Address
                     | ProductID (FK)
                      | SupplierID (FK)
CUSTOMER_ORDER
OrderID (PK)
CustomerName
OrderDate
TotalAmount
      П
       ٧
SALES
| SalesID (PK)
| ProductID (FK)
SaleDate
| QuantitySold
TotalAmount
```

PHYSICAL E.R.DIAGRAM:

```
PRODUCT
| ProductID (PK)
                     VARCHAR(100)
Name
| Description
| Price
                     DECIMAL(10,2)
| StockQuantity
ReorderLevel
                                 I
        ----- SUPPLIER
                        | SupplierID (PK)
                                                  ı
                                           VARCHAR (100)
                        Name
                                            VARCHAR(150)
                        | ContactInfo
                        Address
                                            TEXT |
CUSTOMER_ORDER
OrderID (PK)
CustomerName
                    VARCHAR(100)
OrderDate
TotalAmount
                    DECIMAL(10,2)
        Т
        ----- ORDER_ITEM
                        OrderItemID (PK)
                                                  ı
                        OrderID (FK)
                                                  ı
                        | ProductID (FK)
                                                  Т
                       | Quantity
                                                  П
                        Subtotal
                                            DECIMAL(10,2)
SALES
| SalesID (PK)
| ProductID (FK)
SaleDate
| QuantitySold
| TotalAmount
                    DECIMAL(10,2)
```

```
MYSQL STATEMENTS:
mysql
CREATE DATABASE SalesManagement;
USE SalesManagement;
CREATE TABLE Suppliers (
 SupplierID INT AUTO_INCREMENT PRIMARY KEY,
 SupplierName VARCHAR(100),
 SupplierAddress VARCHAR(255),
 SupplierPhone VARCHAR(20)
);
CREATE TABLE Products (
 ProductID INT AUTO_INCREMENT PRIMARY KEY,
 ProductName VARCHAR(100),
 ProductDescription VARCHAR(255),
 UnitPrice DECIMAL(10, 2),
```

```
StockLevel INT,
 ReorderPoint INT,
 SupplierID INT,
 FOREIGN KEY (SupplierID) REFERENCES
Suppliers(SupplierID)
);
CREATE TABLE Customers (
 CustomerID INT AUTO INCREMENT PRIMARY
KEY,
 CustomerName VARCHAR(100),
 CustomerAddress VARCHAR(255),
 CustomerPhone VARCHAR(20)
);
CREATE TABLE Orders (
 OrderID INT AUTO_INCREMENT PRIMARY KEY,
 CustomerID INT,
 OrderDate DATE,
```

```
TotalCost DECIMAL(10, 2),
 FOREIGN KEY (CustomerID) REFERENCES
Customers(CustomerID)
);
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)
);
CREATE TABLE Sales (
 SaleID INT AUTO_INCREMENT PRIMARY KEY,
 ProductID INT,
```

```
SaleDate DATE,
 Quantity INT,
 SalePrice DECIMAL(10, 2),
 FOREIGN KEY (ProductID) REFERENCES
Products(ProductID)
);
Stored Procedures:
mysql
DELIMITER //
CREATE PROCEDURE sp_RecordSale(
 IN productID INT,
 IN quantity INT,
IN salePrice DECIMAL(10, 2)
```

```
BEGIN
 INSERT INTO Sales (ProductID, SaleDate,
Quantity, SalePrice)
 VALUES (productID, CURDATE(), quantity,
salePrice);
 UPDATE Products
 SET StockLevel = StockLevel - quantity
 WHERE ProductID = productID;
END //
CREATE PROCEDURE sp_FulfillOrder(
IN orderID INT
BEGIN
 DECLARE finished INT DEFAULT 0;
 DECLARE productID INT;
 DECLARE quantity INT;
```

```
DECLARE curOrderItems CURSOR FOR SELECT ProductID, Quantity FROM OrderItems

WHERE OrderID = orderID;
```

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

OPEN curOrderItems;

```
read_loop: LOOP
  FETCH curOrderItems INTO productID, quantity;
IF finished = 1 THEN
    LEAVE read_loop;
END IF;
```

CALL sp_RecordSale(productID, quantity, (SELECT UnitPrice FROM Products WHERE ProductID = productID));

```
END LOOP;
 CLOSE curOrderItems;
END //
DELIMITER;
Triggers:
mysql
DELIMITER //
CREATE TRIGGER tr_UpdateReorderPoint
AFTER UPDATE ON Products
FOR EACH ROW
BEGIN
IF NEW.StockLevel <= NEW.ReorderPoint THEN
```

```
INSERT INTO LowStockAlerts (ProductID,
ProductName, StockLevel)
  VALUES (NEW.ProductID, NEW.ProductName,
NEW.StockLevel);
 END IF;
END //
DELIMITER;
SQL Queries:
mysql
-- Sales Report
SELECT
 ProductName,
 SUM(Quantity) AS TotalSales,
 SUM(SalePrice) AS TotalRevenue
FROM
 Products
```

```
JOIN Sales ON Products.ProductID =
Sales.ProductID
GROUP BY
 ProductName;
-- Top-Selling Products
SELECT
 ProductName,
 SUM(Quantity) AS TotalSales
FROM
 Products
JOIN Sales ON Products.ProductID =
Sales.ProductID
GROUP BY
 ProductName
ORDER BY
TotalSales DESC;
```

-- Low-Stock Alerts

SELECT

ProductName,

StockLevel

FROM

LowStockAlerts;

Conclusion:

This database design provides a comprehensive foundation for managing products, suppliers, sales, and customer orders. The stored procedures simplify sales transactions and order fulfillment, while the triggers ensure data consistency and accuracy. The SQL queries enable reporting on sales, top-selling products, and low-stock alerts.