DATABASE MANAGEMENT SYSTEM - CSA0593

ASSIGNMENT 2

T.ARUN KUMAR

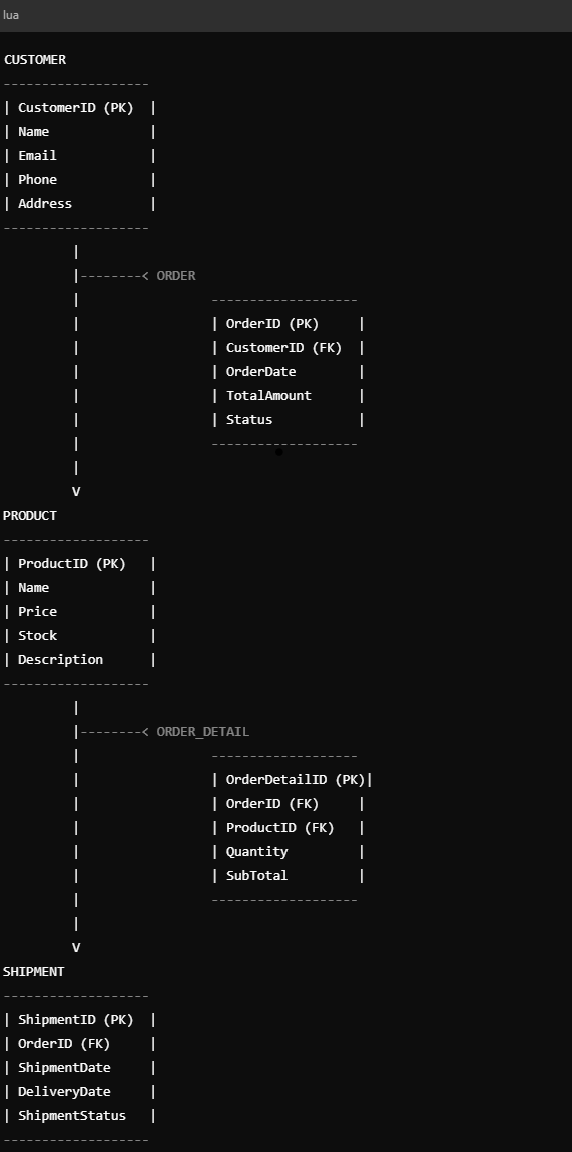
192372325

QUESTION:

Model tables for customers, products, orders, and shipments. Write stored procedures to place an order, process a shipment, and update the status of an order. Implement triggers to track order status changes and notify customers when a shipment is dispatched. Write SQL queries to analyze order volumes, customer satisfaction, and product popularity.

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:

SQL Statements:

Database Schema:

mysql

CREATE DATABASE EcommerceDB;

USE EcommerceDB;

CREATE TABLE Customers (

CustomerID INT AUTO\_INCREMENT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100),

Phone VARCHAR(20)

);

CREATE TABLE Products (

ProductID INT AUTO\_INCREMENT PRIMARY KEY,

ProductName VARCHAR(100),

ProductDescription VARCHAR(255),

Price DECIMAL(10, 2)

);

CREATE TABLE Orders (

OrderID INT AUTO\_INCREMENT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

Status VARCHAR(50),

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 Shipments (

ShipmentID INT AUTO\_INCREMENT PRIMARY KEY,

OrderID INT,

ShipmentDate DATE,

Status VARCHAR(50),

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)

);

Stored Procedures:

mysql

DELIMITER //

CREATE PROCEDURE sp\_PlaceOrder(

IN customerID INT,

IN orderDate DATE

)

BEGIN

INSERT INTO Orders (CustomerID, OrderDate, Status)

VALUES (customerID, orderDate, 'Pending');

END //

CREATE PROCEDURE sp\_ProcessShipment(

IN shipmentID INT,

IN orderID INT,

IN shipmentDate DATE

)

BEGIN

INSERT INTO Shipments (ShipmentID, OrderID, ShipmentDate, Status)

VALUES (shipmentID, orderID, shipmentDate, 'Dispatched');

UPDATE Orders

SET Status = 'Shipped'

WHERE OrderID = orderID;

END //

CREATE PROCEDURE sp\_UpdateOrderStatus(

IN orderID INT,

IN status VARCHAR(50)

)

BEGIN

UPDATE Orders

SET Status = status

WHERE OrderID = orderID;

END //

DELIMITER;

Triggers:

mysql

DELIMITER //

CREATE TRIGGER tr\_TrackOrderStatus

AFTER UPDATE ON Orders

FOR EACH ROW

BEGIN

INSERT INTO OrderStatusHistory (OrderID, Status, Timestamp)

VALUES (NEW.OrderID, NEW.Status, NOW());

END //

CREATE TRIGGER tr\_NotifyCustomer

AFTER INSERT ON Shipments

FOR EACH ROW

BEGIN

DECLARE customerEmail VARCHAR(100);

SELECT Email INTO customerEmail FROM Customers WHERE CustomerID = (SELECT CustomerID FROM Orders WHERE OrderID = NEW.OrderID);

-- Send email notification to customer

INSERT INTO EmailNotifications (CustomerEmail, Subject, Body)

VALUES (customerEmail, 'Shipment Dispatched', 'Your order has been dispatched.');

END //

DELIMITER;

SQL Queries:

mysql

-- Order Volumes

SELECT

MONTH(OrderDate) AS Month,

COUNT(\*) AS TotalOrders

FROM

Orders

GROUP BY

MONTH(OrderDate);

-- Customer Satisfaction

SELECT

CustomerID,

FirstName,

LastName,

COUNT(\*) AS TotalOrders,

SUM(CASE WHEN Status = 'Delivered' THEN 1 ELSE 0 END) AS SuccessfulDeliveries

FROM

Customers

JOIN Orders ON Customers.CustomerID = Orders.CustomerID

GROUP BY

CustomerID, FirstName, LastName;

-- Product Popularity

SELECT

ProductID,

ProductName,

SUM(Quantity) AS TotalSold

FROM

Products

JOIN OrderItems ON Products.ProductID = OrderItems.ProductID

GROUP BY

ProductID, ProductName;

Conclusion:

This database design provides a comprehensive foundation for managing customers, products, orders, and shipments. The stored procedures simplify order placement, shipment processing, and order status updates, while the triggers ensure data consistency and accuracy. The SQL queries enable analysis of order volumes, customer satisfaction, and product popularity.

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 order management.

4. Add support for multiple payment gateways.

5. Integrate with inventory management software.