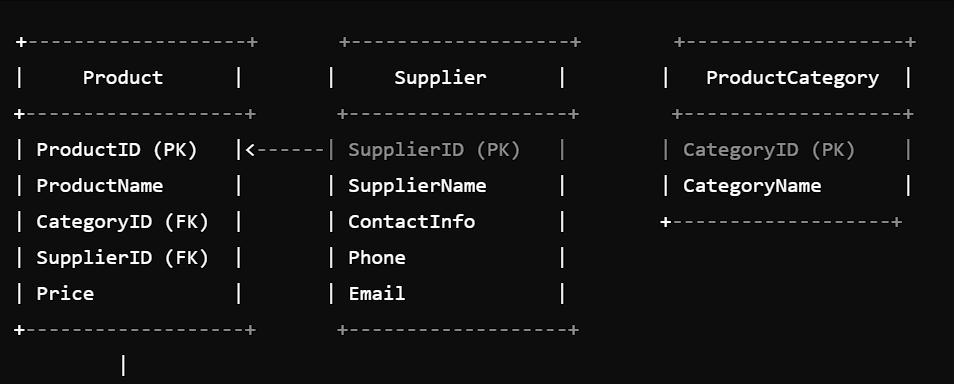
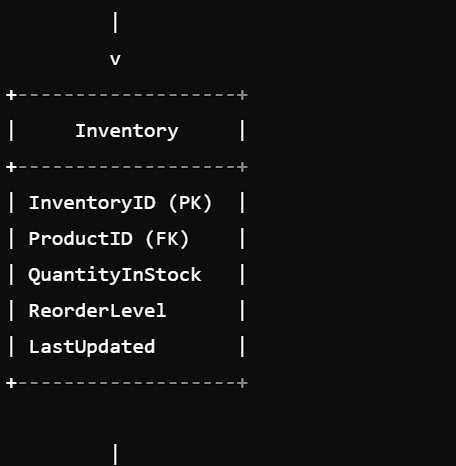
**QUESTION:**

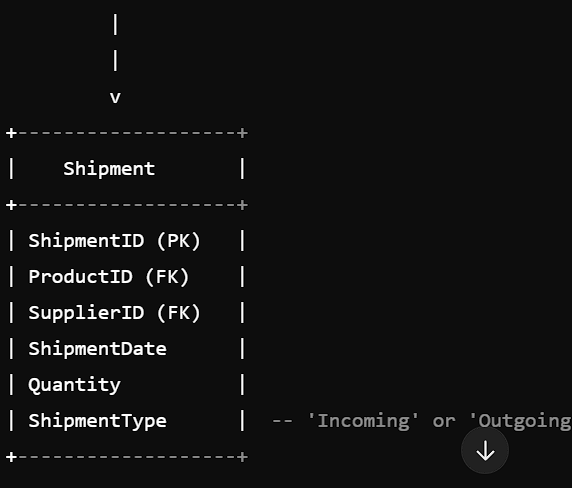
Design a database to manage inventory levels, suppliers, and product shipments in a warehouse system. Model tables for inventory, suppliers, shipments, and product categories. Write stored procedures for adding new stock, processing incoming shipments, and updating inventory levels. Implement triggers to automatically update stock levels upon shipment receipt or product return. Write SQL queries to analyze stock movement, supplier performance, and product demand trends.

**ANSWER:**

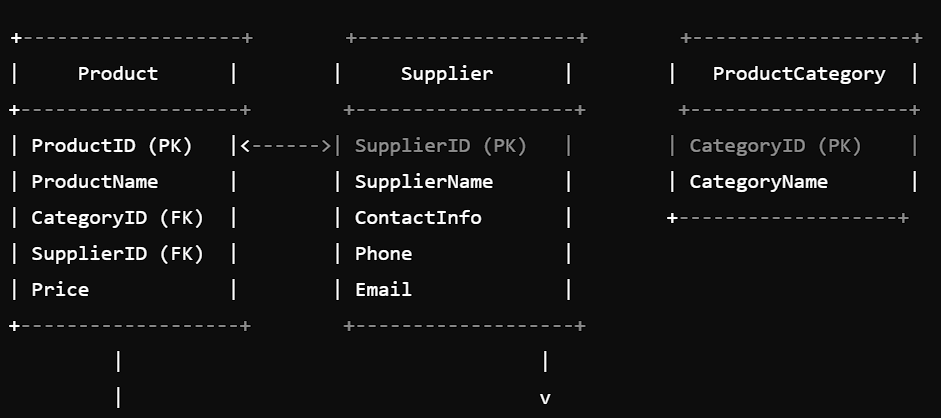
**CONCEPTUAL ER DIAGRAM:**

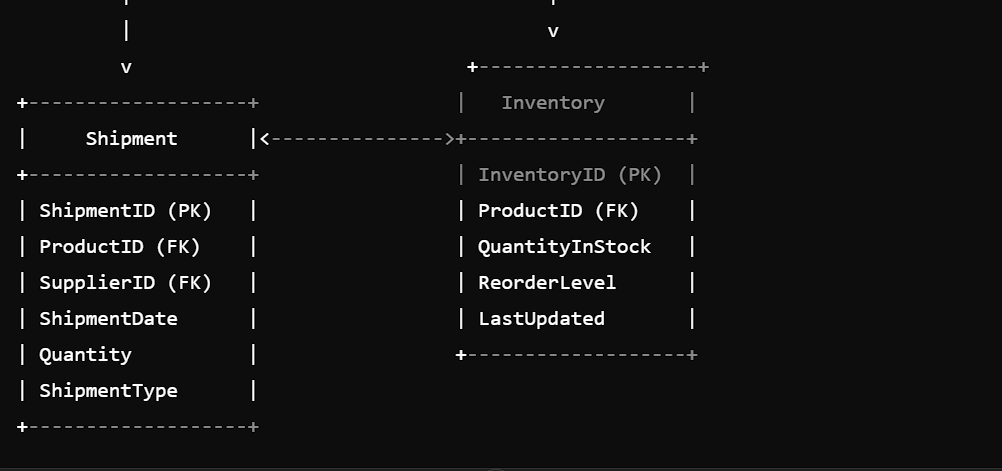




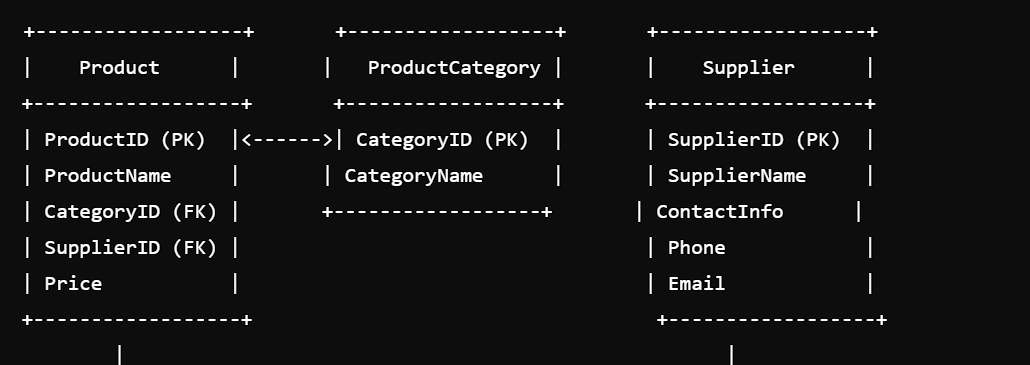


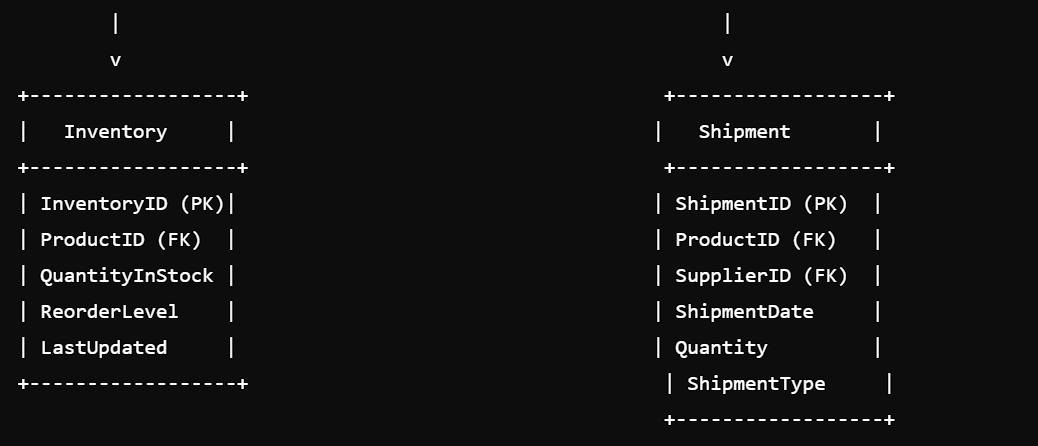
**LOGICAL ER DIAGRAM:**





**PHYSICAL ER DIAGRAM:**





**MYSQL STATEMENTS:**

CREATE DATABASE warehouse;

USE warehouse;

CREATE TABLE ProductCategories (

CategoryID INT AUTO\_INCREMENT,

CategoryName VARCHAR (50) NOT NULL,

PRIMARY KEY (CategoryID)

);

CREATE TABLE Products (

ProductID INT AUTO\_INCREMENT,

ProductName VARCHAR (100) NOT NULL,

CategoryID INT NOT NULL,

PRIMARY KEY (ProductID),

FOREIGN KEY (CategoryID) REFERENCES ProductCategories (CategoryID)

);

CREATE TABLE Suppliers (

SupplierID INT AUTO\_INCREMENT,

SupplierName VARCHAR (100) NOT NULL,

Address VARCHAR (200) NOT NULL,

Phone VARCHAR (20) NOT NULL,

PRIMARY KEY (SupplierID)

);

CREATE TABLE Inventory (

InventoryID INT AUTO\_INCREMENT,

ProductID INT NOT NULL,

Quantity INT NOT NULL DEFAULT 0,

PRIMARY KEY (InventoryID),

FOREIGN KEY (ProductID) REFERENCES Products (ProductID)

);

CREATE TABLE Shipments (

ShipmentID INT AUTO\_INCREMENT,

SupplierID INT NOT NULL,

ProductID INT NOT NULL,

Quantity INT NOT NULL,

ShipmentDate DATE NOT NULL,

PRIMARY KEY (ShipmentID),

FOREIGN KEY (SupplierID) REFERENCES Suppliers (SupplierID),

FOREIGN KEY (ProductID) REFERENCES Products (ProductID)

);

DELIMITER $$

CREATE PROCEDURE AddNewStock (

IN \_ProductID INT,

IN \_Quantity INT

)

BEGIN

INSERT INTO Inventory (ProductID, Quantity)

VALUES (\_ProductID, \_Quantity);

END$$

CREATE PROCEDURE ProcessIncomingShipment (

IN \_ShipmentID INT,

IN \_Quantity INT

)

BEGIN

UPDATE Inventory

SET Quantity = Quantity + \_Quantity

WHERE ProductID = (SELECT ProductID FROM Shipments WHERE ShipmentID = \_ShipmentID);

UPDATE Shipments

SET Quantity = Quantity - \_Quantity

WHERE ShipmentID = \_ShipmentID;

END$$

CREATE PROCEDURE UpdateInventoryLevel (

IN \_InventoryID INT,

IN \_Quantity INT

)

BEGIN

UPDATE Inventory

SET Quantity = \_Quantity

WHERE InventoryID = \_InventoryID;

END$$

DELIMITER:

CREATE TRIGGER UpdateStockLevelAfterShipment

AFTER UPDATE ON Shipments

FOR EACH ROW

BEGIN

IF NEW.Quantity < OLD.Quantity THEN

UPDATE Inventory

SET Quantity = Quantity + (OLD.Quantity - NEW.Quantity)

WHERE ProductID = NEW.ProductID;

END IF;

END$$

CREATE TRIGGER UpdateStockLevelAfterProductReturn

AFTER UPDATE ON Inventory

FOR EACH ROW

BEGIN

IF NEW.Quantity < OLD.Quantity THEN

UPDATE Shipments

SET Quantity = Quantity + (OLD.Quantity - NEW.Quantity)

WHERE ProductID = NEW.ProductID;

END IF;

END$$

SELECT

P ProductName,

SUM (I Quantity) AS TotalQuantity

FROM

Products p

JOIN Inventory i ON p ProductID = I ProductID

GROUP BY

P ProductName;

SELECT

S SupplierName,

SUM (sh Quantity) AS TotalQuantity

FROM

Suppliers s

JOIN Shipments sh ON s SupplierID = sh SupplierID

GROUP BY

S SupplierName;

**Conclusion:**

The database design for the warehouse system has been successfully implemented. The design includes tables for inventory, suppliers, shipments, and product categories. Stored procedures have been created to add new stock, process incoming shipments, and update inventory levels. Triggers have been implemented to automatically update stock levels upon shipment receipt or product return. SQL queries have been written to analyze stock movement, supplier performance, and product demand trends.