CREATE DATABASE IF NOT EXISTS inventory\_db;

USE inventory\_db;

-- Users Table

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) UNIQUE NOT NULL,

email VARCHAR(100) NOT NULL,

password VARCHAR(100) NOT NULL,

role ENUM('user', 'admin') NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Suppliers Table

CREATE TABLE suppliers (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

contact\_name VARCHAR(255),

contact\_email VARCHAR(255),

contact\_phone VARCHAR(20),

address TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Products Table

CREATE TABLE products (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

description TEXT,

price DECIMAL(10,2) NOT NULL,

stock\_quantity INT NOT NULL,

category VARCHAR(100),

image VARCHAR(255),

supplier\_id INT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (supplier\_id) REFERENCES suppliers(id) ON DELETE SET NULL

);

-- Product-Supplier Relationship

CREATE TABLE supplier\_products (

id INT AUTO\_INCREMENT PRIMARY KEY,

supplier\_id INT,

product\_id INT,

supply\_price DECIMAL(10,2) NOT NULL,

FOREIGN KEY (supplier\_id) REFERENCES suppliers(id) ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES products(id) ON DELETE CASCADE

);

-- Orders Table (Combined with Order Items)

CREATE TABLE orders (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

total\_price DECIMAL(10,2) NOT NULL,

status ENUM('pending', 'completed', 'canceled') NOT NULL DEFAULT 'pending',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE SET NULL

);

-- Order Items Table (Tracks product purchases per order)

CREATE TABLE order\_items (

id INT AUTO\_INCREMENT PRIMARY KEY,

order\_id INT,

product\_id INT,

quantity INT NOT NULL,

price DECIMAL(10,2) NOT NULL,

FOREIGN KEY (order\_id) REFERENCES orders(id) ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES products(id) ON DELETE CASCADE

);

-- Purchase Orders Table (For supplier purchases)

CREATE TABLE purchase\_orders (

id INT AUTO\_INCREMENT PRIMARY KEY,

supplier\_id INT,

total\_cost DECIMAL(10,2) NOT NULL,

status ENUM('ordered', 'received', 'canceled') NOT NULL DEFAULT 'ordered',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (supplier\_id) REFERENCES suppliers(id) ON DELETE CASCADE

);

-- Purchase Order Items Table

CREATE TABLE purchase\_order\_items (

id INT AUTO\_INCREMENT PRIMARY KEY,

purchase\_order\_id INT,

product\_id INT,

quantity INT NOT NULL,

cost DECIMAL(10,2) NOT NULL,

FOREIGN KEY (purchase\_order\_id) REFERENCES purchase\_orders(id) ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES products(id) ON DELETE CASCADE

);

-- Stock Movements (Tracks all stock updates)

CREATE TABLE stock\_movements (

id INT AUTO\_INCREMENT PRIMARY KEY,

product\_id INT,

change\_type ENUM('purchase', 'sale', 'restock', 'adjustment', 'supplier\_purchase') NOT NULL,

quantity INT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (product\_id) REFERENCES products(id) ON DELETE CASCADE

);

-- Inventory Logs (Tracks all stock updates)

CREATE TABLE inventory\_logs (

id INT AUTO\_INCREMENT PRIMARY KEY,

product\_id INT,

user\_id INT,

action ENUM('added', 'removed', 'updated') NOT NULL,

quantity INT NOT NULL,

note TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (product\_id) REFERENCES products(id) ON DELETE CASCADE,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE SET NULL

);

-- Product Performance Table (Tracks best & worst selling products)

CREATE TABLE product\_performance (

product\_id INT PRIMARY KEY,

total\_sales INT DEFAULT 0,

total\_revenue DECIMAL(10,2) DEFAULT 0.00,

total\_cost DECIMAL(10,2) DEFAULT 0.00,

last\_updated TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (product\_id) REFERENCES products(id) ON DELETE CASCADE

);

-- Trigger to Update Performance Table when an Order is Completed

DELIMITER //

CREATE TRIGGER after\_order\_complete

AFTER UPDATE ON orders

FOR EACH ROW

BEGIN

IF NEW.status = 'completed' THEN

UPDATE product\_performance pp

JOIN order\_items oi ON pp.product\_id = oi.product\_id

SET pp.total\_sales = pp.total\_sales + oi.quantity,

pp.total\_revenue = pp.total\_revenue + (oi.quantity \* oi.price),

pp.total\_cost = pp.total\_cost + (oi.quantity \* (SELECT cost FROM purchase\_order\_items WHERE product\_id = oi.product\_id ORDER BY created\_at DESC LIMIT 1))

WHERE oi.order\_id = NEW.id;

END IF;

END //

DELIMITER ;

-- Trigger to Update Stock Quantity on Stock Movement

DELIMITER //

CREATE TRIGGER after\_stock\_movement

AFTER INSERT ON stock\_movements

FOR EACH ROW

BEGIN

UPDATE products

SET stock\_quantity = stock\_quantity + NEW.quantity

WHERE id = NEW.product\_id;

END //

DELIMITER ;

SHOW TABLES;

SELECT \* FROM products;