```
/*
E-Commerce Sales & Supply Chain Performance Analytics System (MySQL
Project)

📌 Project Overview
This project simulates a real-world e-commerce + supply chain
operation, capturing how an online marketplace manages sales orders,
customers, products, shipping, inventory, vendors, and warehouse
operations.
The objective is to showcase your ability to:
Build a normalized retail & supply chain database
Represent the end-to-end customer order lifecycle
Analyze sales, customer behavior, product performance, and logistics
Deliver leadership-focused insights using SQL
This mirrors real company workflows at Amazon, Flipkart, Reliance
Retail, BigBasket, BlinkIt, etc.

📄 Dataset Summary
Simulated e-commerce system contains eight interconnected tables
forming a hybrid star schema.

Tables included:
Customers
Products
Orders
OrderItems
Inventory
Warehouses
Shipments
Vendors

Each table is linked via foreign keys, enabling deep analysis across:
revenue trends
product lifecycle
stock-outs
fulfilment delays
customer loyalty
top SKUs, top customers, top warehouses
vendor performance
logistics performance
*/




--- Setting Up the Database Schema
--- 1. Create Database
create database ecommerce;
use ecommerce;

--- 2. Customer Table
create table customers(
CustomerID int primary key,
CustomerName varchar(100),
Email varchar(100),
Phone varchar(15),
City varchar(50),
```

```
RegistrationDate Date );

INSERT INTO Customers VALUES
(1,'Amit Sharma','amit@gmail.com','9876543210','Delhi','2022-01-01'),
(2,'Riya Verma','riya@gmail.com','9988776655','Mumbai','2022-03-15'),
(3,'John Doe','john@gmail.com','7788994455','Bangalore','2022-04-10');



--- 3. Products Table
create table products(
ProductsID int primary key,
ProductName varchar(150),
Category varchar(50),
Brand varchar(50),
CostPrice decimal(10,2),
SellingPrice decimal(10,2) );

INSERT INTO Products VALUES
(101,'iPhone 14','Mobiles','Apple',60000,70000),
(102,'Samsung M32','Mobiles','Samsung',12000,15000),
(103,'Nike Shoes','Fashion','Nike',2000,3500);




--- 4. Warehouses Table
CREATE TABLE Warehouses (
WarehouseID INT PRIMARY KEY,
WarehouseName VARCHAR(100),
City VARCHAR(50),
Capacity INT
);

INSERT INTO Warehouses VALUES
(1,'Del NCR WH','Delhi',5000),
(2,'Mumbai West WH','Mumbai',4500);




--- 5. Inventory Table
CREATE TABLE Inventory (
InventoryID INT PRIMARY KEY,
ProductID INT,
WarehouseID INT,
Quantity INT,
LastUpdated DATE,
FOREIGN KEY (ProductID) REFERENCES Products(ProductsID),
FOREIGN KEY (WarehouseID) REFERENCES Warehouses(WarehouseID)
);

INSERT INTO Inventory VALUES
(1,101,1,120,'2024-01-10'),
(2,102,1,80,'2024-01-10'),
(3,103,2,200,'2024-01-09');
```

```
--- 6. Orders Table
CREATE TABLE Orders (
OrderID INT PRIMARY KEY,
CustomerID INT,
OrderDate DATE,
OrderStatus VARCHAR(20),
PaymentMode VARCHAR(20),
FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

INSERT INTO Orders VALUES
(1001,1,'2024-01-10','Delivered','UPI'),
(1002,2,'2024-01-11','Delivered','Card'),
(1003,1,'2024-01-12','Cancelled','COD');



--- 7. OrderItems Table
CREATE TABLE OrderItems (
OrderItemID INT PRIMARY KEY,
OrderID INT,
ProductID INT,
Quantity INT,
UnitPrice DECIMAL(10,2),
FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
FOREIGN KEY (ProductID) REFERENCES Products(ProductsID)
);

INSERT INTO OrderItems VALUES
(1,1001,101,1,70000),
(2,1002,102,2,15000),
(3,1003,103,1,3500);



--- 8. Shipments Table
CREATE TABLE Shipments (
ShipmentID INT PRIMARY KEY,
OrderID INT,
WarehouseID INT,
DispatchDate DATE,
DeliveryDate DATE,
DeliveryStatus VARCHAR(20),
FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
FOREIGN KEY (WarehouseID) REFERENCES Warehouses(WarehouseID)
);

INSERT INTO Shipments VALUES
(501,1001,1,'2024-01-11','2024-01-15','Delivered'),
(502,1002,2,'2024-01-12','2024-01-17','Delivered');
```

```
--- SQL Queries to Explore Insights
--- Now that the database is set up with records, beginning with
answering real-world questions through SQL queries.

--- 1. Daily Revenue Trend
SELECT
OrderDate,
SUM(oi.Quantity * oi.UnitPrice) AS DailyRevenue
FROM Orders o
JOIN OrderItems oi
ON o.OrderID = oi.OrderID
WHERE OrderStatus = 'Delivered'
GROUP BY OrderDate
ORDER BY OrderDate;

--- 2. Monthly GMV (Gross Merchandise Value)
SELECT
DATE_FORMAT(OrderDate, '%Y-%m') AS Month,
SUM(oi.Quantity * oi.UnitPrice) AS GMV
FROM Orders o
JOIN OrderItems oi
ON o.OrderID = oi.OrderID
WHERE OrderStatus = 'Delivered'
GROUP BY Month
ORDER BY Month;

--- 3. Top 10 Selling SKUs by Revenue
SELECT
p.ProductName,
SUM(oi.Quantity * oi.UnitPrice) AS Revenue
FROM OrderItems oi
JOIN Products p
ON oi.ProductID = p.ProductID
GROUP BY p.ProductName
ORDER BY Revenue DESC
LIMIT 10;

--- 4. Top 10 Selling SKUs by Units
SELECT p.ProductName,
SUM(oi.Quantity) AS UnitsSold
FROM OrderItems oi
JOIN Products p
ON oi.ProductID = p.ProductsID
GROUP BY p.ProductName
ORDER BY UnitsSold DESC
LIMIT 10;

--- 5. Category-Wise Revenue Contribution
SELECT p.Category,
SUM(oi.Quantity * oi.UnitPrice) AS Revenue
FROM OrderItems oi
JOIN Products p
ON oi.ProductID = p.ProductsID
GROUP BY p.Category
ORDER BY Revenue DESC;

--- 6. Average Order Value (AOV)
```

```
SELECT
SUM(oi.Quantity * oi.UnitPrice) / COUNT(DISTINCT o.OrderID) AS AOV
FROM Orders o
JOIN OrderItems oi
ON o.OrderID = oi.OrderID

--- 7. Repeat Purchase Rate
SELECT
(SELECT COUNT(*) FROM (
    SELECT CustomerID FROM Orders
    WHERE OrderStatus='Delivered'
    GROUP BY CustomerID HAVING COUNT(*) > 1
) t)
/
(SELECT COUNT(*) FROM Customers) * 100 AS RepeatPurchaseRate;

--- 8. Customer Lifetime Revenue
SELECT
c.CustomerID,
c.CustomerName,
SUM(oi.Quantity * oi.UnitPrice) AS LifetimeValue
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
JOIN OrderItems oi ON o.OrderID = oi.OrderID
WHERE o.OrderStatus='Delivered'
GROUP BY c.CustomerID, c.CustomerName
ORDER BY LifetimeValue DESC;

--- 9. High-Value Customers (LTV > ₹50,000)
SELECT
CustomerID,
CustomerName,
SUM(TotalValue) AS LTV
FROM (
    SELECT c.CustomerID, c.CustomerName,
    (oi.Quantity * oi.UnitPrice) AS TotalValue
    FROM Customers c
    JOIN Orders o ON c.CustomerID=o.CustomerID
    JOIN OrderItems oi ON o.OrderID=oi.OrderID
    WHERE o.OrderStatus='Delivered'
) x
GROUP BY CustomerID, CustomerName
HAVING LTV > 50000
ORDER BY LTV DESC;

--- 10. Order Status Mix (Delivered / Cancelled / Returned)
SELECT OrderStatus, COUNT(*) AS TotalOrders
FROM Orders
GROUP BY OrderStatus;

--- 11. Cancellation Rate
SELECT
(SUM(CASE WHEN OrderStatus='Cancelled' THEN 1 END) /
 SUM(1)) * 100 AS CancellationRate
FROM Orders;

--- 12. Warehouse Stock Availability
SELECT w.WarehouseName, p.ProductName, i.Quantity
```

```
FROM Inventory i
JOIN Warehouses w ON i.WarehouseID = w.WarehouseID
JOIN Products p ON i.ProductID = p.ProductsID
ORDER BY w.WarehouseName, p.ProductName;

--- 13. Stockout SKUs
SELECT p.ProductName, w.WarehouseName
FROM Inventory i
JOIN Products p ON i.ProductID = p.ProductsID
JOIN Warehouses w ON i.WarehouseID = w.WarehouseID
WHERE i.Quantity = 0;

--- 14. Potential Lost Revenue Due to Stockouts
SELECT p.ProductName,
       SUM(oi.Quantity * oi.UnitPrice) AS LostRevenue
FROM OrderItems oi
JOIN Inventory i ON oi.ProductID=i.ProductID
JOIN Products p ON p.ProductsID=i.ProductID
WHERE i.Quantity = 0
GROUP BY p.ProductName;

--- 15. Warehouse Fulfillment Lead Time
SELECT w.WarehouseName,
       AVG(DATEDIFF(s.DeliveryDate, s.DispatchDate)) AS AvgLeadTime
FROM Shipments s
JOIN Warehouses w ON s.WarehouseID=w.WarehouseID
WHERE DeliveryStatus='Delivered'
GROUP BY w.WarehouseName;

--- 16. Late Deliveries Count
SELECT COUNT(*) AS LateDeliveries
FROM Shipments
WHERE DeliveryStatus='Delivered'
AND DeliveryDate > DispatchDate + INTERVAL 5 DAY;

--- 17. On-Time Delivery Rate
SELECT
(SUM(CASE WHEN DeliveryDate <= DispatchDate + INTERVAL 5 DAY THEN 1
END) /
 COUNT(*)) * 100 AS OnTimeDeliveryRate
FROM Shipments;

--- 18. City-Wise Sales Performance
SELECT c.City,
       SUM(oi.Quantity * oi.UnitPrice) AS Revenue
FROM Customers c
JOIN Orders o ON c.CustomerID=o.CustomerID
JOIN OrderItems oi ON o.OrderID=oi.OrderID
WHERE o.OrderStatus='Delivered'
GROUP BY c.City
ORDER BY Revenue DESC;

--- 19. Fast-Moving Products (High Inventory Turnover)
SELECT p.ProductName,
       SUM(oi.Quantity) AS UnitsSold
FROM OrderItems oi
JOIN Products p ON p.ProductsID=oi.ProductID
GROUP BY p.ProductName
```

```
ORDER BY UnitsSold DESC;

--- 20. Slow-Moving Products (Low Sales)
SELECT p.ProductName,
       SUM(oi.Quantity) AS UnitsSold
FROM OrderItems oi
JOIN Products p ON p.ProductsID=oi.ProductID
GROUP BY p.ProductName
HAVING UnitsSold < 10
ORDER BY UnitsSold ASC;

--- 21. Revenue by Payment Mode
SELECT PaymentMode,
       SUM(oi.Quantity * oi.UnitPrice) AS Revenue
FROM Orders o
JOIN OrderItems oi ON o.OrderID=oi.OrderID
WHERE OrderStatus='Delivered'
GROUP BY PaymentMode;

--- 22. Weekday vs Weekend Revenue
SELECT
CASE
    WHEN DAYOFWEEK(OrderDate) IN (1,7) THEN 'Weekend'
    ELSE 'Weekday'
END AS DayType,
SUM(oi.Quantity * oi.UnitPrice) AS Revenue
FROM Orders o
JOIN OrderItems oi ON o.OrderID=oi.OrderID
WHERE OrderStatus='Delivered'
GROUP BY DayType;

--- 23. Basket Size (Average Items per Order)
SELECT AVG(ItemCount) AS AvgItemsPerOrder
FROM (
    SELECT OrderID, SUM(Quantity) AS ItemCount
    FROM OrderItems
    GROUP BY OrderID
) t;

--- 24. Overall Business Health Summary (GMV, Orders, Customers)
SELECT
COUNT(DISTINCT o.OrderID) AS TotalOrders,
COUNT(DISTINCT o.CustomerID) AS ActiveCustomers,
SUM(oi.Quantity * oi.UnitPrice) AS GMV
FROM Orders o
JOIN OrderItems oi ON o.OrderID=oi.OrderID
WHERE OrderStatus='Delivered';
```