**1. Table Creation: Defining Relationships & Constraints**

In the first step, I created five tables: **Customers, Suppliers, Products, Orders, and OrderDetails**.

* The **PRIMARY KEY** ensures that each record in a table has a **unique identifier** (e.g., CustomerID in Customers, ProductID in Products).
* The **FOREIGN KEY** constraints establish **relationships** between tables.
* Example: The OrderID in OrderDetails must match an OrderID in Orders, ensuring **referential integrity**.
* The **ON DELETE CASCADE** ensures that when a parent record (e.g., a Customer) is deleted, all related records (e.g., their Orders and OrderDetails) are automatically deleted to prevent orphan records.

**2.Data Insertion in the Tables**

After defining the table structures, I inserted **sample data** into each table using INSERT INTO.

* Customers: Contains information like **name, email, phone, and address**.
* Suppliers: Stores **supplier names and contact details**.
* Products: Includes **product name, description, price, stock quantity, and supplier reference**.
* Orders: Captures **order date, shipping address, payment method, and total amount**.
* OrderDetails: Links **orders with products** and includes **quantity and unit price**.

**3.Retrieving Orders for a Specific Customer**

📌 **Query**:

SELECT c.FirstName, c.LastName, p.ProductName, o.OrderDate, od.Quantity

FROM Orders o

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Products p ON od.ProductID = p.ProductID

JOIN Customers c ON o.CustomerID = c.CustomerID

WHERE c.CustomerID = 1;

**Explanation**

**Joins:** The database **links** multiple tables together based on matching keys:

* Orders & OrderDetails → **by OrderID**
* OrderDetails & Products → **by ProductID**
* Orders & Customers → **by CustomerID** **Filtering:** The WHERE clause ensures we only get orders **for CustomerID = 1**.  
   **Selection:** The query retrieves **Customer Name, Product Name, Order Date, and Quantity**.

**4️ Finding the Most Purchased Product**

📌 **Query**:

SELECT p.ProductName, SUM(od.Quantity) AS TotalQuantity

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY p.ProductName

ORDER BY TotalQuantity DESC

LIMIT 1;

**✅ Explanation**

**Join:** Connects OrderDetails with Products to fetch product names.  
**Grouping:** Groups results **by product name**.  
 **Aggregation:** Uses SUM(od.Quantity) to **calculate total units sold** per product.  
 **Ordering:** ORDER BY TotalQuantity DESC sorts the results **from highest to lowest**.  
 **Limiting:** LIMIT 1 ensures that **only the most purchased product is returned**.

**5.Updating Stock Quantity After an Order**

📌 **Query**:

UPDATE Products

SET StockQuantity = StockQuantity - (

SELECT SUM(Quantity) FROM OrderDetails

WHERE OrderDetails.ProductID = Products.ProductID

AND OrderID = 1

)

WHERE ProductID IN (

SELECT ProductID FROM OrderDetails WHERE OrderID = 1

);

**✅ Explanation**

**Identifies affected products:** Finds ProductID values for **OrderID 1**.  
 **Reduces stock quantity:** Subtracts SUM(Quantity) ordered from the StockQuantity.  
✔ **Ensures accuracy:** Only updates **products that were actually ordered** in OrderID 1.

**6️. Deleting a Customer’s Record (Ensuring Referential Integrity)**

📌 **Query with ON DELETE CASCADE (Simplified Delete)**:

DELETE FROM Customers WHERE CustomerID = 1;

**✅ Explanation**

* **Because ON DELETE CASCADE is used**, deleting a customer will:  
   **Automatically delete all related orders** from Orders.  
   **Automatically delete order details** from OrderDetails.  
   Prevent **foreign key constraint errors**.

**Purpose:**This approach ensures that **when a customer account is removed, all related data is cleaned up**, keeping the database consistent.