-- 1. View top 10 most expensive products

SELECT ProductID, ProductName, Price

FROM Products

ORDER BY Price DESC

LIMIT 10;

-- 2. Count total number of orders per customer

SELECT c.CustomerID, c.CustomerName, COUNT(o.OrderID) AS TotalOrders

FROM Customers c

INNER JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID, c.CustomerName;

-- 3. Total revenue generated per product

SELECT p.ProductID, p.ProductName, SUM(od.Quantity \* od.UnitPrice) AS TotalRevenue

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY p.ProductID, p.ProductName

ORDER BY TotalRevenue DESC;

-- 4. Subquery: Customers with above average number of orders

SELECT CustomerID, CustomerName

FROM (

SELECT c.CustomerID, c.CustomerName, COUNT(o.OrderID) AS OrderCount

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID, c.CustomerName

) AS CustomerOrders

WHERE OrderCount > (

SELECT AVG(OrderCount) FROM (

SELECT COUNT(OrderID) AS OrderCount

FROM Orders

GROUP BY CustomerID

) AS AvgOrders

);

-- 5. Create view: Product Revenue Summary

CREATE VIEW ProductRevenue AS

SELECT p.ProductID, p.ProductName, SUM(od.Quantity \* od.UnitPrice) AS Revenue

FROM Products p

JOIN OrderDetails od ON p.ProductID = od.ProductID

GROUP BY p.ProductID, p.ProductName;

-- 6. Use of index optimization (assumes execution in MySQL/PostgreSQL with proper permissions)

CREATE INDEX idx\_orders\_customerid ON Orders(CustomerID);

CREATE INDEX idx\_orderdetails\_productid ON OrderDetails(ProductID);