1. Inventory Overview

```
SELECT p.ProductName, pi.StockQuantity, pi.SafetyStockLevel, pi.ReorderPoint
FROM product p
JOIN productinventory pi ON p.ProductID = pi.ProductID
WHERE pi.StockQuantity < pi.ReorderPoint
ORDER BY pi.StockQuantity ASC;
```

2. Stock Value Calculation

```
SELECT p.ProductName, (pi.Quantity * p.StandardCost) AS TotalStockValue
FROM product p
JOIN productinventory pi ON p.ProductID = pi.ProductID
ORDER BY TotalStockValue DESC;
```

3. Product Sales Performance

```
SELECT p.ProductName, SUM(sod.Quantity) AS TotalQuantitySold, so.SalesOrderDate
FROM salesorderdetail sod
JOIN product p ON sod.ProductID = p.ProductID
JOIN salesorderheader so ON sod.SalesOrderID = so.SalesOrderID
WHERE so.SalesOrderDate >= DATEADD(MONTH, -1, GETDATE())
GROUP BY p.ProductName, so.SalesOrderDate
ORDER BY TotalQuantitySold DESC;
```

4. Supplier Performance

```
SELECT pv.VendorID, AVG(pv.LeadTime) AS AverageLeadTime,
STRING_AGG(p.ProductName, ', ') AS ProductNames
FROM productvendor pv
JOIN product p ON pv.ProductID = p.ProductID
GROUP BY pv.VendorID
ORDER BY AverageLeadTime ASC;
```

5. Stock Out Analysis

```
SELECT p.ProductName, pi.LastRestockDate, pi.Quantity
FROM productinventory pi
JOIN product p ON pi.ProductID = p.ProductID
WHERE pi.Quantity = 0 AND DATEDIFF(DAY, pi.LastRestockDate, GETDATE()) > 30;
```

6. Product Category Stock Levels

```
SELECT pc.CategoryName, SUM(pi.Quantity) AS TotalStockQuantity
FROM productcategory pc
JOIN product p ON pc.CategoryID = p.CategoryID
JOIN productinventory pi ON p.ProductID = pi.ProductID
GROUP BY pc.CategoryName;
```

7. Reorder Suggestion

```
SELECT p.ProductName, pi.Quantity, pi.ReorderPoint
FROM product p
JOIN productinventory pi ON p.ProductID = pi.ProductID
WHERE pi.Quantity < pi.ReorderPoint;
```

8. Top-Selling Products

```
SELECT p.ProductName, SUM(sod.Quantity) AS TotalQuantitySold
FROM salesorderdetail sod
JOIN product p ON sod.ProductID = p.ProductID
JOIN salesorderheader so ON sod.SalesOrderID = so.SalesOrderID
WHERE so.SalesOrderDate >= DATEADD(QUARTER, -1, GETDATE())
GROUP BY p.ProductName
ORDER BY TotalQuantitySold DESC
LIMIT 5;
```

9. Product Availability by Location

```
SELECT p.ProductName, l.LocationName, pi.Quantity AS AvailableQuantity
FROM productinventory pi
JOIN product p ON pi.ProductID = p.ProductID
JOIN location l ON pi.LocationID = l.LocationID;
```

10. Stock Optimization with Promotions

```
SELECT p.ProductName, so.SpecialOfferID, SUM(sod.Quantity) AS TotalQuantitySold
FROM salesorderdetail sod
JOIN product p ON sod.ProductID = p.ProductID
JOIN salesorderheader so ON sod.SalesOrderID = so.SalesOrderID
JOIN specialoffer so ON so.SpecialOfferID = so.SpecialOfferID
WHERE so.PromotionPeriod = 'Active'
```

```
GROUP BY p.ProductName, so.SpecialOfferID
ORDER BY TotalQuantitySold DESC;
```

11. Product Inventory Analysis

```
SELECT p.ProductName, l.LocationName, SUM(pi.Quantity) AS TotalStockQuantity
FROM productinventory pi
JOIN product p ON pi.ProductID = p.ProductID
JOIN location l ON pi.LocationID = l.LocationID
GROUP BY p.ProductName, l.LocationName;
```

12. Reorder Point Alert

```
SELECT p.ProductName, pi.Quantity, pi.ReorderPoint
FROM product p
JOIN productinventory pi ON p.ProductID = pi.ProductID
WHERE pi.Quantity < pi.ReorderPoint;
```

13. Top 5 Best-Selling Products by Quantity

```
SELECT p.ProductName, SUM(sod.OrderQty) AS TotalQuantitySold
FROM salesorderdetail sod
JOIN product p ON sod.ProductID = p.ProductID
GROUP BY p.ProductName
ORDER BY TotalQuantitySold DESC
LIMIT 5;
```

14. Product Stock Movement

```
SELECT p.ProductName,
    SUM(po.QuantityReceived) AS TotalReceived,
    SUM(sod.OrderQty) AS TotalSold
FROM purchaseorderde po
JOIN salesorderdetail sod ON po.ProductID = sod.ProductID
JOIN product p ON po.ProductID = p.ProductID
WHERE po.ReceiptDate >= DATEADD(MONTH, -1, GETDATE())
GROUP BY p.ProductName;
```

15. Average Lead Time per Vendor

```
SELECT pv.VendorID, AVG(DATEDIFF(DAY, po.PurchaseOrderDate, po.LastReceiptDate)) AS
AverageLeadTime
FROM productvendor pv
JOIN purchaseorderde po ON pv.ProductID = po.ProductID
GROUP BY pv.VendorID;
```

16. Stock Valuation

```
SELECT p.ProductName, (pi.Quantity * p.StandardCost) AS TotalStockValue
FROM product p
JOIN productinventory pi ON p.ProductID = pi.ProductID;
```

17. Identify Slow-Moving Products

```
SELECT p.ProductName, DATEDIFF(DAY, MAX(sod.OrderDate), GETDATE()) AS
DaysSinceLastSale
FROM salesorderdetail sod
JOIN product p ON sod.ProductID = p.ProductID
GROUP BY p.ProductName
HAVING DATEDIFF(DAY, MAX(sod.OrderDate), GETDATE()) > 90;
```

18. Optimal Order Quantity

```
SELECT p.ProductName,
    (AVG(sod.OrderQty) * 30) - pi.Quantity AS OptimalOrderQuantity
FROM salesorderdetail sod
JOIN product p ON sod.ProductID = p.ProductID
JOIN productinventory pi ON p.ProductID = pi.ProductID
WHERE sod.OrderDate >= DATEADD(DAY, -30, GETDATE())
GROUP BY p.ProductName, pi.Quantity;
```

19. Discontinued Product Report

```
SELECT p.ProductName, p.DiscontinuedDate, p.SafetyStockLevel
FROM product p
WHERE p.DiscontinuedDate IS NOT NULL;
```

20. Vendor Performance Analysis

```
SELECT pv.VendorID,
    AVG(po.QuantityReceived) AS AverageReceiptQuantity,
```

```
    AVG(po.UnitCost) AS AverageCostPerProduct
FROM productvendor pv
JOIN purchaseorderde po ON pv.ProductID = po.ProductID
GROUP BY pv.VendorID;
```

21. Product Stock Level Check

```
SELECT p.ProductName, pi.Quantity, p.SafetyStockLevel
FROM product p
JOIN productinventory pi ON p.ProductID = pi.ProductID
WHERE pi.Quantity < p.SafetyStockLevel;
```

22. Inventory Turnover Calculation

```
SELECT p.ProductName,
    (SUM(sod.OrderQty) / AVG(pi.Quantity)) AS InventoryTurnoverRatio
FROM salesorderdetail sod
JOIN product p ON sod.ProductID = p.ProductID
JOIN productinventory pi ON p.ProductID = pi.ProductID
GROUP BY p.ProductName;
```

Note: This query assumes that total sales (`SUM(sod.OrderQty)`) and average inventory (`AVG(pi.Quantity)`) are being calculated based on the sales order and inventory data available.

23. Stock Replenishment Analysis

```
SELECT p.ProductName, pi.Quantity, p.ReorderPoint
FROM product p
JOIN productinventory pi ON p.ProductID = pi.ProductID
WHERE pi.Quantity < p.ReorderPoint;
```

24. Product Vendor Performance

```
SELECT pv.VendorID,
    AVG(DATEDIFF(DAY, po.PurchaseOrderDate, po.LastReceiptDate)) AS AverageLeadTime
FROM productvendor pv
JOIN purchaseorderde po ON pv.ProductID = po.ProductID
GROUP BY pv.VendorID
```

```
ORDER BY AverageLeadTime DESC
LIMIT 1;
```

25. Sales and Inventory Correlation

```
SELECT p.ProductName,
    SUM(sod.OrderQty) AS TotalSales,
    pi.Quantity AS StockLevel
FROM salesorderdetail sod
JOIN product p ON sod.ProductID = p.ProductID
JOIN productinventory pi ON p.ProductID = pi.ProductID
GROUP BY p.ProductName, pi.Quantity;
```

26. Stockouts During Promotions

```
SELECT p.ProductName,
    po.PromotionID,
    pi.Quantity
FROM product p
JOIN productinventory pi ON p.ProductID = pi.ProductID
JOIN purchaseorderde po ON p.ProductID = po.ProductID
WHERE pi.Quantity = 0
AND po.PromotionStartDate <= GETDATE()
AND po.PromotionEndDate >= GETDATE();
```

27. Product Category Wise Sales Report

```
SELECT pc.CategoryName,
    SUM(sod.LineTotal) AS TotalSales,
    SUM(sod.OrderQty) AS UnitsSold
FROM salesorderdetail sod
JOIN product p ON sod.ProductID = p.ProductID
JOIN productcategory pc ON p.CategoryID = pc.CategoryID
GROUP BY pc.CategoryName;
```

28. Stock Movement Analysis

```
SELECT p.ProductName,
    pi.Quantity AS CurrentQuantity,
    (SELECT Quantity FROM productinventory WHERE ProductID = p.ProductID AND Year =
YEAR(GETDATE()) AND Month = 1) AS InitialQuantity,
    (pi.Quantity - (SELECT Quantity FROM productinventory WHERE ProductID = p.ProductID
AND Year = YEAR(GETDATE()) AND Month = 1)) AS StockMovement
```

```
FROM product p
JOIN productinventory pi ON p.ProductID = pi.ProductID;
```

29. Supplier Stock Availability

```
SELECT pv.VendorID,
    po.ProductID,
    po.QuantityReceived,
    pv.MinOrderQty
FROM productvendor pv
JOIN purchaseorderde po ON pv.ProductID = po.ProductID
WHERE po.QuantityReceived >= pv.MinOrderQty
GROUP BY pv.VendorID, po.ProductID, pv.MinOrderQty;
```

30. Out-of-Stock Products Trend

```
SELECT p.ProductName,
    COUNT(pi.Quantity = 0) AS OutOfStockCount,
    MONTH(po.PurchaseOrderDate) AS Month
FROM product p
JOIN productinventory pi ON p.ProductID = pi.ProductID
JOIN purchaseorderde po ON p.ProductID = po.ProductID
WHERE pi.Quantity = 0
GROUP BY p.ProductName, MONTH(po.PurchaseOrderDate)
ORDER BY Month;
```

31. How would you calculate the total stock value (Quantity * UnitPrice) for each product in the inventory?

```
SELECT p.ProductName,
    pi.Quantity * p.StandardCost AS TotalStockValue
FROM product p
JOIN productinventory pi ON p.ProductID = pi.ProductID;
```

32. Write a SQL query to find the top 5 products with the highest sales in the last quarter, considering sales from the `salesorderdetail` and `salesorderheader` tables.

```
SELECT p.ProductName,
```

```
    SUM(sod.OrderQty) AS TotalSales
FROM salesorderdetail sod
JOIN salesorderheader soh ON sod.SalesOrderID = soh.SalesOrderID
JOIN product p ON sod.ProductID = p.ProductID
WHERE soh.OrderDate >= DATEADD(QUARTER, -1, GETDATE())
GROUP BY p.ProductName
ORDER BY TotalSales DESC
LIMIT 5;
```

33. How would you identify the products that have been discontinued but still have stock available in the inventory?

```
SELECT p.ProductName, pi.Quantity
FROM product p
JOIN productinventory pi ON p.ProductID = pi.ProductID
WHERE p.Discontinued = 1 AND pi.Quantity > 0;
```

34. Write a SQL query to determine which product categories have the highest number of products that are low in stock (defined as stock < SafetyStockLevel).

```
SELECT pc.CategoryName,
    COUNT(p.ProductID) AS LowStockProductCount
FROM product p
JOIN productcategory pc ON p.CategoryID = pc.CategoryID
JOIN productinventory pi ON p.ProductID = pi.ProductID
WHERE pi.Quantity < p.SafetyStockLevel
GROUP BY pc.CategoryName
ORDER BY LowStockProductCount DESC;
```

35. How would you update the stock quantity for a product after receiving a shipment, using data from the `purchaseorderde` table?

```
UPDATE productinventory
SET Quantity = Quantity + (SELECT SUM(ReceivedQty) FROM purchaseorderde WHERE
ProductID = productinventory.ProductID)
WHERE ProductID = [YourProductID];
```

Note: Replace `[YourProductID]` with the actual product ID. You can adapt this query for bulk updates by joining tables or using dynamic parameters.

36. Write a query to find the products that are frequently ordered together based on `salesorderdetail` data.

```
SELECT sod1.ProductID AS Product1, sod2.ProductID AS Product2, COUNT(*) AS Frequency
FROM salesorderdetail sod1
JOIN salesorderdetail sod2 ON sod1.SalesOrderID = sod2.SalesOrderID
WHERE sod1.ProductID != sod2.ProductID
GROUP BY sod1.ProductID, sod2.ProductID
ORDER BY Frequency DESC
LIMIT 10;
```

37. How would you calculate the lead time for a product by comparing the `LastReceiptDate` from the `productvendor` table with the `PurchaseOrderDate` from the `purchaseorderhe` table?

```
SELECT p.ProductName,
    DATEDIFF(DAY, po.PurchaseOrderDate, pv.LastReceiptDate) AS LeadTime
FROM product p
JOIN productvendor pv ON p.ProductID = pv.ProductID
JOIN purchaseorderde po ON p.ProductID = po.ProductID
WHERE pv.LastReceiptDate IS NOT NULL AND po.PurchaseOrderDate IS NOT NULL;
```

38. Write a SQL query to find the products that have not been sold in the last six months from the `salesorderdetail` table.

```
SELECT p.ProductName
FROM product p
LEFT JOIN salesorderdetail sod ON p.ProductID = sod.ProductID
WHERE sod.OrderDate < DATEADD(MONTH, -6, GETDATE()) OR sod.OrderDate IS NULL;
```

39. How would you identify products with the highest return rates based on data in the `productinventory` and `salesorderdetail` tables?

```
SELECT p.ProductName,
    SUM(sod.ReturnQty) / SUM(sod.OrderQty) AS ReturnRate
FROM product p
JOIN salesorderdetail sod ON p.ProductID = sod.ProductID
JOIN productinventory pi ON p.ProductID = pi.ProductID
GROUP BY p.ProductName
HAVING SUM(sod.OrderQty) > 0
ORDER BY ReturnRate DESC;
```

40. Write a query to calculate the average order quantity for each product across all purchase orders, based on data from the `purchaseorderde` table.

```
SELECT p.ProductName,
    AVG(po.OrderQty) AS AvgOrderQuantity
FROM purchaseorderde po
JOIN product p ON po.ProductID = p.ProductID
GROUP BY p.ProductName;
```

41. Product Inventory Analysis

Question: Write an SQL query to find the total quantity of each product in inventory across all locations. Include the product name, location, and the total quantity of stock available.

```
SELECT p.ProductName, pi.Location, SUM(pi.Quantity) AS TotalStock
FROM product p
JOIN productinventory pi ON p.ProductID = pi.ProductID
GROUP BY p.ProductName, pi.Location;
```

42. Stock Availability

Question: Write an SQL query to list all products where the stock quantity is less than the reorder point. Include product name, quantity in stock, and reorder point.

```
SELECT p.ProductName, pi.Quantity, p.ReorderPoint
FROM product p
JOIN productinventory pi ON p.ProductID = pi.ProductID
WHERE pi.Quantity < p.ReorderPoint;
```

43. Stock Turnover Calculation

Question: How would you calculate the stock turnover rate for each product for the last month? Provide the necessary SQL query to calculate this, using data from `salesorderdetail` and `productinventory`.

```
SELECT p.ProductName,
    SUM(sod.OrderQty) / AVG(pi.Quantity) AS StockTurnover
FROM product p
JOIN salesorderdetail sod ON p.ProductID = sod.ProductID
```

```
JOIN productinventory pi ON p.ProductID = pi.ProductID
WHERE sod.OrderDate >= DATEADD(MONTH, -1, GETDATE())
GROUP BY p.ProductName;
```

44. Sales vs. Stock Analysis

Question: Write an SQL query to compare the total quantity sold (`salesorderdetail`) of each product with its current stock (`productinventory`). Include product name, total sales quantity, and current stock quantity.

```
SELECT p.ProductName,
    SUM(sod.OrderQty) AS TotalSales,
    pi.Quantity AS CurrentStock
FROM product p
JOIN salesorderdetail sod ON p.ProductID = sod.ProductID
JOIN productinventory pi ON p.ProductID = pi.ProductID
GROUP BY p.ProductName, pi.Quantity;
```

45. Inventory Valuation

Question: Write an SQL query to calculate the total value of the inventory for each product, using the current stock quantity and the standard cost of the product (`productcosthistor`).

```
SELECT p.ProductName,
    SUM(pi.Quantity * pc.StandardCost) AS TotalInventoryValue
FROM product p
JOIN productinventory pi ON p.ProductID = pi.ProductID
JOIN productcosthistor pc ON p.ProductID = pc.ProductID
GROUP BY p.ProductName;
```

46. Vendor Performance

Question: Write an SQL query to determine the average lead time for products from each vendor, based on data in `productvendor`. Include the vendor name and the average lead time.

```
SELECT pv.VendorName,
    AVG(DATEDIFF(DAY, po.PurchaseOrderDate, pv.LastReceiptDate)) AS AvgLeadTime
FROM productvendor pv
JOIN purchaseorderhe po ON pv.ProductID = po.ProductID
WHERE pv.LastReceiptDate IS NOT NULL AND po.PurchaseOrderDate IS NOT NULL
```

```
GROUP BY pv.VendorName;
```

47. Top Selling Products

Question: Write an SQL query to retrieve the top 10 best-selling products based on total sales quantity in the last 3 months. Include product name and total quantity sold.

```
SELECT p.ProductName,
    SUM(sod.OrderQty) AS TotalSales
FROM product p
JOIN salesorderdetail sod ON p.ProductID = sod.ProductID
WHERE sod.OrderDate >= DATEADD(MONTH, -3, GETDATE())
GROUP BY p.ProductName
ORDER BY TotalSales DESC
LIMIT 10;
```

48. Order History of a Product

Question: Write an SQL query to retrieve the order history (purchase and sales) for a specific product (given the product ID), including order date, quantity ordered, and order type (purchase or sales).

```
SELECT o.OrderDate,
    o.OrderQty,
    CASE
       WHEN po.PurchaseOrderID IS NOT NULL THEN 'Purchase'
       WHEN sod.SalesOrderID IS NOT NULL THEN 'Sales'
    END AS OrderType
FROM product p
LEFT JOIN purchaseorderde po ON p.ProductID = po.ProductID
LEFT JOIN salesorderdetail sod ON p.ProductID = sod.ProductID
WHERE p.ProductID = [YourProductID];
```

Note: Replace `[YourProductID]` with the actual product ID.

49. Stock and Sales Forecasting

Question: Write an SQL query that calculates the average monthly sales quantity of each product and compares it with the current stock to estimate how many months of inventory are available. Include product name, average monthly sales, and months of inventory available.

```
SELECT p.ProductName,
    AVG(sod.OrderQty) / DATEDIFF(MONTH, MIN(sod.OrderDate), GETDATE()) AS
AvgMonthlySales,
    pi.Quantity / (AVG(sod.OrderQty) / DATEDIFF(MONTH, MIN(sod.OrderDate), GETDATE()))
AS MonthsOfInventory
FROM product p
JOIN salesorderdetail sod ON p.ProductID = sod.ProductID
JOIN productinventory pi ON p.ProductID = pi.ProductID
GROUP BY p.ProductName, pi.Quantity;
```

50. Product Profitability Analysis

Question: Write an SQL query to calculate the profit for each product, using the total sales revenue from `salesorderdetail` and the cost from `productcosthistor`. Include product name, total revenue, total cost, and calculated profit.

```
SELECT p.ProductName,
    SUM(sod.OrderQty * sod.UnitPrice) AS TotalRevenue,
    SUM(sod.OrderQty * pc.StandardCost) AS TotalCost,
    SUM(sod.OrderQty * sod.UnitPrice) - SUM(sod.OrderQty * pc.StandardCost) AS Profit
FROM product p
JOIN salesorderdetail sod ON p.ProductID = sod.ProductID
JOIN productcosthistor pc ON p.ProductID = pc.ProductID
GROUP BY p.ProductName;
```