```sql
-- Drop existing procedures
IF OBJECT_ID('InsertOrderDetails', 'P') IS NOT NULL DROP PROCEDURE InsertOrderDetails;
GO

IF OBJECT_ID('UpdateOrderDetails', 'P') IS NOT NULL DROP PROCEDURE UpdateOrderDetails;
GO

IF OBJECT_ID('GetOrderDetails', 'P') IS NOT NULL DROP PROCEDURE GetOrderDetails;
GO

IF OBJECT_ID('DeleteOrderDetails', 'P') IS NOT NULL DROP PROCEDURE DeleteOrderDetails;
GO

-- Drop existing functions
IF OBJECT_ID('FormatDateMMDDYYYY', 'FN') IS NOT NULL DROP FUNCTION FormatDateMMDDYYYY;
GO

IF OBJECT_ID('FormatDateYYYYMMDD', 'FN') IS NOT NULL DROP FUNCTION FormatDateYYYYMMDD;
GO

-- Drop existing views
IF OBJECT_ID('vwCustomerOrders', 'V') IS NOT NULL DROP VIEW vwCustomerOrders;
GO

IF OBJECT_ID('vwCustomerOrdersYesterday', 'V') IS NOT NULL DROP VIEW
  vwCustomerOrdersYesterday;
GO

IF OBJECT_ID('MyProducts', 'V') IS NOT NULL DROP VIEW MyProducts;
GO

-- Drop existing triggers
IF OBJECT_ID('trgDeleteOrderDetails', 'TR') IS NOT NULL DROP TRIGGER
  trgDeleteOrderDetails;
GO

IF OBJECT_ID('trgCheckInventory', 'TR') IS NOT NULL DROP TRIGGER trgCheckInventory;
GO

-- Create InsertOrderDetails procedure
CREATE PROCEDURE InsertOrderDetails
    @OrderID INT,
    @ProductID INT,
    @UnitPrice MONEY = NULL,
    @Quantity INT,
    @Discount DECIMAL(5, 2) = 0
AS
BEGIN
    IF @UnitPrice IS NULL
    BEGIN
        SELECT @UnitPrice = ListPrice
        FROM Production.Product
```

```
        WHERE ProductID = @ProductID;
    END

    BEGIN TRY
        BEGIN TRANSACTION;

        INSERT INTO Sales.SalesOrderDetail (SalesOrderID, ProductID, UnitPrice,
          OrderQty, UnitPriceDiscount)
        VALUES (@OrderID, @ProductID, @UnitPrice, @Quantity, @Discount);

        UPDATE Production.ProductInventory
        SET Quantity = Quantity - @Quantity
        WHERE ProductID = @ProductID;

        IF EXISTS (SELECT 1 FROM Production.ProductInventory WHERE ProductID =
          @ProductID AND Quantity < 0)
        BEGIN
            RAISERROR('Not enough stock', 16, 1);
            ROLLBACK TRANSACTION;
            RETURN;
        END

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION;
        RAISERROR('Failed to place the order. Please try again.', 16, 1);
    END CATCH
END;
GO

-- Create UpdateOrderDetails procedure
CREATE PROCEDURE UpdateOrderDetails
    @OrderID INT,
    @ProductID INT,
    @UnitPrice MONEY = NULL,
    @Quantity INT = NULL,
    @Discount DECIMAL(5, 2) = NULL
AS
BEGIN
    -- Variable declarations to store original values
    DECLARE @OriginalUnitPrice MONEY;
    DECLARE @OriginalQuantity INT;
    DECLARE @OriginalDiscount DECIMAL(5, 2);

    -- Fetch original values if input values are NULL
    SELECT
        @OriginalUnitPrice = UnitPrice,
        @OriginalQuantity = OrderQty,
        @OriginalDiscount = UnitPriceDiscount
    FROM
        Sales.SalesOrderDetail
```

```
    WHERE
        SalesOrderID = @OrderID
        AND ProductID = @ProductID;

    -- If input parameters are NULL, use original values
    SET @UnitPrice = ISNULL(@UnitPrice, @OriginalUnitPrice);
    SET @Quantity = ISNULL(@Quantity, @OriginalQuantity);
    SET @Discount = ISNULL(@Discount, @OriginalDiscount);

    -- Update the order details
    UPDATE Sales.SalesOrderDetail
    SET
        UnitPrice = @UnitPrice,
        OrderQty = @Quantity,
        UnitPriceDiscount = @Discount
    WHERE
        SalesOrderID = @OrderID
        AND ProductID = @ProductID;

    -- Adjust inventory
    DECLARE @QuantityDifference INT = @Quantity - @OriginalQuantity;
    UPDATE Production.ProductInventory
    SET Quantity = Quantity - @QuantityDifference
    WHERE ProductID = @ProductID;
END;
GO

-- Create GetOrderDetails procedure
CREATE PROCEDURE GetOrderDetails
    @OrderID INT
AS
BEGIN
    IF NOT EXISTS (SELECT 1 FROM Sales.SalesOrderDetail WHERE SalesOrderID = @OrderID)
    BEGIN
        RAISERROR('The OrderID %d does not exist', 16, 1, @OrderID);
        RETURN;
    END

    SELECT *
    FROM Sales.SalesOrderDetail
    WHERE SalesOrderID = @OrderID;
END;
GO

-- Create DeleteOrderDetails procedure
CREATE PROCEDURE DeleteOrderDetails
    @OrderID INT,
    @ProductID INT
AS
BEGIN
    IF NOT EXISTS (SELECT 1 FROM Sales.SalesOrderDetail WHERE SalesOrderID = @OrderID ↵
      AND ProductID = @ProductID)
```

```
    BEGIN
        PRINT 'Invalid parameters';
        RETURN -1;
    END

    DELETE FROM Sales.SalesOrderDetail
    WHERE SalesOrderID = @OrderID AND ProductID = @ProductID;
END;
GO

-- Create FormatDateMMDDYYYY function
CREATE FUNCTION FormatDateMMDDYYYY (@date DATETIME)
RETURNS VARCHAR(10)
AS
BEGIN
    RETURN CONVERT(VARCHAR(10), @date, 101);
END;
GO

-- Create FormatDateYYYYMMDD function
CREATE FUNCTION FormatDateYYYYMMDD (@date DATETIME)
RETURNS VARCHAR(8)
AS
BEGIN
    RETURN CONVERT(VARCHAR(8), @date, 112);
END;
GO

-- Create vwCustomerOrders view
CREATE VIEW vwCustomerOrders
AS
SELECT
    Name AS CompanyName,
    SOH.SalesOrderID AS OrderID,
    SOH.OrderDate,
    SOD.ProductID,
    P.Name AS ProductName,
    SOD.OrderQty AS Quantity,
    SOD.UnitPrice,
    SOD.UnitPrice * SOD.OrderQty AS TotalPrice
FROM
    Sales.Customer AS C
JOIN
    Sales.SalesOrderHeader AS SOH ON C.CustomerID = SOH.CustomerID
JOIN
    Sales.SalesOrderDetail AS SOD ON SOH.SalesOrderID = SOD.SalesOrderID
JOIN
    Production.Product AS P ON SOD.ProductID = P.ProductID;
GO

-- Create vwCustomerOrdersYesterday view
CREATE VIEW vwCustomerOrdersYesterday
```

```sql
AS
SELECT *
FROM vwCustomerOrders
WHERE OrderDate = CONVERT(DATE, GETDATE() - 1);
GO

-- Create MyProducts view
CREATE VIEW MyProducts
AS
SELECT
    P.ProductID,
    P.Name AS ProductName,
    P.ListPrice AS UnitPrice,
    V.Name AS CompanyName,
    PC.Name AS CategoryName
FROM
    Production.Product P
JOIN
    Purchasing.ProductVendor PV ON P.ProductID = PV.ProductID
JOIN
    Purchasing.Vendor V ON PV.BusinessEntityID = V.BusinessEntityID
JOIN
    Production.ProductSubcategory PSC ON P.ProductSubcategoryID =
      PSC.ProductSubcategoryID
JOIN
    Production.ProductCategory PC ON PSC.ProductCategoryID = PC.ProductCategoryID
WHERE
    P.DiscontinuedDate IS NULL;
GO

-- Create trgDeleteOrderDetails trigger
CREATE TRIGGER trgDeleteOrderDetails
ON Sales.SalesOrderHeader
INSTEAD OF DELETE
AS
BEGIN
    DELETE FROM Sales.SalesOrderDetail
    WHERE SalesOrderID IN (SELECT SalesOrderID FROM deleted);

    DELETE FROM Sales.SalesOrderHeader
    WHERE SalesOrderID IN (SELECT SalesOrderID FROM deleted);
END;
GO
```