```sql
USE AdventureWork;
GO
-- 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

```

```sql
-- 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,
```

```
87      @Quantity INT = NULL,
88      @Discount DECIMAL(5, 2) = NULL
89  AS
90  BEGIN
91      -- Variable declarations to store original values
92      DECLARE @OriginalUnitPrice MONEY;
93      DECLARE @OriginalQuantity INT;
94      DECLARE @OriginalDiscount DECIMAL(5, 2);
95
96      -- Fetch original values if input values are NULL
97      SELECT
98          @OriginalUnitPrice = UnitPrice,
99          @OriginalQuantity = OrderQty,
100         @OriginalDiscount = UnitPriceDiscount
101     FROM
102         Sales.SalesOrderDetail
103     WHERE
104         SalesOrderID = @OrderID
105         AND ProductID = @ProductID;
106
107     -- If input parameters are NULL, use original values
108     SET @UnitPrice = ISNULL(@UnitPrice, @OriginalUnitPrice);
109     SET @Quantity = ISNULL(@Quantity, @OriginalQuantity);
110     SET @Discount = ISNULL(@Discount, @OriginalDiscount);
111
112     -- Update the order details
113     UPDATE Sales.SalesOrderDetail
114     SET
115         UnitPrice = @UnitPrice,
116         OrderQty = @Quantity,
117         UnitPriceDiscount = @Discount
118     WHERE
119         SalesOrderID = @OrderID
120         AND ProductID = @ProductID;
121
122     -- Adjust inventory
123     DECLARE @QuantityDifference INT = @Quantity - @OriginalQuantity;
124     UPDATE Production.ProductInventory
125     SET Quantity = Quantity - @QuantityDifference
126     WHERE ProductID = @ProductID;
127 END;
128 GO
129
130 -- Create GetOrderDetails procedure
131 CREATE PROCEDURE GetOrderDetails
132     @OrderID INT
133 AS
134 BEGIN
135     IF NOT EXISTS (SELECT 1 FROM Sales.SalesOrderDetail WHERE
```

```
          SalesOrderID = @OrderID)
136     BEGIN
137         RAISERROR('The OrderID %d does not exist', 16, 1, @OrderID);
138         RETURN;
139     END
140
141     SELECT *
142     FROM Sales.SalesOrderDetail
143     WHERE SalesOrderID = @OrderID;
144 END;
145 GO
146
147 -- Create DeleteOrderDetails procedure
148 CREATE PROCEDURE DeleteOrderDetails
149     @OrderID INT,
150     @ProductID INT
151 AS
152 BEGIN
153     IF NOT EXISTS (SELECT 1 FROM Sales.SalesOrderDetail WHERE
          SalesOrderID = @OrderID AND ProductID = @ProductID)
154     BEGIN
155         PRINT 'Invalid parameters';
156         RETURN -1;
157     END
158
159     DELETE FROM Sales.SalesOrderDetail
160     WHERE SalesOrderID = @OrderID AND ProductID = @ProductID;
161 END;
162 GO
163
164 -- Create FormatDateMMDDYYYY function
165 CREATE FUNCTION FormatDateMMDDYYYY (@date DATETIME)
166 RETURNS VARCHAR(10)
167 AS
168 BEGIN
169     RETURN CONVERT(VARCHAR(10), @date, 101);
170 END;
171 GO
172
173 -- Create FormatDateYYYYMMDD function
174 CREATE FUNCTION FormatDateYYYYMMDD (@date DATETIME)
175 RETURNS VARCHAR(8)
176 AS
177 BEGIN
178     RETURN CONVERT(VARCHAR(8), @date, 112);
179 END;
180 GO
181
182 -- Create vwCustomerOrders view
```

```sql
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
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
```