### **LEVEL-B TASK:**

### Stored Procedures

**1.InsertOrderDetails Procedure**

CREATE PROCEDURE InsertOrderDetails

@OrderID INT,

@ProductID INT,

@UnitPrice DECIMAL(10, 2) = NULL,

@Quantity INT,

@Discount DECIMAL(10, 2) = 0

AS

BEGIN

DECLARE @DefaultUnitPrice DECIMAL(10, 2)

IF @UnitPrice IS NULL

BEGIN

SELECT @DefaultUnitPrice = UnitPrice FROM Product WHERE ProductID = @ProductID

SET @UnitPrice = @DefaultUnitPrice

END

INSERT INTO OrderDetails (OrderID, ProductID, UnitPrice, Quantity, Discount)

VALUES (@OrderID, @ProductID, @UnitPrice, @Quantity, @Discount)

IF @@ROWCOUNT = 0

BEGIN

PRINT 'Failed to place the order. Please try again.'

RETURN

END

UPDATE Product

SET UnitsInStock = UnitsInStock - @Quantity

WHERE ProductID = @ProductID

IF @@ROWCOUNT = 0 OR EXISTS (SELECT 1 FROM Product WHERE ProductID = @ProductID AND UnitsInStock < 0)

BEGIN

ROLLBACK TRANSACTION

PRINT 'Not enough stock available. Order aborted.'

RETURN

END

IF EXISTS (SELECT 1 FROM Product WHERE ProductID = @ProductID AND UnitsInStock < ReorderLevel)

BEGIN

PRINT 'Warning: Stock level below reorder level.'

END

END

**2.UpdateOrderDetails Procedure**

CREATE PROCEDURE UpdateOrderDetails

@OrderID INT,

@ProductID INT,

@UnitPrice DECIMAL(10, 2) = NULL,

@Quantity INT = NULL,

@Discount DECIMAL(10, 2) = NULL

AS

BEGIN

DECLARE @CurrentUnitPrice DECIMAL(10, 2)

DECLARE @CurrentQuantity INT

DECLARE @CurrentDiscount DECIMAL(10, 2)

SELECT @CurrentUnitPrice = UnitPrice, @CurrentQuantity = Quantity, @CurrentDiscount = Discount

FROM OrderDetails

WHERE OrderID = @OrderID AND ProductID = @ProductID

UPDATE OrderDetails

SET

UnitPrice = ISNULL(@UnitPrice, @CurrentUnitPrice),

Quantity = ISNULL(@Quantity, @CurrentQuantity),

Discount = ISNULL(@Discount, @CurrentDiscount)

WHERE OrderID = @OrderID AND ProductID = @ProductID

UPDATE Product

SET UnitsInStock = UnitsInStock - ISNULL(@Quantity, @CurrentQuantity) + @CurrentQuantity

WHERE ProductID = @ProductID

END

**3.GetOrderDetails Procedure**

CREATE PROCEDURE GetOrderDetails

@OrderID INT

AS

BEGIN

IF NOT EXISTS (SELECT 1 FROM OrderDetails WHERE OrderID = @OrderID)

BEGIN

PRINT 'The OrderID ' + CAST(@OrderID AS NVARCHAR) + ' does not exist'

RETURN 1

END

SELECT \* FROM OrderDetails WHERE OrderID = @OrderID

EN**D**

**4.DeleteOrderDetails Procedure**

CREATE PROCEDURE DeleteOrderDetails

@OrderID INT,

@ProductID INT

AS

BEGIN

IF NOT EXISTS (SELECT 1 FROM OrderDetails WHERE OrderID = @OrderID AND ProductID = @ProductID)

BEGIN

PRINT 'Invalid parameters. OrderID or ProductID does not exist.'

RETURN -1

END

DELETE FROM OrderDetails

WHERE OrderID = @OrderID AND ProductID = @ProductID

END

### Functions

**1.Function to return date in MM/DD/YYYY format**

CREATE FUNCTION FormatDate\_MMDDYYYY (@InputDate DATETIME)

RETURNS NVARCHAR(10)

AS

BEGIN

RETURN CONVERT(NVARCHAR(10), @InputDate, 101)

END

**2.Function to return date in YYYYMMDD format**

CREATE FUNCTION FormatDate\_YYYYMMDD (@InputDate DATETIME)

RETURNS NVARCHAR(8)

AS

BEGIN

RETURN CONVERT(NVARCHAR(8), @InputDate, 112)

END

### Views

1. **vwCustomerOrders View**

**CREATE VIEW vwCustomerOrders AS**

**SELECT**

**c.CompanyName,**

**o.OrderID,**

**o.OrderDate,**

**od.ProductID,**

**p.ProductName,**

**od.Quantity,**

**od.UnitPrice,**

**od.Quantity \* od.UnitPrice AS TotalPrice**

**FROM**

**Orders o**

**JOIN**

**Customers c ON o.CustomerID = c.CustomerID**

**JOIN**

**OrderDetails od ON o.OrderID = od.OrderID**

**JOIN**

**Products p ON od.ProductID = p.ProductID**

**2.View for orders placed yesterday**

CREATE VIEW vwCustomerOrdersYesterday AS

SELECT

c.CompanyName,

o.OrderID,

o.OrderDate,

od.ProductID,

p.ProductName,

od.Quantity,

od.UnitPrice,

od.Quantity \* od.UnitPrice AS TotalPrice

FROM

Orders o

JOIN

Customers c ON o.CustomerID = c.CustomerID

JOIN

OrderDetails od ON o.OrderID = od.OrderID

JOIN

Products p ON od.ProductID = p.ProductID

WHERE

o.OrderDate = CAST(GETDATE() - 1 AS DATE)

**3.MyProducts View**

CREATE VIEW MyProducts AS

SELECT

p.ProductID,

p.ProductName,

p.QuantityPerUnit,

p.UnitPrice,

s.CompanyName,

c.CategoryName

FROM

Products p

JOIN

Suppliers s ON p.SupplierID = s.SupplierID

JOIN

Categories c ON p.CategoryID = c.CategoryID

WHERE

p.Discontinued = 0

### Triggers

**1.Instead of Delete Trigger on Orders Table**

CREATE TRIGGER trgInsteadOfDeleteOrder

ON Orders

INSTEAD OF DELETE

AS

BEGIN

DELETE FROM OrderDetails

WHERE OrderID IN (SELECT OrderID FROM deleted)

DELETE FROM Orders

WHERE OrderID IN (SELECT OrderID FROM deleted)

END

**2.Trigger to check stock before placing order**

CREATE TRIGGER trgBeforeInsertOrderDetails

ON OrderDetails

INSTEAD OF INSERT

AS

BEGIN

DECLARE @ProductID INT, @Quantity INT

SELECT @ProductID = ProductID, @Quantity = Quantity

FROM inserted

IF EXISTS (SELECT 1 FROM Products WHERE ProductID = @ProductID AND UnitsInStock >= @Quantity)

BEGIN

INSERT INTO OrderDetails (OrderID, ProductID, UnitPrice, Quantity, Discount)

SELECT OrderID, ProductID, UnitPrice, Quantity, Discount

FROM inserted

UPDATE Products

SET UnitsInStock = UnitsInStock - @Quantity

WHERE ProductID = @ProductID

END

ELSE

BEGIN

PRINT 'Order could not be placed due to insufficient stock.'

END

END