Assignment - Week II

Domain: SQL

Name: Harsh Bhasin

Student ID: CT_CSI_SQ_3533

Contact No :9098835618

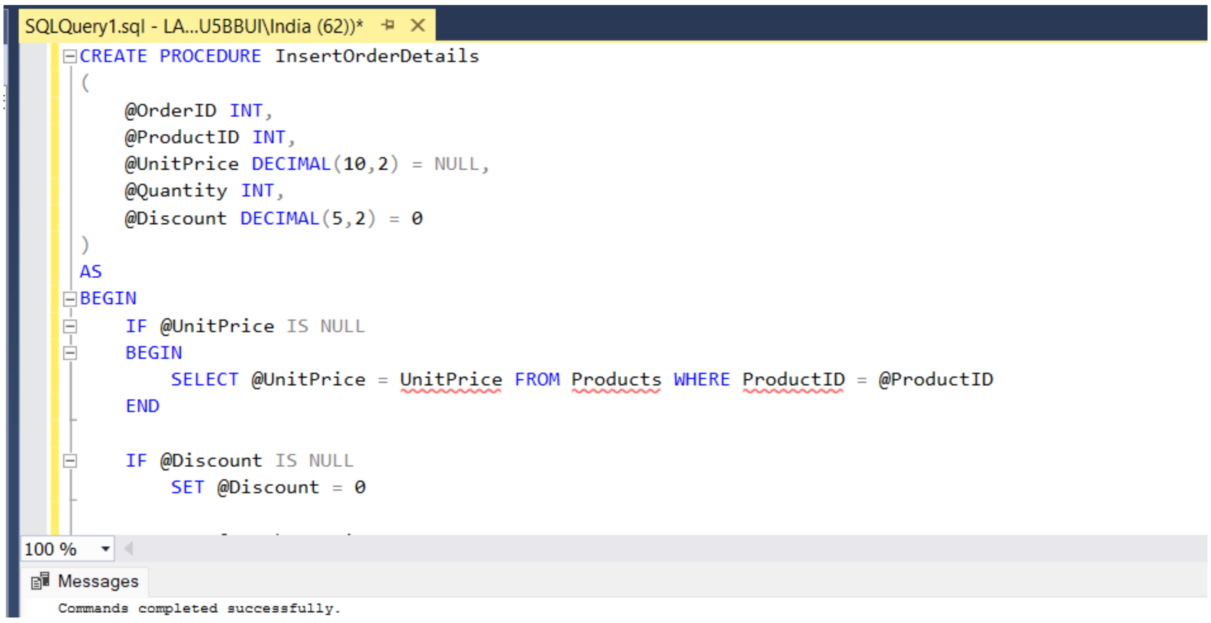
Email ID :Harshbha30@gmail.com

Part 1: Stored Procedures

1. InsertOrderDetails Procedure:

```
Query: CREATE PROCEDURE InsertOrderDetails
(
  @OrderID INT,
  @ProductID INT,
  @UnitPrice DECIMAL(10,2) = NULL,
  @Quantity INT,
  @Discount DECIMAL(5,2) = 0
)
AS
BEGIN
  IF @UnitPrice IS NULL
  BEGIN
    SELECT @UnitPrice = UnitPrice FROM Products WHERE ProductID = @ProductID
  END

  IF @Discount IS NULL
    SET @Discount = 0

  DECLARE @StockQuantity INT
  SELECT @StockQuantity = UnitsInStock FROM Products WHERE ProductID = @ProductID

  IF @StockQuantity >= @Quantity
  BEGIN
    INSERT INTO [Order Details] (OrderID, ProductID, UnitPrice, Quantity, Discount)
    VALUES (@OrderID, @ProductID, @UnitPrice, @Quantity, @Discount)

    UPDATE Products
    SET UnitsInStock = UnitsInStock - @Quantity
    WHERE ProductID = @ProductID

    DECLARE @ReorderLevel INT
    SELECT @ReorderLevel = ReorderLevel FROM Products WHERE ProductID = @ProductID

    IF (@StockQuantity - @Quantity) < @ReorderLevel
    BEGIN
      PRINT 'Warning: Product stock is below reorder level!'
    END

    PRINT 'Order placed successfully!'
  END
  ELSE
  BEGIN
    PRINT 'Failed to place order. Not enough stock available.'
  END
END
```

Result:



SQLQuery1.sql - LA...U5BBUI\India (62))*

```
CREATE PROCEDURE InsertOrderDetails
(
    @OrderID INT,
    @ProductID INT,
    @UnitPrice DECIMAL(10,2) = NULL,
    @Quantity INT,
    @Discount DECIMAL(5,2) = 0
)
AS
BEGIN
    IF @UnitPrice IS NULL
    BEGIN
        SELECT @UnitPrice = UnitPrice FROM Products WHERE ProductID = @ProductID
    END

    IF @Discount IS NULL
        SET @Discount = 0
```

100 %

Messages

```
Commands completed successfully.
```

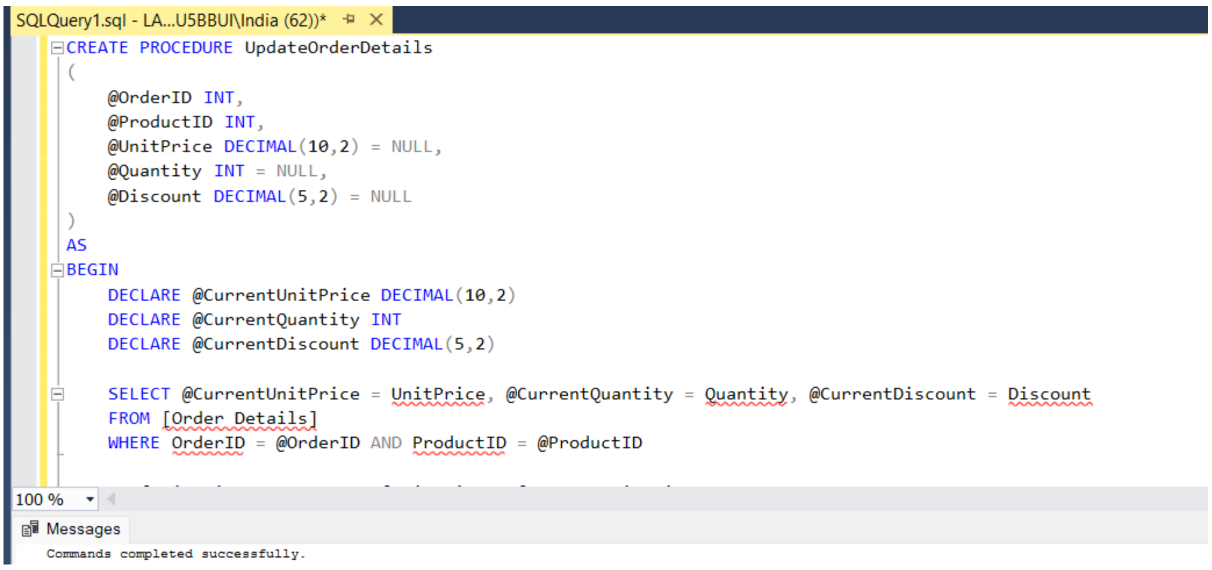
2. UpdateOrderDetails Procedure:

Query:

```
CREATE PROCEDURE UpdateOrderDetails
(
   @OrderID INT,
   @ProductID INT,
   @UnitPrice DECIMAL(10,2) = NULL,
   @Quantity INT = NULL,
   @Discount DECIMAL(5,2) = NULL
)
AS
BEGIN
   DECLARE @CurrentUnitPrice DECIMAL(10,2)
   DECLARE @CurrentQuantity INT
   DECLARE @CurrentDiscount DECIMAL(5,2)

   SELECT @CurrentUnitPrice = UnitPrice, @CurrentQuantity = Quantity,
@CurrentDiscount = Discount
   FROM [Order Details]
   WHERE OrderID = @OrderID AND ProductID = @ProductID

   IF @UnitPrice IS NULL SET @UnitPrice = @CurrentUnitPrice
   IF @Quantity IS NULL SET @Quantity = @CurrentQuantity
```

```
    IF @Discount IS NULL SET @Discount = @CurrentDiscount

    UPDATE [Order Details]
    SET UnitPrice = @UnitPrice, Quantity = @Quantity, Discount = @Discount
    WHERE OrderID = @OrderID AND ProductID = @ProductID

    UPDATE Products
    SET UnitsInStock = UnitsInStock + @CurrentQuantity - @Quantity
    WHERE ProductID = @ProductID
END
```

Result:



```
CREATE PROCEDURE UpdateOrderDetails
(
    @OrderID INT,
    @ProductID INT,
    @UnitPrice DECIMAL(10,2) = NULL,
    @Quantity INT = NULL,
    @Discount DECIMAL(5,2) = NULL
)
AS
BEGIN
    DECLARE @CurrentUnitPrice DECIMAL(10,2)
    DECLARE @CurrentQuantity INT
    DECLARE @CurrentDiscount DECIMAL(5,2)

    SELECT @CurrentUnitPrice = UnitPrice, @CurrentQuantity = Quantity, @CurrentDiscount = Discount
    FROM [Order Details]
    WHERE OrderID = @OrderID AND ProductID = @ProductID
```

```
Commands completed successfully.
```

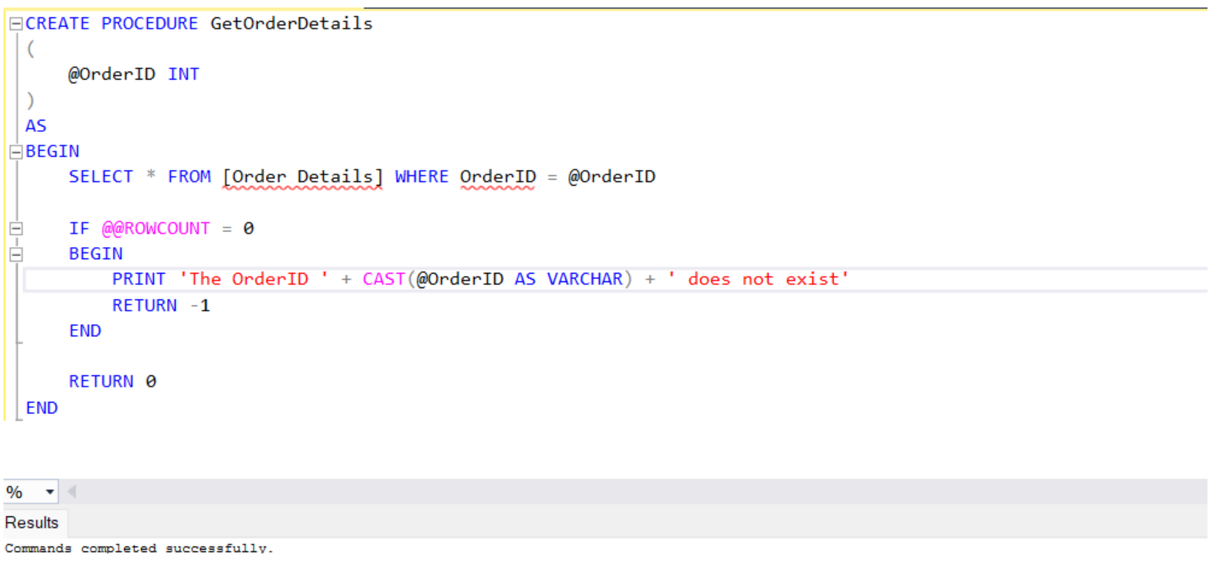
3. GetOrderDetails Procedure

Query:

```
CREATE PROCEDURE GetOrderDetails
(
   @OrderID INT
)
AS
BEGIN
   SELECT * FROM [Order Details] WHERE OrderID = @OrderID
```

```
    IF @@ROWCOUNT = 0
    BEGIN
        PRINT 'The OrderID ' + CAST(@OrderID AS VARCHAR) + ' does not exist'
        RETURN -1
    END

    RETURN 0
END
```

Result:



```
CREATE PROCEDURE GetOrderDetails
(
    @OrderID INT
)
AS
BEGIN
    SELECT * FROM [Order Details] WHERE OrderID = @OrderID

    IF @@ROWCOUNT = 0
    BEGIN
        PRINT 'The OrderID ' + CAST(@OrderID AS VARCHAR) + ' does not exist'
        RETURN -1
    END

    RETURN 0
END
```

```
Commands completed successfully.
```

4. DeleteOrderDetails Procedure:

Query:

```
CREATE PROCEDURE DeleteOrderDetails
(
    @OrderID INT,
    @ProductID INT
)
AS
```

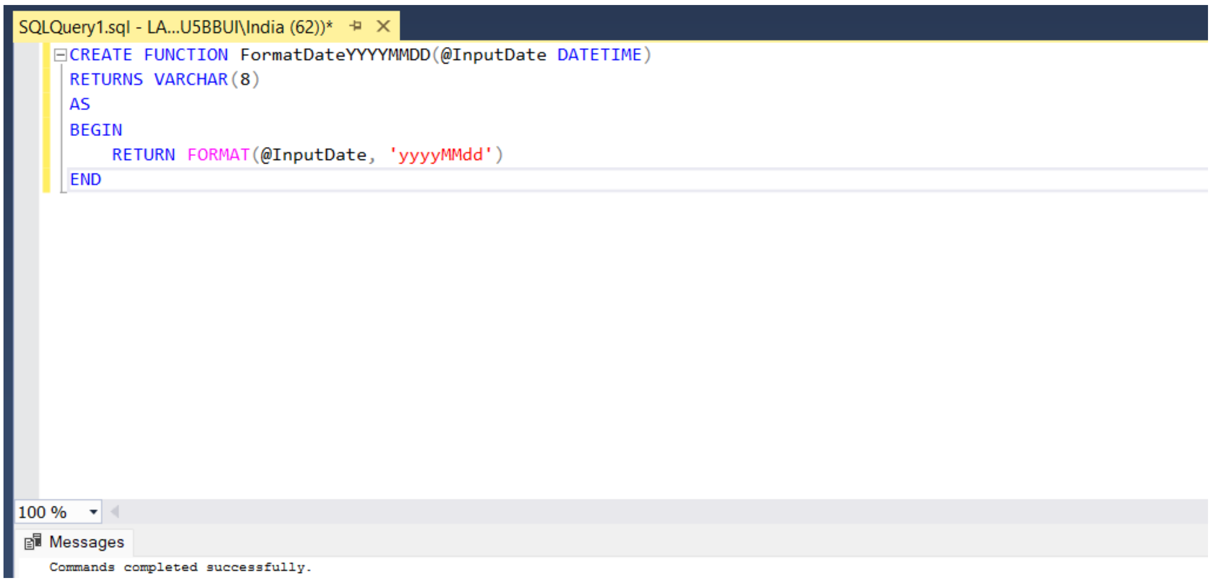
```
BEGIN

   IF NOT EXISTS (SELECT 1 FROM [Order Details] WHERE OrderID = @OrderID AND ProductID =
@ProductID)

   BEGIN

      PRINT 'Order details not found'

      RETURN -1

   END


   DECLARE @Quantity INT

   SELECT @Quantity = Quantity FROM [Order Details]

   WHERE OrderID = @OrderID AND ProductID = @ProductID


   DELETE FROM [Order Details] WHERE OrderID = @OrderID AND ProductID = @ProductID


   UPDATE Products SET UnitsInStock = UnitsInStock + @Quantity WHERE ProductID =
@ProductID


   PRINT 'Order details deleted successfully'

   RETURN 0

END
```

Functions:

1. **Date Function:**
   Query:

```
CREATE FUNCTION FormatDateYYYYMMDD(@InputDate DATETIME)
RETURNS VARCHAR(8)
AS
BEGIN
   RETURN FORMAT(@InputDate, 'yyyyMMdd')
END
```

Output:



```
CREATE FUNCTION FormatDateYYYYMMDD(@InputDate DATETIME)
RETURNS VARCHAR(8)
AS
BEGIN
    RETURN FORMAT(@InputDate, 'yyyyMMdd')
END
```

```
Commands completed successfully.
```

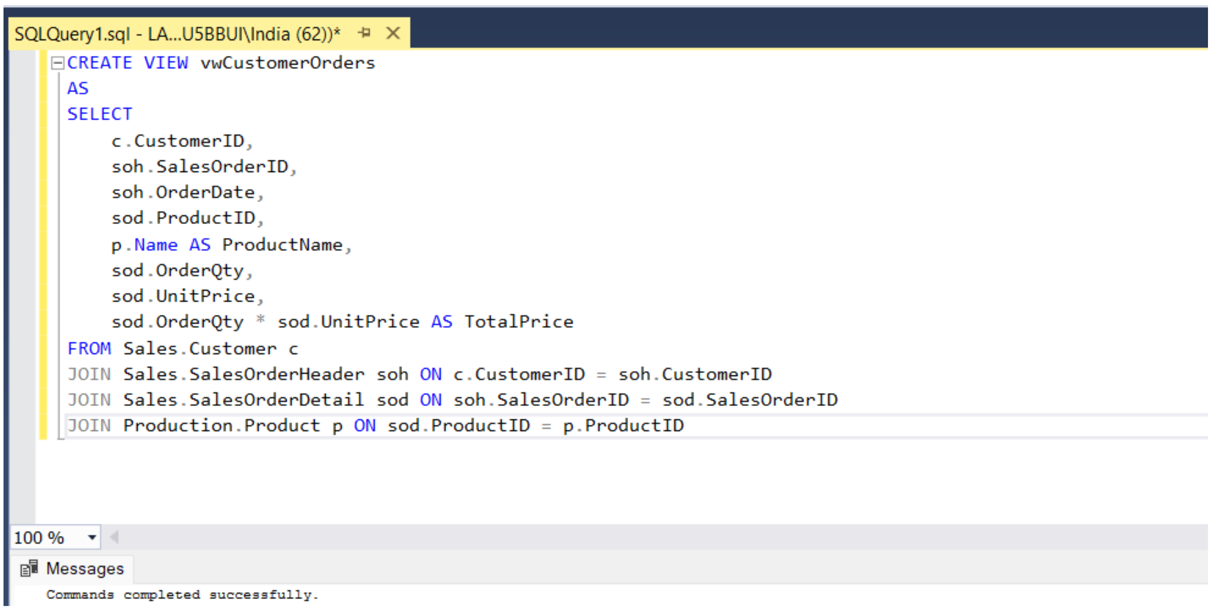
Views:

1. Customer Orders View

Query:

```
CREATE VIEW vwCustomerOrders
AS
SELECT
   c.CustomerID,
   soh.SalesOrderID,
   soh.OrderDate,
   sod.ProductID,
   p.Name AS ProductName,
   sod.OrderQty,
   sod.UnitPrice,
   sod.OrderQty * sod.UnitPrice AS TotalPrice
FROM Sales.Customer c
JOIN Sales.SalesOrderHeader soh ON c.CustomerID = soh.CustomerID
JOIN Sales.SalesOrderDetail sod ON soh.SalesOrderID = sod.SalesOrderID
JOIN Production.Product p ON sod.ProductID = p.ProductID
```

Output:



```
CREATE VIEW vwCustomerOrders
AS
SELECT
    c.CustomerID,
    soh.SalesOrderID,
    soh.OrderDate,
    sod.ProductID,
    p.Name AS ProductName,
    sod.OrderQty,
    sod.UnitPrice,
    sod.OrderQty * sod.UnitPrice AS TotalPrice
FROM Sales.Customer c
JOIN Sales.SalesOrderHeader soh ON c.CustomerID = soh.CustomerID
JOIN Sales.SalesOrderDetail sod ON soh.SalesOrderID = sod.SalesOrderID
JOIN Production.Product p ON sod.ProductID = p.ProductID
```

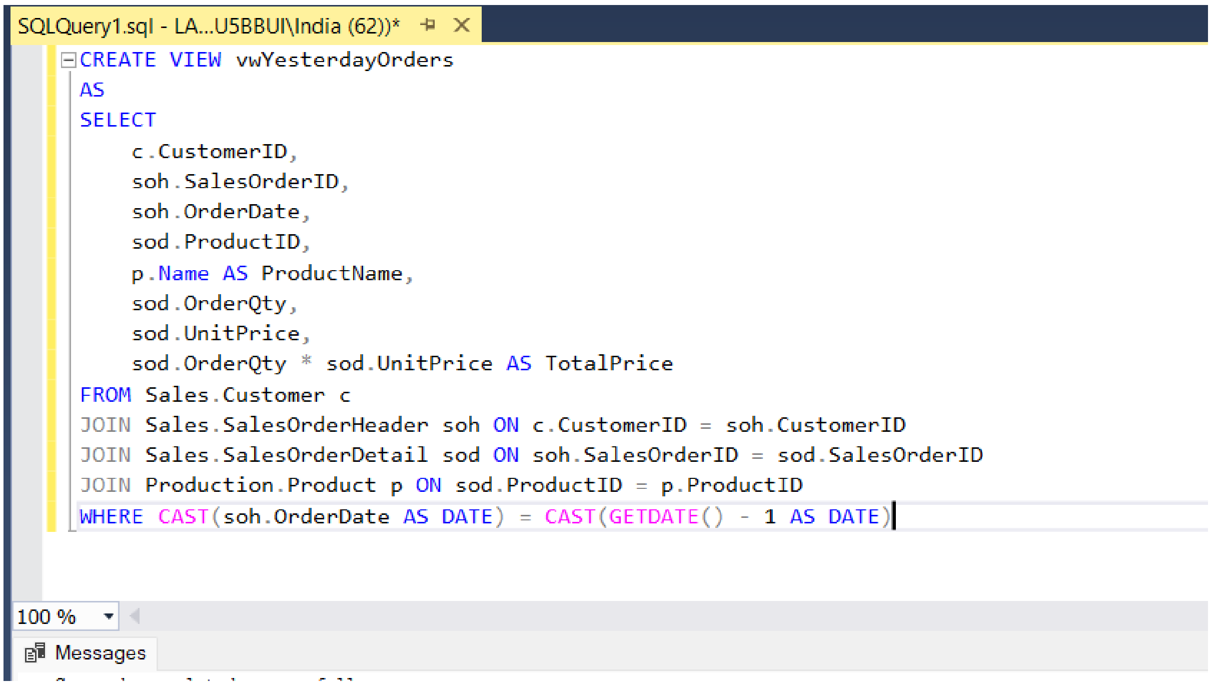
```
Commands completed successfully.
```

2. Yesterday Orders View:

Query:

```
CREATE VIEW vwYesterdayOrders

AS

SELECT

   c.CustomerID,

   soh.SalesOrderID,

   soh.OrderDate,

   sod.ProductID,

   p.Name AS ProductName,

   sod.OrderQty,

   sod.UnitPrice,

   sod.OrderQty * sod.UnitPrice AS TotalPrice

FROM Sales.Customer c

JOIN Sales.SalesOrderHeader soh ON c.CustomerID = soh.CustomerID

JOIN Sales.SalesOrderDetail sod ON soh.SalesOrderID = sod.SalesOrderID

JOIN Production.Product p ON sod.ProductID = p.ProductID

WHERE CAST(soh.OrderDate AS DATE) = CAST(GETDATE() - 1 AS DATE)
```

Output:


SQLQuery1.sql - LA...U5BBUI\India (62))*

```
CREATE VIEW vwYesterdayOrders
AS
SELECT
    c.CustomerID,
    soh.SalesOrderID,
    soh.OrderDate,
    sod.ProductID,
    p.Name AS ProductName,
    sod.OrderQty,
    sod.UnitPrice,
    sod.OrderQty * sod.UnitPrice AS TotalPrice
FROM Sales.Customer c
JOIN Sales.SalesOrderHeader soh ON c.CustomerID = soh.CustomerID
JOIN Sales.SalesOrderDetail sod ON soh.SalesOrderID = sod.SalesOrderID
JOIN Production.Product p ON sod.ProductID = p.ProductID
WHERE CAST(soh.OrderDate AS DATE) = CAST(GETDATE() - 1 AS DATE)
```

3. MyProducts View:

Query:

CREATE VIEW MyProducts

AS

SELECT

p.ProductID,

p.Name AS ProductName,

p.ProductNumber,

p.ListPrice,

pc.Name AS CategoryName

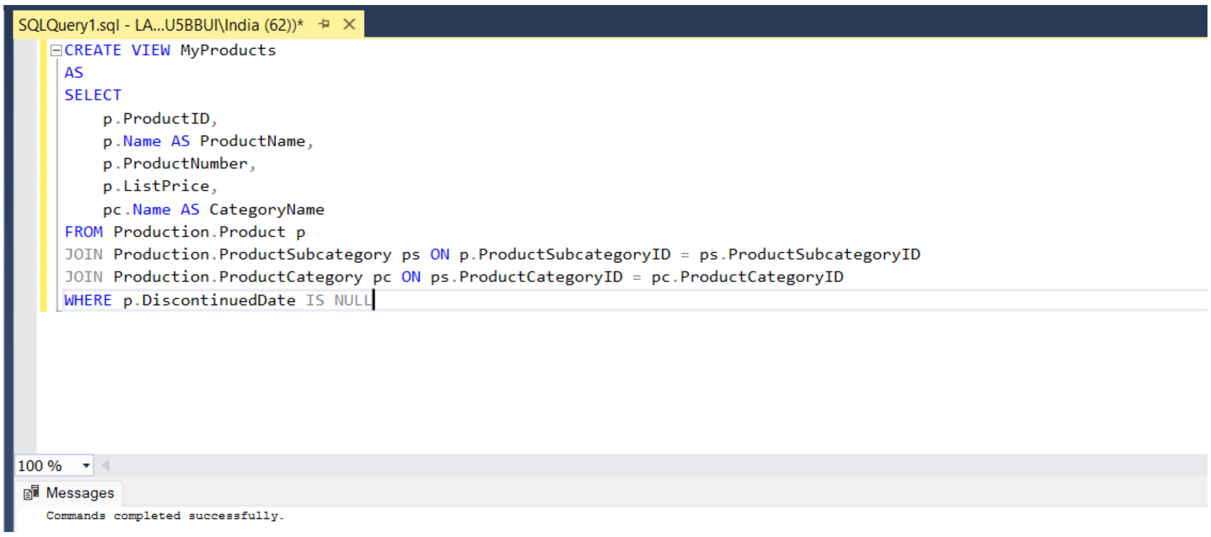
FROM Production.Product p

JOIN Production.ProductSubcategory ps ON p.ProductSubcategoryID = ps.ProductSubcategoryID

JOIN Production.ProductCategory pc ON ps.ProductCategoryID = pc.ProductCategoryID

WHERE p.DiscontinuedDate IS NULL

Output:



```
CREATE VIEW MyProducts
AS
SELECT
    p.ProductID,
    p.Name AS ProductName,
    p.ProductNumber,
    p.ListPrice,
    pc.Name AS CategoryName
FROM Production.Product p
JOIN Production.ProductSubcategory ps ON p.ProductSubcategoryID = ps.ProductSubcategoryID
JOIN Production.ProductCategory pc ON ps.ProductCategoryID = pc.ProductCategoryID
WHERE p.DiscontinuedDate IS NULL
```

```
Commands completed successfully.
```

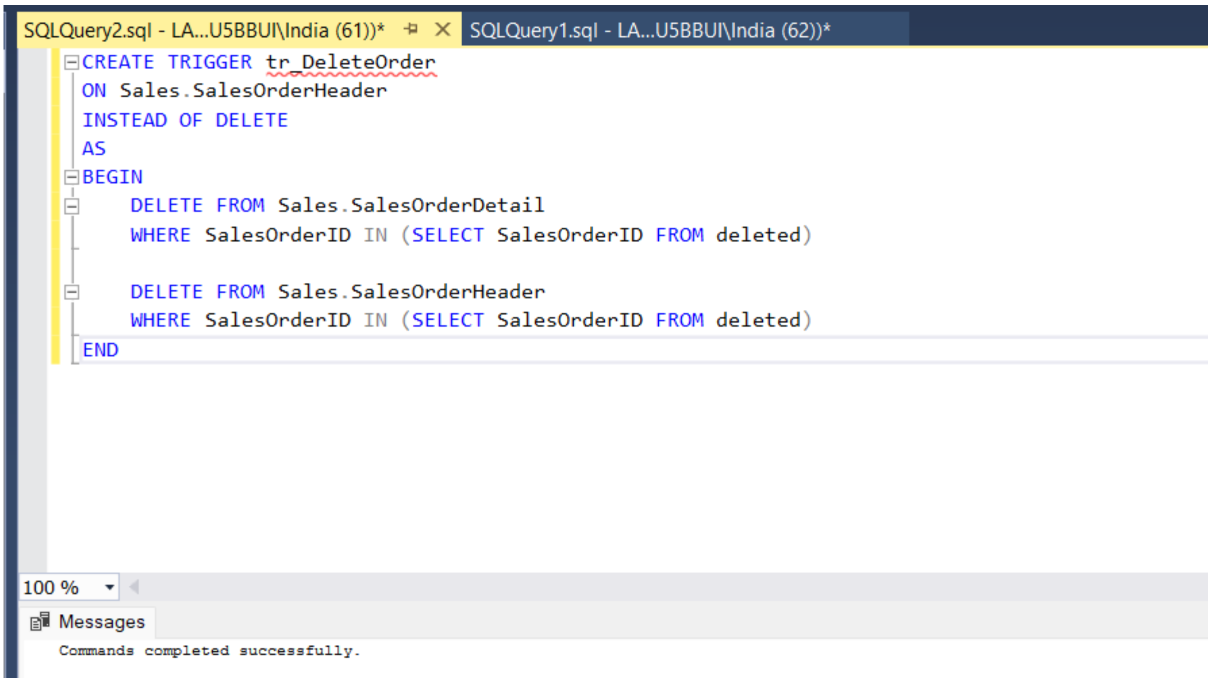
Triggers

1. Delete Trigger:

Query: CREATE TRIGGER tr_DeleteOrder

```
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
```

Output:


SQLQuery2.sql - LA...U5BBUI\India (61))* SQLQuery1.sql - LA...U5BBUI\India (62))*

```
CREATE TRIGGER tr_DeleteOrder
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
```

```
Commands completed successfully.
```

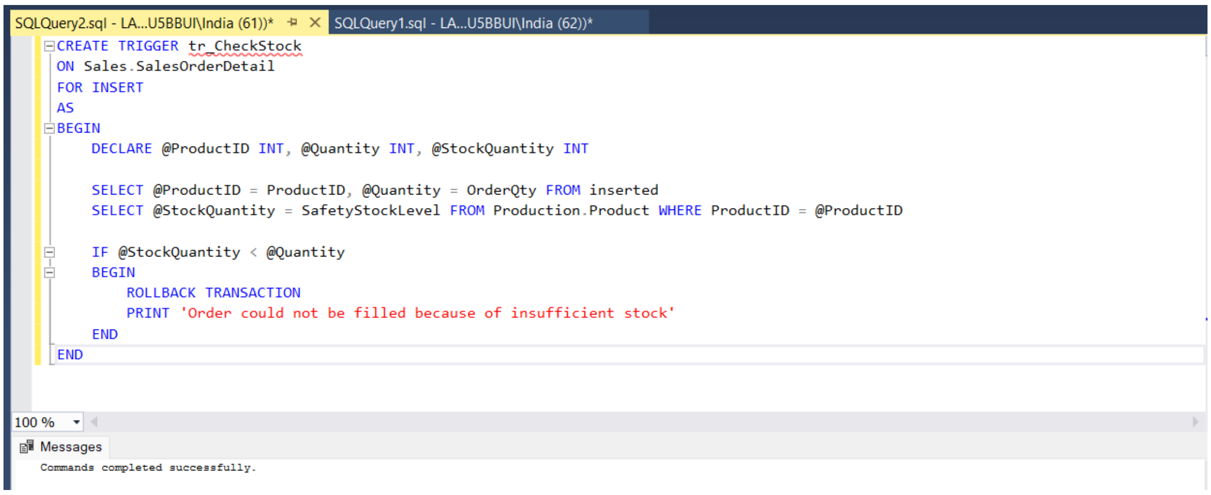
2. **Stock Check Trigger**

Query:

```
CREATE TRIGGER tr_CheckStock
ON Sales.SalesOrderDetail
FOR INSERT
AS
BEGIN
  DECLARE @ProductID INT, @Quantity INT, @StockQuantity INT

  SELECT @ProductID = ProductID, @Quantity = OrderQty FROM inserted
  SELECT @StockQuantity = SafetyStockLevel FROM Production.Product WHERE ProductID =
@ProductID

  IF @StockQuantity < @Quantity
  BEGIN
    ROLLBACK TRANSACTION
    PRINT 'Order could not be filled because of insufficient stock'
  END
```

END

Output:


SQLQuery2.sql - LA...U5BBUI\India (61))* SQLQuery1.sql - LA...U5BBUI\India (62))*

```
CREATE TRIGGER tr_CheckStock
ON Sales.SalesOrderDetail
FOR INSERT
AS
BEGIN
    DECLARE @ProductID INT, @Quantity INT, @StockQuantity INT

    SELECT @ProductID = ProductID, @Quantity = OrderQty FROM inserted
    SELECT @StockQuantity = SafetyStockLevel FROM Production.Product WHERE ProductID = @ProductID

    IF @StockQuantity < @Quantity
    BEGIN
        ROLLBACK TRANSACTION
        PRINT 'Order could not be filled because of insufficient stock'
    END
END
```

100 %

Messages

```
Commands completed successfully.
```