**Name: Gaurav Shivaji Tanpure**

**PRN: 123B2B329**

**College: PCCOE**

**Domain: SQL**

## WEEK 2 TASK

- **InsertOrderDetails:**

```
CREATE PROCEDURE InsertOrderDetails
   @OrderID INT,
   @ProductID INT,
   @UnitPrice MONEY = NULL,
   @Quantity INT,
   @Discount FLOAT = 0
AS
BEGIN
   SET NOCOUNT ON;

   DECLARE @ProductPrice MONEY;
   SELECT @ProductPrice = ListPrice FROM Production.Product WHERE ProductID =
@ProductID;

   IF @UnitPrice IS NULL
      SET @UnitPrice = @ProductPrice;

   IF @Discount IS NULL
      SET @Discount = 0;

   DECLARE @Stock INT;
   SELECT @Stock = SafetyStockLevel FROM Production.Product WHERE ProductID =
@ProductID;

   IF @Stock < @Quantity
   BEGIN
      PRINT 'Failed to place the order. Please try again.';
      RETURN;
   END
```

```
    INSERT INTO Sales.SalesOrderDetail (
        SalesOrderID, ProductID, OrderQty, UnitPrice, UnitPriceDiscount, SpecialOfferID
    )
    VALUES (
        @OrderID, @ProductID, @Quantity, @UnitPrice, @Discount, 1
    );

    UPDATE Production.Product
    SET SafetyStockLevel = SafetyStockLevel - @Quantity
    WHERE ProductID = @ProductID;

    DECLARE @NewStock INT;
    SELECT @NewStock = SafetyStockLevel FROM Production.Product WHERE ProductID =
@ProductID;

    IF @NewStock < 100
        PRINT 'Warning: Stock below reorder level.';
END;
GO
```

- **UpdateOrderDetails:**

```
CREATE PROCEDURE UpdateOrderDetails

    @OrderID INT,

    @ProductID INT,

    @UnitPrice MONEY = NULL,

    @Quantity INT = NULL,

    @Discount FLOAT = NULL

AS

BEGIN

    SET NOCOUNT ON;


    UPDATE Sales.SalesOrderDetail

    SET

        OrderQty = ISNULL(@Quantity, OrderQty),
```

```
        UnitPrice = ISNULL(@UnitPrice, UnitPrice),
        UnitPriceDiscount = ISNULL(@Discount, UnitPriceDiscount)
    WHERE SalesOrderID = @OrderID AND ProductID = @ProductID;
END;
GO
```

- **GetOrderDetails:**

```
CREATE PROCEDURE GetOrderDetails
    @OrderID INT
AS
BEGIN
    SET NOCOUNT ON;

    IF NOT EXISTS (
        SELECT 1 FROM Sales.SalesOrderDetail WHERE SalesOrderID = @OrderID
    )
    BEGIN
        PRINT 'The OrderID ' + CAST(@OrderID AS VARCHAR) + ' does not exist';
        RETURN;
    END

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

- **DeleteOrderDetails:**

```
CREATE PROCEDURE DeleteOrderDetails
    @OrderID INT,
    @ProductID INT
AS
BEGIN
    SET NOCOUNT ON;

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

- **Function: Format_MMDDYYYY:**

```
CREATE FUNCTION Format_MMDDYYYY (@inputDate DATETIME)
RETURNS VARCHAR(10)
AS
BEGIN
    RETURN CONVERT(VARCHAR(10), @inputDate, 101);
END;
GO
```

- **Function: Format_YYYYMMDD:**

```
CREATE FUNCTION Format_YYYYMMDD (@inputDate DATETIME)
RETURNS VARCHAR(8)
```

```
AS
BEGIN
    RETURN CONVERT(VARCHAR(8), @inputDate, 112);
END;
GO
```

- **View: vwCustomerOrders:**

```
CREATE VIEW vwCustomerOrders AS
SELECT
    c.CompanyName,
    o.SalesOrderID AS OrderID,
    o.OrderDate,
    od.ProductID,
    p.Name AS ProductName,
    od.OrderQty AS Quantity,
    od.UnitPrice,
    od.OrderQty * od.UnitPrice AS TotalPrice
FROM
    Sales.SalesOrderHeader o
    JOIN Sales.SalesOrderDetail od ON o.SalesOrderID = od.SalesOrderID
    JOIN Production.Product p ON od.ProductID = p.ProductID
    JOIN Sales.Customer c ON o.CustomerID = c.CustomerID;
GO
```

- **View: vwCustomerOrders_Yesterday:**

```
CREATE VIEW vwCustomerOrders_Yesterday AS

SELECT * FROM vwCustomerOrders

WHERE CONVERT(DATE, OrderDate) = CONVERT(DATE, DATEADD(DAY, -1,
GETDATE()));

GO
```

- **View: MyProducts:**

```
CREATE VIEW MyProducts AS

SELECT

   p.ProductID,

   p.Name AS ProductName,

   p.Size AS QuantityPerUnit,

   p.StandardCost 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 sc ON p.ProductSubcategoryID =
sc.ProductSubcategoryID

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

WHERE

   p.DiscontinuedDate IS NULL OR p.DiscontinuedDate IS NULL;

GO
```

- **Trigger: trg_InsteadOfDeleteOrder:**

```
CREATE TRIGGER trg_InsteadOfDeleteOrder
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
```

- **Trigger: trg_CheckStockBeforeInsert:**

```
CREATE TRIGGER trg_CheckStockBeforeInsert
ON Sales.SalesOrderDetail
INSTEAD OF INSERT
AS
BEGIN
   DECLARE @ProductID INT, @Quantity INT, @Available INT, @OrderID INT;

   SELECT TOP 1
      @ProductID = ProductID,
      @Quantity = OrderQty,
```

```
    @OrderID = SalesOrderID
  FROM INSERTED;

  SELECT @Available = SafetyStockLevel
  FROM Production.Product
  WHERE ProductID = @ProductID;

  IF @Available >= @Quantity
  BEGIN
    INSERT INTO Sales.SalesOrderDetail (
      SalesOrderID, ProductID, OrderQty, UnitPrice, UnitPriceDiscount, SpecialOfferID
    )
    SELECT
      SalesOrderID, ProductID, OrderQty, UnitPrice, UnitPriceDiscount, 1
    FROM INSERTED;

    UPDATE Production.Product
    SET SafetyStockLevel = SafetyStockLevel - @Quantity
    WHERE ProductID = @ProductID;
  END
  ELSE
  BEGIN
    PRINT 'Order could not be placed due to insufficient stock.';
  END
END;
GO
```