JASON CHAN A00698160 - TERM PROJECT PART D

```
-- Term Project PartD.D1
CREATE PROCEDURE sp_emp_info
(       @EmployeeID          int     OUTPUT        )
AS
SELECT EmployeeID                  AS EmployeeID,
       LastName                    AS LastName,
       FirstName                   AS FirstName,
       Phone                       AS Phone
FROM Employees
WHERE EmployeeID = @EmployeeID
GO


-- Term Project PartD.D2
CREATE PROCEDURE sp_orders_by_dates
(
       @StartDate    date,
       @EndDate      date
)
AS
SELECT o.OrderID            AS OrderID,
       o.CustomerID         AS CustomerID,
       c.CompanyName        AS CompanyName,
       s.CompanyName        AS ShipperName,
       o.ShippedDate        AS ShippedDate
FROM Orders o
INNER JOIN Customers c             ON o.CustomerID = c.CustomerID
INNER JOIN Shippers s              ON o.ShipperID = s.ShipperID
WHERE o.ShippedDate BETWEEN @StartDate AND @EndDate
GO


-- Term Project PartD.D3
CREATE PROCEDURE sp_products
(
       @ProductName  nvarchar(40),
       @Month               VARCHAR(20),
       @Year                int
)
AS
SELECT p.ProductName AS ProductName,
          p.UnitPrice             AS UnitPrice,
          p.UnitsInStock    AS UnitInStock,
          s.Name                  AS Name
FROM Products p
INNER JOIN Suppliers s             ON p.SupplierID = s.SupplierID
INNER JOIN OrderDetails od  ON p.ProductID = od.ProductID
INNER JOIN Orders o                ON od.OrderID = o.OrderID
WHERE p.ProductName LIKE @ProductName
  AND DATENAME(MONTH,o.OrderDate) LIKE @Month
  AND YEAR(o.OrderDate) = @Year
GO
```

```sql
-- Term Project PartD.D4
CREATE PROCEDURE sp_unit_prices
(
       @FirstValue   money,
       @SecondValue  money
)
AS
SELECT ProductID,
       ProductName,
       EnglishName,
       UnitPrice
FROM Products
WHERE UnitPrice BETWEEN @FirstValue AND @SecondValue
GO


-- Term Project PartD.D5
CREATE PROCEDURE sp_customer_city
(
       @City         VARCHAR(30)
)
AS
SELECT CustomerID,
          CompanyName,
          Address,
          City,
          Phone
FROM Customers
WHERE City LIKE @City
GO


-- Term Project PartD.D6
CREATE PROCEDURE sp_reorder_qty
(
       @UnitValue           int
)
AS
SELECT p.ProductID,
       p.ProductName,
       s.Name,
       p.UnitsInStock,
       p.ReorderLevel
FROM Products p
INNER JOIN Suppliers s            ON p.SupplierID = s.SupplierID
WHERE (p.UnitsInStock - p.ReorderLevel) < @UnitValue
GO
```

```
-- Term Project PartD.D7
CREATE PROCEDURE sp_shipping_date
(
       @ShippedDate  date
)
AS
SELECT o.OrderID,
       c.CompanyName                            AS CustomerName,
          s.CompanyName                         AS ShipperName,
          o.OrderDate,
          CONVERT(datetime, @ShippedDate)       AS ShippedDate
FROM Orders o
INNER JOIN Customers c            ON o.CustomerID = c.CustomerID
INNER JOIN Shippers s             ON o.ShipperID = s.ShipperID
WHERE DATEADD(DAY, 10, o.orderDate) = @ShippedDate
GO


-- Term Project PartD.D8
CREATE PROCEDURE sp_del_inactive_cust
AS
DELETE c
FROM Customers c
LEFT JOIN Orders o          ON c.CustomerID = o.CustomerID
WHERE o.CustomerID IS NULL


-- Term Project PartD.D9
CREATE TRIGGER tr_check_qty
ON OrderDetails
FOR UPDATE
AS
DECLARE              @Qty          int
DECLARE              @UnitsInStock int
SELECT        @Qty = Quantity FROM INSERTED
SELECT        @UnitsInStock = p.UnitsInStock
FROM INSERTED i
INNER JOIN products p ON i.ProductID = p.ProductID
IF(@Qty > @UnitsInStock)
       BEGIN
              PRINT ('Units In Stock is insufficient for the order quantity. Please enter
                     a new Quantity.')
              ROLLBACK TRANSACTION
       END
```

```
-- Term Project PartD.D10
CREATE TRIGGER tr_insert_shippers
ON Shippers
INSTEAD OF INSERT
AS
BEGIN
	IF EXISTS
	(	SELECT * FROM Shippers s
		INNER JOIN INSERTED i ON i.CompanyName LIKE s.CompanyName )
		BEGIN
			PRINT 'Shipper already exists'
		END
	ELSE
		BEGIN
			INSERT INTO Shippers
			SELECT * FROM INSERTED i
		END
END
```