Create a stored procedure called sp\_emp\_info to display the employee id, last name, first name, and phone number from the Employees table for a particular employee. The employee id will be an input parameter for the stored procedure.

-- D1

CREATE PROCEDURE sp\_emp\_info

( @parameter1 int )

AS

SELECT EmployeeID,

LastName,

FirstName,

Phone

FROM Employees

WHERE EmployeeID = @parameter1

GO

Create a stored procedure called sp\_orders\_by\_dates displaying the orders shipped between particular dates. The start and end date will be input parameters for the stored procedure. Display the order id, customer id, and shipped date from the Orders table, the company name from the Customer table, and the shipper name from the Shippers table.

-- D2

CREATE PROCEDURE sp\_orders\_by\_dates

( @parameter1 date,

@parameter2 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 @parameter1 AND @parameter2

GO

Create a stored procedure called sp\_products listing a specified product ordered during a specified month and year. The product name, month, and year will be input parameters for the stored procedure. Display the product name, unit price, and units in stock from the Products table, and the supplier name from the Suppliers table.

-- D3

CREATE PROCEDURE sp\_products

( @parameter1 varchar(30),

@parameter2 varchar(30),

@parameter3 int )

AS

SELECT p.ProductName,

p.UnitPrice,

p.UnitsInStock,

s.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 @parameter1

AND DATEPART(MM, o.OrderDate) = DATEPART(MM,@parameter2 + ' 1 2000')

AND DATEPART(yy, o.OrderDate) = @parameter3

GO

Create a stored procedure called sp\_unit\_prices listing the products where the unit price is between particular values. The two unit prices will be input parameters for the stored procedure. Display the product id, product name, English name, and unit price from the Products table.

-- D4

CREATE PROCEDURE sp\_unit\_prices

( @parameter1 money,

@parameter2 money )

AS

SELECT ProductID,

ProductName,

EnglishName,

UnitPrice

FROM Products

WHERE UnitPrice BETWEEN @parameter1 AND @parameter2

GO

Create a stored procedure called sp\_customer\_city displaying the customers living in a particular city. The city will be an input parameter for the stored procedure. Display the customer id, company name, address, city and phone from the Customers table.

-- D5

CREATE PROCEDURE sp\_customer\_city

( @parameter1 nvarchar(15) )

AS

SELECT CustomerID,

CompanyName,

Address,

City,

Phone

FROM Customers

WHERE City = @parameter1

GO

Create a stored procedure called sp\_reorder\_qty to show when the reorder level subtracted from the units in stock is less than a specified value. The unit value will be an input parameter for the stored procedure. Display the product id, product name, units in stock, and reorder level from the Products table, and the supplier name from the Suppliers table.

-- D6

CREATE PROCEDURE sp\_reorder\_qty

( @parameter1 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) < @parameter1

GO

Create a stored procedure called sp\_shipping\_date where the shipped date is equal to the order date plus 10 days. The shipped date will be an input parameter for the stored procedure. Display the order id, order date and shipped date from the Orders table, the company name from the Customers table, and the company name from the Shippers table.

-- D7

CREATE PROCEDURE sp\_shipping\_date

( @parameter1 smalldatetime )

AS

SELECT o.OrderID AS OrderID,

c.CompanyName AS CustomerName,

s.CompanyName AS ShipperName,

o.OrderDate AS OrderDate,

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 = @parameter1

AND DATEDIFF(day, o.OrderDate, o.ShippedDate) = 10

GO

Create a stored procedure called sp\_del\_inactive\_cust to delete customers that have no orders. Use the following query to test your procedure. The stored procedure should delete 1 row.

-- D8

CREATE PROCEDURE sp\_del\_inactive\_cust

AS

DELETE c

FROM Customers c

WHERE NOT EXISTS

(SELECT \*

FROM Orders o

WHERE o.CustomerID = c.CustomerID

)

GO

Create an UPDATE trigger called tr\_check\_qty on the OrderDetails table to prevent the updating of orders of products that have units-in-stock less than the quantity ordered.

-- D9

CREATE TRIGGER tr\_check\_qty

ON OrderDetails

FOR UPDATE

AS

IF (SELECT Quantity FROM inserted) >

(SELECT UnitsInStock

FROM Products

WHERE (SELECT ProductID FROM inserted) = Products.ProductID)

BEGIN

PRINT 'Quantity cannot exceed units in stock'

ROLLBACK TRANSACTION

END

Create an INSTEAD OF INSERT trigger called tr\_insert\_shippers on the Shippers table preventing inserting a row with a company name which already exists.

-- D10

CREATE TRIGGER tr\_insert\_shippers

ON Shippers

INSTEAD OF INSERT

AS

IF (SELECT CompanyName FROM inserted) IN

(SELECT CompanyName

FROM Shippers)

BEGIN

PRINT 'Shipper Company Name already exists'

END

ELSE

BEGIN

INSERT INTO Shippers

SELECT \*

FROM inserted

END