**IST 278 Unit 3 Lab**

Name: Michael Bacu Date: 6/10/2023

# **Lab Instructions**

1. Type your name and the date in the spaces provided.
2. Use the SQL Server Management Studio and the AP database and the IST278EagleCorp database as specified below.
3. Complete all seven exercises in this lab (exercise 1, 2a, 2b, 2c, 2d, 3, and 4)
4. Upload and submit a completed copy of this lab sheet before the due date.

**Stored Procedures Lab**

# **1. Unit 3 Lab Exercise 1**

Use the IST278EagleCorp database and write a stored procedure named spOrderQuantityRangeXX where the XX are your initials. This stored procedure is to receive three optional parameters and use these parameters to filter to reduce the number of rows returned in the result set. The parameter @PartNumberVar is a mask that is to be used with a LIKE operator to filter by part number. The @OrderMin and @OrderMax are parameters to be used to specify the requested range of OrderQuantities.

Code this stored procedure to have it:

* Return a result set consisting of CustomerID, OrderID, PartNumber, and OrderQuantity for each order that meets the filtering criteria, sorted by OrderQuantity with the largest quantity appearing first.
* Handle a call where no value is passed for the @PartNumberVar parameter by using a % as the value for the @PartNumberVar so that all part numbers pass the part number filter.
* Handle a call where no values are passed to the parameters by returning the CustomerID, OrderID, PartNumber, and OrderQuantity for all orders.
* Handle a call where a zero or negative number is passed to the @OrderMin or @OrderMax parameters, by using a value of 0 for the @OrderMin and 99999999 for the @OrderMax so that all OrderQuantities pass the order quantity filter.

Hint: Work page 507 exercise #1 & #2 using the AP database before attempting Lab exercise 1 & 2. Those exercises are similar.

**---** **Paste below this line the Stored Procedure you wrote for this exercise --**

USE IST278EagleCorp;

GO

CREATE PROCEDURE spOrderQuantityRangeMB

@PartNumberVar NVARCHAR(255) = NULL,

@OrderMin INT = NULL,

@OrderMax INT = NULL

AS

BEGIN

-- Set default values if no values or invalid values are passed

SET @PartNumberVar = COALESCE(@PartNumberVar, '%');

SET @OrderMin = COALESCE(NULLIF(@OrderMin, 0), NULLIF(@OrderMin, -1), 0);

SET @OrderMax = COALESCE(NULLIF(@OrderMax, 0), NULLIF(@OrderMax, -1), 99999999);

-- Query with the provided filters

SELECT CustomerID, OrderID, PartNumber, OrderQuantity

FROM Orders

WHERE PartNumber LIKE @PartNumberVar

AND OrderQuantity BETWEEN @OrderMin AND @OrderMax

ORDER BY OrderQuantity DESC;

END;

GO

# **2a. Unit 3 Lab Exercise 2a**

Code a call to the procedure you created in exercise 1 using pass by position to pass to the @PartNumberVar the value 'M%'. Do not pass values to the other parameters.

**--- Paste below this line the statement(s) you wrote you wrote for 2a –**

EXEC spOrderQuantityRangeMB 'M%';

**---** **Type below this line the Number of rows returned in the run results from executing the 2a call (Do not paste the rows returned for this one)--**

**How Many Rows Where Returned: 2**

# **2b. Unit 3 Lab Exercise 2b**

Code a call to the procedure you created in exercise 1 using pass by name to pass a 27 to the @OrderMin and a 39 to the @OrderMax. Do not pass a value to the @PartNumberVar parameter.

**-- Paste below this line the statement(s) you wrote you wrote for 2b –**

EXEC spOrderQuantityRangeMB @OrderMin = 27, @OrderMax = 39;

**---** **Paste below this line the run results from executing the 2b call--**

A screenshot of a computer

Description automatically generated

# **2c. Unit 3 Lab Exercise 2c**

Code a call to the procedure you created in exercise 1 using pass by position to pass a 0 to the @OrderMin, pass a 0 to the @OrderMax , and pass to the @PartNumberVar a mask value that will only allow the selection of part numbers that begin with M or K or P.

**-- Paste below this line the statement(s) you wrote you wrote for 2c –**

EXEC spOrderQuantityRangeMB '[MKP]%';

**--- Type below this line the Number of rows returned in the run results from executing the 2c call (Do not paste the rows returned for this one)--**

**How Many Rows Where Returned: 4**

# **2d. Unit 3 Lab Exercise 2d**

Code a call to the procedure you created in exercise 1 using pass by position to pass a 21 to the @OrderMin, pass a 45 to the @OrderMax , and pass to the @PartNumberVar a mask value that will only allow the selection of part numbers that begin with M or K or P.

**-- Paste below this line the statement(s) you wrote you wrote for 2d –**

EXEC spOrderQuantityRangeMB '[MKP]%', 21, 45;

**---** **Paste below this line the run results from executing the 2d call--**

A screenshot of a computer

Description automatically generated

# **3. Unit 3 Lab Exercise 3**

Use the AP database and write a stored procedure named spDateRangeXX where the XX are your initials. This stored procedure is to accept two parameters, @DateMinXX and @DateMaxXX (where the XX are your initals). Each of these parameters are to have a data type of varchar and default value null. If the procedure is called with no parameters or with null values, have it raise an error that describes the problem. If the procedure is called with non-null values, validate the parameters. Test that the literal strings are valid dates and test that @DateMinXX is earlier than @DateMaxXX. If the parameters are valid, return a result set that consists of the VendorID, InvoiceNumber, InvoiceDate, and InvoiceTotal for each invoice for which the InvoiceDate is within the date range, sorted with earliest invoice first.

Hint: Work page 507 exercise #3 & #4 using the AP database before attempting Lab exercise 3 & 4. Those exercises are similar.

**--- Paste below this line the Stored Procedure you wrote for this exercise --**

USE AP;

GO

CREATE PROCEDURE spDateRangeMB

@DateMinMB VARCHAR(255) = NULL,

@DateMaxMB VARCHAR(255) = NULL

AS

BEGIN

IF @DateMinMB IS NULL OR @DateMaxMB IS NULL

BEGIN

RAISERROR('Both date parameters must have a value.', 16, 1);

RETURN;

END

IF ISDATE(@DateMinMB) = 0 OR ISDATE(@DateMaxMB) = 0

BEGIN

RAISERROR('Invalid date format provided for one or both date parameters.', 16, 1);

RETURN;

END

DECLARE @MinDate DATE = CAST(@DateMinMB AS DATE);

DECLARE @MaxDate DATE = CAST(@DateMaxMB AS DATE);

IF @MinDate > @MaxDate

BEGIN

RAISERROR('@DateMinMF must be earlier than @DateMaxMF.', 16, 1);

RETURN;

END

SELECT VendorID,InvoiceNumber,InvoiceDate,InvoiceTotal

FROM Invoices

WHERE InvoiceDate BETWEEN @MinDate AND @MaxDate

ORDER BY InvoiceDate;

END;

GO

# **4. Unit 3 Lab Exercise 4**

Code a call to the stored procedure created in unit 3 lab exercise 3 that returns results for invoices with an InvoiceDate between December 25, 2019 and January 11, 2020. This call should also catch any errors that are raised by the procedure and print the error number and description.

**-- Paste below this line the statement(s) you wrote you wrote for exercise 4 –**

BEGIN TRY

EXEC spDateRangeMB @DateMinMB = '2019-12-25', @DateMaxMB = '2020-01-11';

END TRY

BEGIN CATCH

PRINT 'Error Number: ' + CAST(ERROR\_NUMBER() AS VARCHAR(255));

PRINT 'Error Description: ' + ERROR\_MESSAGE();

END CATCH;

**---** **Paste below this line the run results from executing the code you wrote for exercise 4--**

A screenshot of a computer

Description automatically generated with medium confidence