**IST 278 Unit 3 Lab**

Name: Ramon Rodriguez Date: 5/28/2024

# **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 --**

Paste here

USE [IST278EagleCorp13-1]

GO

DROP PROC IF EXISTS spOrderQuantityRangeRR;

GO

CREATE PROC spOrderQuantityRangeRR

@PartNumberVar varchar(40) = '%',

@OrderMin INT = NULL,

@OrderMax INT = NULL

AS

IF @OrderMin IS NULL OR @OrderMin < 0

SET @OrderMin = 0;

IF @OrderMax IS NULL OR @OrderMax <= 0

SET @OrderMax = 99999999;

IF (@OrderMin IS NULL AND @OrderMax IS NULL)

BEGIN

SELECT co.CustomerID, co.OrderID, col.PartNumber, col.OrderQuantity FROM CustOrder co

JOIN CustOrderLine col ON co.OrderID = col.OrderID ORDER BY col.OrderQuantity DESC

END

SELECT co.CustomerID, co.OrderID, col.PartNumber, col.OrderQuantity FROM CustOrder co

JOIN CustOrderLine col ON co.OrderID = col.OrderID

WHERE col.PartNumber LIKE @PartNumberVar AND col.OrderQuantity BETWEEN @OrderMin and @OrderMax

ORDER BY col.OrderQuantity DESC

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 –**

Paste here

EXEC spOrderQuantityRangeRR '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: 500**

# **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 –**

Paste here

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

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

Paste here

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 –**

Paste here

EXEC spOrderQuantityRangeRR '[MKP]%', 0, 0;

**--- 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: 851**

# **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 –**

Paste here

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

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

Paste here

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 --**

Paste here

USE AP;

GO

DROP PROCEDURE IF EXISTS spDateRangeRR;

GO

CREATE PROCEDURE spDateRangeRR

@DateMinRR VARCHAR(15) = NULL,

@DateMaxRR VARCHAR(15) = NULL

AS

IF (@DateMinRR IS NULL AND @DateMaxRR IS NULL)

THROW 51000, 'Error: Both minimum and maximum dates for the date range are NULL. Please provide valid dates.', 1;

BEGIN

DECLARE @MinDate DATE

DECLARE @MaxDate DATE

BEGIN TRY

SET @MinDate = CAST(@DateMinRR AS DATE)

SET @MaxDate = CAST(@DateMaxRR AS DATE)

END TRY

BEGIN CATCH

THROW 51000, 'Error: Invalid date format. Dates must be in YYYY-MM-DD format.', 1;

END CATCH

END

IF @MinDate > @MaxDate

THROW 51000, 'Error: @DateMinRR must be earlier than @DateMaxRR.', 1;

SELECT VendorID, InvoiceNumber, InvoiceDate, InvoiceTotal

FROM Invoices

WHERE InvoiceDate BETWEEN @MinDate AND @MaxDate

ORDER BY InvoiceDate ASC;

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 –**

Paste here

BEGIN TRY

EXEC spDateRangeRR @DateMinRR = '2019-12-25', @DateMaxRR = '2020-01-11';

END TRY

BEGIN CATCH

PRINT 'An error occurred.';

PRINT 'Message: ' + CONVERT(varchar(200), ERROR\_MESSAGE());

END CATCH

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

Paste here

A screenshot of a computer screen

Description automatically generated