**IST 278 Unit 1 Lab**

Name: Michael Bacu Date: 5/25/2023

# **Lab Instructions**

1. Type your name and the date in the spaces provided.
2. Use the SQL Server Management Studio and the IST278EagleCorp.
3. Complete the 4 exercises in this unit 2 lab (two advanced query exercises and two view exercises) per the directions provided in this document.
4. Upload and submit a completed copy of this lab sheet before the due date.

**Advanced Queries – Background information for exercise 1**

# **A procedure for building advanced queries**

1. State the problem to be solved in English.
2. Use English or Pseudocode to outline the query. In this step you need to identify the subqueries you will use and the data they will return. You should include the aliases you will use for any derived tables.
3. If any of the subqueries identified in step 2 are particularly involved, use English or pseudocode to outline the further details for it.
4. Code the subqueries and test them to be sure that they return the correct data.
5. Code and test the final query.

**Exercise 1 (an Advanced Query Exercise):**

Code a query that uses the IST278EagleCorp Database to produce a result set showing South Carolina and Georgia Customer order information that consists of CustomerID, OrderID, State, City, and OrderWeight

Hints:

Approach this problem using the advanced building procedure provided above.

Instructions:

Paste below the work you did to solve this problem and the run results. Include all outline / Pseudocode you wrote as well as the query code and the run results you got when executing it.

**Paste work for Exercise 1 here**

SELECT CustomerID, CustOrder.OrderID, Shipment.ShipState, Shipment.ShipCity, SUM(OrderQuantity \* InventoryPart.Weight) AS OrderWeight

FROM CustOrder

JOIN CustOrderLine ON CustOrder.OrderID = CustOrderLine.OrderID

JOIN InventoryPart ON CustOrderLine.PartNumber = InventoryPart.PartNumber

JOIN Shipment ON CustOrder.OrderID = Shipment.OrderID

WHERE ShipState IN ('SC', 'GA')

GROUP BY CustomerID, CustOrder.OrderID, Shipment.ShipState, Shipment.ShipCity

A screenshot of a computer

Description automatically generated

**Exercise 2 (an Advanced Query Exercise):**

Code a query that uses the IST278EagleCorp Database to produce a result set showing Virginia and Pennsylvania Customers that have never ordered anything. For each Virginia and Pennsylvania Customers that never ordered anything show the CustFirstName, CustLastName, and State.

Hints:

A correlated SQL subquery is an advanced type of query that you may want to use to solve this problem (refer back to chapter 6 of the book). A correlated SQL Subquery is just a subquery that is executed many times—once for each record (row) returned by the outer (main) query. In other words, the outer query returns a table with multiple rows; the inner query then runs once for each of those rows. If your outer query returns 10 rows, then the inner query will run 10 times. And if your outer query returns 100 rows, the inner query will run 100 times. The structure of a correlated subquery looks a little like a join in that the inner and outer query match up values (you can see this in example on page 199 WHERE Invoices.VendorID = Vendors.VendorID). For negative data questions a correlated subquery using NOT EXISTS (see example on page 199) is a frequently used coding technique.

Instructions:

Paste below the work you did to solve this problem and the run results. Include all outline / Pseudocode you wrote as well as the query code and the run results you got when executing it.

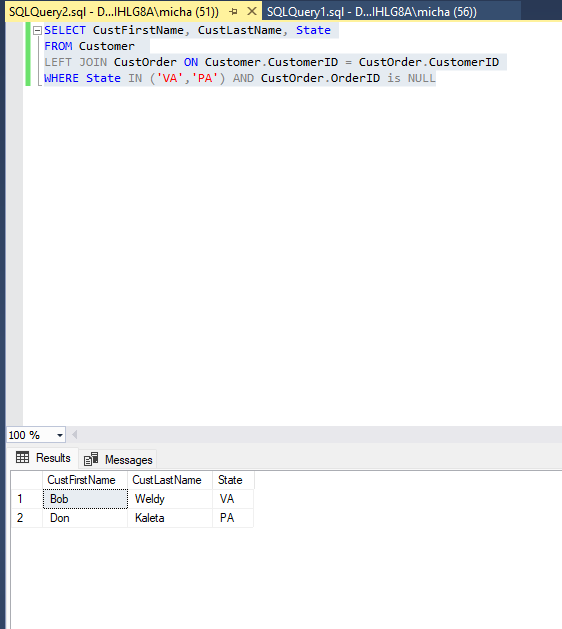
**Paste work for Exercise 2 here**

SELECT CustFirstName, CustLastName, State

FROM Customer

LEFT JOIN CustOrder ON Customer.CustomerID = CustOrder.CustomerID

WHERE State IN ('VA','PA') AND CustOrder.OrderID is NULL



**Exercise 3 (a view exercise)**

Write a CREATE VIEW statement to create a view named xx*EmployeeFortSuttonInfo* (where the xx are your initials). The view is to return the Employee’s LastName *and StreetAddress of each employee in the city of ‘Fort Sutton’*.

Paste below the code for creating your view, the code for selecting everything from it, and the results of selecting everything from it.

**Paste work for Exercise 3 here**

CREATE VIEW MBEmployeeFortSuttonInfo AS

SELECT LastName, StreetAddress

FROM Employee

WHERE City = 'Fort Sutton';

**A screenshot of a computer

Description automatically generated with medium confidence**

A screenshot of a computer

Description automatically generated

**Exercise 4 (a view exercise)**

Write a CREATE VIEW statement to create a view named xxPackagePackers (where the xx are your initials). The view is to return the Employee’s LastName, Firstname, *and a count of the number of times their employeeid appears on the PackingSlip table (name this columns PackagesPacked).*

Paste below the code for creating your view, the code for selecting everything from it, and the results of selecting everything from it.

**Paste work for Exercise 4 here**

CREATE VIEW MBPackagePackers AS

SELECT Employee.LastName, Employee.FirstName, COUNT(PackingSlip.EmployeeID) AS PackagesPacked

FROM Employee

LEFT JOIN PackingSlip ON Employee.EmployeeID = PackingSlip.EmployeeID

GROUP BY Employee.LastName, Employee.FirstName;

A screenshot of a computer program

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated