**IST 278 Unit 1 Lab**

Name: Levi Noell-Baba Date: January 12, 2024

# **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**

Exercise 1 is asking us to execute a query that returns customer information for customers in South Carolina and Georgia. The results should contain the CustomerID, OrderID, State, City, and OrderWeight. I will need to pull the information from the Customer, CustOrder, CustOrderLine, and InventoryPart.

I will use The sum of the OrderQuantity in CustOrderLine and Weight in InventoryPart as the OrderWeight Column.

SELECT C.CustomerID, C.State, C.City, CO.OrderID, SUM(COL.OrderQuantity \* IP.Weight) AS OrderWeight

FROM Customer AS C

JOIN CustOrder AS CO ON C.CustomerID = CO.CustomerID

JOIN CustOrderLine AS COL ON CO.OrderID = COL.OrderID

JOIN InventoryPart AS IP ON COL.PartNumber = IP.PartNumber

WHERE C.State IN ('GA', 'SC')

GROUP BY C.CustomerID, C.State, C.City, CO.OrderID;

102 Rows Affected

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**

Exercise 2 is asking me to code a query that shows results that show Virginia and Pennsylvania customers that have never ordered anything. For each customer from Virginia and Pennsylvania that have never ordered anything, The query should return the CustFirstName, CustLastName, and State. I will need to pull information from the CustOrder table to determine if a customer has ever ordered anything.

SELECT CustFirstName, CustLastName, State

FROM Customer

LEFT JOIN CustOrder ON Customer.CustomerID = CustOrder.CustomerID

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

A white box with black text

Description automatically generated

**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**

In Exercise 3, I have been tasked to write a create view statement to create a view name LNEmployeeFortSuttonInfo that will return the Employee’s LastName and Street Address of each employee in the city of Fort Sutton. I will also need to write a select statement that displays the LNEmployeeFortSuttonInfo view that has been created.

CREATE VIEW LNEmployeeFortSuttonInfo AS

SELECT LastName, StreetAddress

FROM Employee

WHERE City = 'Fort Sutton'

SELECT \*

FROM LNEmployeeFortSuttonInfo

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**

In Exercise 4, I have been assigned to write a create view statement to create a view named LNPackagePackers that will return the employee’s last name, first name, and a count of the number of times their employeeid appears on the packing slip table. I will also have to write a select statement that returns everything on the LNPackagePackers view and display the results.

CREATE VIEW LNPackagePackers 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;

And

SELECT \*

FROM LNPackagePackers

A screenshot of a computer

Description automatically generated