**IST 278 Unit 1 Lab**

Name: Ramon Rodriguez Date: 5/19/2024

# **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**

SELECT co.CustomerID, co.OrderID, s.ShipState, s.ShipCity,

SUM(col.OrderQuantity \* ip.Weight) AS OrderWeight

FROM

CustOrder co

JOIN

CustOrderLine col ON co.OrderID = col.OrderID

JOIN

InventoryPart ip ON col.PartNumber = ip.PartNumber

JOIN

Shipment s ON co.OrderID = s.OrderID

WHERE

s.ShipState IN ('SC', 'GA')

GROUP BY

s.ShipState,

s.ShipCity,

co.CustomerID,

co.OrderID;

CustomerID OrderID ShipState ShipCity OrderWeight

C-300011 1999000014 GA Adel 131.503

C-300011 1999000118 GA Adel 612.725

C-300011 1999000232 GA Adel 154.280

C-300011 1999000493 GA Adel 181.554

C-300011 2000000573 GA Adel 327.840

C-300011 2000000659 GA Adel 122.000

I-300086 1999000251 GA Athens 27.570

I-300086 2000000756 GA Athens 30.000

I-300095 1999000321 GA Columbus 3.128

C-300029 2000000594 GA Elberton 90.942

C-300029 2000000617 GA Elberton 339.700

C-300029 2000000705 GA Elberton 37.000

C-300029 2000000769 GA Elberton 249.780

I-300129 1999000068 GA Macon 14.500

I-300129 1999000314 GA Macon 30.500

I-300139 1999000392 GA Nahunta 5.876

I-300139 2000000510 GA Nahunta 36.320

I-300139 2000000523 GA Nahunta 4.938

I-300139 2000000608 GA Nahunta 12.252

I-300013 1999000016 GA Swainsboro 4.064

I-300013 1999000106 GA Swainsboro 32.880

I-300013 2000000680 GA Swainsboro 5.813

I-300013 2000000702 GA Swainsboro 9.688

I-300013 2000000803 GA Swainsboro 33.250

I-300016 1999000125 SC Barnwell 3.875

I-300016 1999000136 SC Barnwell 38.320

C-300043 1999000316 SC Charleston 13.500

C-300043 1999000388 SC Charleston 399.840

C-300043 1999000454 SC Charleston 4.126

C-300043 2000000531 SC Charleston 357.840

C-300056 1999000208 SC Lake City 62.642

C-300056 1999000269 SC Lake City 16.875

C-300056 1999000299 SC Lake City 39.125

C-300038 1999000026 SC Sumter 675.000

C-300038 1999000229 SC Sumter 20.752

C-300038 1999000279 SC Sumter 13.380

C-300038 2000000708 SC Sumter 13.388

**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

WHERE CustomerID NOT IN (SELECT CustomerID FROM CustOrder)

AND [State] in ('VA', 'PA');

CustFirstName CustLastName State

Don Kaleta PA

Bob Weldy VA

**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 RREmployeeFortSuttonInfo AS

SELECT LastName, StreetAddress FROM Employee

WHERE City = 'Fort Sutton';

SELECT \* FROM RREmployeeFortSuttonInfo;

**Rosner 3604 N. Howard Ave.**

**Eckelman 1362 W. Cass St.**

**Nairn 7321 N. Nebraska Ave.**

**Vigus 2403 E. 19th St.**

**Hess 2118 W. Spruce St.**

**Osman 5628 W. Evold Ave.**

**Moore 4410 W. Spruce St.**

**Brose 2176 E. 19th St.**

**Reece 1204 N. Nebraska Ave**

**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 RRPackagePackers AS

SELECT e.LastName, e.FirstName, COUNT(ps.EmployeeID) AS PackagesPacked FROM Employee e

JOIN PackingSlip ps ON e.EmployeeID = ps.EmployeeID

GROUP BY e.LastName, e.FirstName;

SELECT \* FROM RRPackagePackers;

**Ortman Austin 563**

**Keck David 109**

**Hettinger Gregory 401**

**Brose Jack 681**

**Rosner Joanne 165**

**Nairn Michelle 7**

**Albregts Nicholas 198**

**Underwood Patricha 1244**

**Reece Phil 128**