Chapter 5 Lab – Coding Summary Queries

Name: \_Bryan Bibb\_\_ Date: \_\_\_\_Sept 19, 2023\_\_\_\_

# Exercise Instructions

1. Type your name and the date into the space provided.
2. Use the SQL Server Management Studio to complete this lab.
3. Write T-SQL statements to query the tables contained in the IST272EagleCorp database and complete each of the exercises in this lab per the directions provided below.
4. Upload and submit before the due date.

1. Write a SELECT statement that returns two columns from the CustOrderLine table: PartNumber and NumOfPartsOrdered, where NumOfPartsOrdered is the sum of the OrderQuantity column. Group the result set by PartNumber.

Paste below the **code** you wrote and **type the number of rows returned** for exercise 1:

SELECT PartNumber,

SUM(OrderQuantity) AS NumOfPartsOrdered

FROM CustOrderLine

GROUP BY PartNumber;

Rows Returned: 192

------------------------------------------------

2. Write a SELECT statement that returns the CustomerID and NumOfPartsOrdered, where NumOfPartsOrdered is the total number of parts ordered by the customer (sum the OrderQuantity). Group the result set by CustomerID. Return only 3 rows, corresponding to the 3 customers who've ordered the most parts.

Hint: Use the TOP clause and join CustOrder to CustOrderLine.

Paste below the **code** you wrote **and the run results** you obtained for exercise 2:

SELECT TOP 3 CO.CustomerID,

SUM(COL.OrderQuantity) AS NumOfPartsOrdered

FROM CustOrderLine COL

JOIN CustOrder CO

ON COL.OrderID = CO.OrderID

GROUP BY CO.CustomerId

ORDER BY NumOfPartsOrdered DESC;

CustomerID NumOfPartsOrdered

C-300010 282

C-300001 269

C-300020 262

------------------------------------------------

3. Write a SELECT statement that returns three columns consisting of the CustomerID, the total number of orders the customer has placed, and the undiscounted total dollar value of the orders the customer has placed. Name these columns:

CustomerID

NumberOfOrders

TotalDollarValueOfOrders

Sort the result set by number of orders in either DESC or ASC sequence.

Paste below the **code** you wrote and **type the number of rows returned** for exercise 3:

SELECT CustomerID, COUNT(\*) AS NumberOfOrders,

SUM(COL.OrderQuantity \* COL.UnitPrice) AS TotalDollarValueOfOrders

FROM CustOrder C

JOIN CustOrderLine COL

ON C.OrderID = COL.OrderID

GROUP BY CustomerID

ORDER BY NumberOfOrders DESC;

Rows Returned: 227

------------------------------------------------

4. Modify the solution to exercise 3 to filter for CustomerID’s that start with C-30001 (only include rows where the CustomerID starts with C-30001)

Hint: add a where clause

Paste below the **code** you wrote **and the run results** you obtained for exercise 4:

SELECT CustomerID, COUNT(\*) AS NumberOfOrders,

SUM(COL.OrderQuantity \* COL.UnitPrice) AS TotalDollarValueOfOrders

FROM CustOrder C

JOIN CustOrderLine COL

ON C.OrderID = COL.OrderID

WHERE CustomerID LIKE 'C-30001%'

GROUP BY CustomerID

ORDER BY NumberOfOrders DESC;

Rows Returned: 10

CustomerID NumberOfOrders TotalDollarValueOfOrders

C-300019 22 47632.21

C-300011 21 79766.48

C-300017 21 63018.64

C-300015 20 36821.12

C-300012 17 36366.20

C-300013 17 29425.01

C-300010 15 34250.22

C-300016 14 21480.14

C-300018 11 55036.66

C-300014 9 12815.81

------------------------------------------------

5. Write a SELECT statement that returns three columns consisting of the CustomerID, the total number of orders the customer has placed, and the average undiscounted dollar value of the orders the customer has placed. Name these columns:

CustomerID

NumberOfOrders

AvgDollarValueOfOrders

Sort the result set so the Customer with the least orders appears first.

hints:

1. Join the CustOrderLine table to the CustOrder table. Make sure to count Orders not orderline items. The following example does NOT produce the correct answer for exercise 5.

SELECT CustomerID, COUNT(CO.OrderID)As NumberofLineItems,

AVG(UnitPrice \* OrderQuantity) AS AVGDollarValueOfLineItems

FROM CustOrderLine AS COL Join CustOrder AS CO

ON COL.OrderID = CO.OrderID

GROUP BY CustomerID

ORDER BY NumberofLineITems DESC

1. The above query can be changed to count Orders by replacing the COUNT(Co.OrderID) with COUNT(Distinct Co.OrderID).
2. The above query can be changed to calculate AvgDollarValueOfOrders by replacing the AVG(UnitPrice \* OrderQuantity) with SUM(UnitPrice \* OrderQuantity)/COUNT(DISTINCT CustOrderLine.OrderID). The problem with using AVG as shown above for problem five is that it bases its calculation on the number of lineitems not the number of orders.

Paste below the **code** you wrote and **type the number of rows returned** for exercise 5:

SELECT CustomerID, COUNT(DISTINCT CO.OrderID) AS NumberofOrders,

SUM(UnitPrice \* OrderQuantity)/COUNT(DISTINCT COL.OrderID) AS AvgDollarValueOfOrders

FROM CustOrderLine COL

JOIN CustOrder CO

ON COL.OrderID = CO.OrderID

GROUP BY CustomerID

ORDER BY NumberofOrders DESC;

Rows returned: 227

------------------------------------------------

6. Modify the solution to exercise 5 to only include groups in the final result set that have a NumberOfOrders value of 5 or greater.

Hint: add a HAVING clause

Paste below the **code** you wrote **and the run results** you obtained for exercise 6:

SELECT CustomerID, COUNT(DISTINCT CO.OrderID) AS NumberofOrders,

SUM(UnitPrice \* OrderQuantity)/COUNT(DISTINCT COL.OrderID) AS AvgDollarValueOfOrders

FROM CustOrderLine COL

JOIN CustOrder CO

ON COL.OrderID = CO.OrderID

GROUP BY CustomerID

HAVING COUNT(DISTINCT CO.OrderID) >= 5

ORDER BY NumberofOrders DESC;

CustomerID NumberofOrders AvgDollarValueOfOrders

C-300004 7 4378.464285

C-300005 7 5218.965714

C-300006 7 8554.612857

C-300001 7 12510.610000

C-300002 7 13071.544285

I-300024 7 885.260000

I-300087 6 1039.783333

C-300023 6 16779.000000

C-300028 6 6477.396666

C-300065 6 8921.680000

C-300067 6 8887.688333

C-300011 6 13294.413333

C-300007 6 11279.116666

C-300015 6 6136.853333

C-300016 6 3580.023333

C-300017 6 10503.106666

C-300019 6 7938.701666

C-300020 6 7192.733333

I-300128 6 732.968333

I-300015 6 1033.425000

I-300022 5 585.406000

I-300132 5 2631.618000

I-300139 5 941.106000

I-300141 5 1020.706000

I-300148 5 486.560000

I-300149 5 219.512000

I-300155 5 951.388000

I-300156 5 538.850000

I-300157 5 754.188000

C-300021 5 16132.270000

C-300022 5 2069.756000

C-300018 5 11007.332000

C-300009 5 12163.958000

C-300010 5 6850.044000

C-300012 5 7273.240000

C-300013 5 5885.002000

C-300003 5 1843.656000

I-300003 5 1338.238000

I-300004 5 484.482000

I-300008 5 965.946000

I-300010 5 761.434000

I-300013 5 802.010000

C-300033 5 5145.790000

C-300035 5 8782.430000

C-300024 5 3360.226000

I-300093 5 600.228000

I-300101 5 514.232000

I-300102 5 2683.134000

I-300105 5 555.028000

I-300120 5 821.978000

I-300032 5 1015.298000

I-300044 5 912.862000

I-300049 5 1206.710000

I-300065 5 1171.126000

------------------------------------------------

7. Write a SELECT statement that returns two columns from the PackingSlip table: EmployeeID and PackageCount, where PackageCount is a count of the rows on the packing slip table associated with the EmployeeID. Sort the result set by EmployeeID in Descending sequence.

Paste below the **code** you wrote **and the run results** you obtained for exercise 7:

SELECT EmployeeID, COUNT(\*) AS PackageCount

FROM PackingSlip

GROUP BY EmployeeID

ORDER BY EmployeeID DESC;

EmployeeID PackageCount

101166 128

101154 401

101097 681

101088 1244

101045 563

100967 198

100204 109

100120 7

100101 165

------------------------------------------------

8. Modify the solution to exercise 7 to filter to only include rows from the source table that have a ShippedDate that falls in the following inclusive range:

'2016-07-06' to '2016-07-08'

Paste below the **code** you wrote **and the run results** you obtained for exercise 8:

SELECT EmployeeID, COUNT(\*) AS PackageCount

FROM PackingSlip

WHERE ShippedDate BETWEEN '2016-07-06' AND '2016-07-08'

GROUP BY EmployeeID

ORDER BY EmployeeID DESC;

EmployeeID PackageCount

101166 1

101154 4

101097 12

101088 27

101045 44

100120 4

100101 1

------------------------------------------------

9. Modify the solution to exercise 8 to only include groups in the final result set that have a PackageCount value of 10 or greater.

Paste below the **code** you wrote **and the run results** you obtained for exercise 9:

SELECT EmployeeID, COUNT(\*) AS PackageCount

FROM PackingSlip

WHERE ShippedDate BETWEEN '2016-07-06' AND '2016-07-08'

GROUP BY EmployeeID

HAVING COUNT(\*) >=10

ORDER BY EmployeeID DESC;

EmployeeID PackageCount

101097 12

101088 27

101045 44

------------------------------------------------

10. Write a SELECT statement that returns two columns from the PackingSlip table: ShippedDate and PackageCount, where PackageCount is a count of the rows on the packing slip table associated with the ShippedDate. Include a row that gives the total number of packages packed.

Hint: make use of WITH ROLLUP

Paste below the **code** you wrote and **type the number of rows returned** for exercise 10:

SELECT ShippedDate, COUNT(\*) AS PackageCount

FROM PackingSlip

GROUP BY ROLLUP(ShippedDate)

ORDER BY ShippedDate ASC;

Rows Returned: 194

------------------------------------------------