/\* Orysya Stus - Assignment 1 \*/

USE JProCo

/\* Lab 3.1 Working with NULLS; Skill Checks 1, 2 and 3 \*/

/\*

Skill Check 1: The Employee table in JProCo has a field called ManagerID. Write a query to show all Employees who don’t have a ManagerID.

\*/

SELECT \*

FROM Employee

WHERE ManagerID IS NULL

/\*

Skill Check 2: Find the two records in the Customer table where there is an existing company name, for each record, as shown in the figure below.

\*/

SELECT \*

FROM Customer

WHERE CompanyName IS NOT NULL

/\*

Skill Check 3: Using your result set from Skill Check 2, write a statement to update the CustomerType field of the Customer table to “Consumer”, for each record where the company name is not NULL.

\*/

UPDATE Customer

SET CustomerType='Company'

WHERE CustomerType='Business'

SELECT \*

FROM Customer

WHERE CompanyName IS NOT NULL

/\* Lab 3.2 Expression Fields; Skill Checks 1, 2, 3 and 4 \*/

/\*

Skill Check 1: Using the CurrentProducts table of JProCo, make another field to express the price in Canadian currency called CDN $ that is 1.1 times the stated RetailPrice in JProCo’s CurrentProducts table. Then create two additional fields named Aussie $ ( 1.4 times the RetailPrice) and Euro (which is .82 times the RetailPrice). When you’re done your result should resemble the figure below

\*/

SELECT

ProductName

, RetailPrice

, RetailPrice\*1.1 AS cdn$

, RetailPrice\*1.4 AS Aussie$

, RetailPrice\*.82 AS Euro

FROM CurrentProducts

/\*

Skill Check 2: In JProCo find the 773 records in your Customer table where the CustomerType is Consumer. Show the CustomerID, CustomerType field and the FullName expression field. Your results should be sorted by the FullName field (Z-A). When you’re done your result should resemble the figure below.

\*/

SELECT

CustomerID

, CustomerType

, FirstName + ' ' + LastName AS FullName

FROM Customer

WHERE CustomerType= 'Consumer'

ORDER BY FullName DESC

/\*

Skill Check 3: Join the SalesInvoiceDetail table to the CurrentProducts table. Show the ProductID, ProductName and RetailPrice from the CurrentProducts table. Show Quantity from the SalesInvoiceDetail table. Create an expression field called SubTotal which multiplies RetailPrice by Quantity. Your result should look like the figure below.

\*/

SELECT

CP.ProductID

, CP.ProductName

, CP.RetailPrice

, SID.Quantity

, CP.RetailPrice\*SID.Quantity AS SubTotal

FROM SalesInvoiceDetail AS SID

LEFT OUTER JOIN CurrentProducts AS CP

ON SID.ProductID = CP.ProductID

/\*

Skill Check 4: Modify your query from Skill Check 3. Using the Round function, show Retail Price and the SubTotal expression field rounded to the nearest penny.

\*/

SELECT

CP.ProductID

, CP.ProductName

, ROUND(CP.RetailPrice, 2)

, SID.Quantity

, ROUND(CP.RetailPrice\*SID.Quantity, 2) AS SubTotal

FROM SalesInvoiceDetail AS SID

LEFT OUTER JOIN CurrentProducts AS CP

ON SID.ProductID = CP.ProductID

/\* Lab 3.3 Identity Fields; Skill Check 1 \*/

/\*

Skill Check 1: The MgmtTraining table in JProCo has ClassID as an identity field. Previously, this table had many records deleted from it. We want to insert a value of ‘Empowering Others’ in the ClassName field with a ClassID of 4. If we run a simple INSERT statement, the identity counter is already past the number 4. We must set the table’s property for inserting values to allow manually inserting all fields for this record. The ApprovedDate field should be set using the CURRENT\_TIMESTAMP property. When done, the results should resemble those shown in Figure below.

\*/

SET IDENTITY\_INSERT MgmtTraining ON

INSERT INTO MgmtTraining (ClassID, ClassName, ClassDurationHours, ApprovedDate)

VALUES(4, 'Empowering Others', 18, CURRENT\_TIMESTAMP)

SET IDENTITY\_INSERT MgmtTraining OFF

SELECT \*

FROM MgmtTraining

/\* Lab 4.1 Using Group By; Skill Checks 1, 2, 3 and 4 \*/

/\*

Skill Check 1: Query the Employee table of JProCo to see how many people work for each ManagerID . Select the ManagerID and Count the EmpID field. Alias the field as EmpIDCount.

\*/

SELECT

ManagerID

, Count(EmpID) AS EmpIDCount

FROM Employee

GROUP BY ManagerID

/\*

Skill Check 2: Perform a grouping query on the Customer table to get a count of how many consumers versus Business customers you have. Alias the field as CustomerCount. Group on the CustomerType field.

\*/

SELECT

CustomerType

, COUNT(CustomerType) AS CustomerCount

FROM Customer

GROUP BY CustomerType

/\*

Skill Check 3: Get a list of all Customers and how many Invoice orders each one has placed. You will need to join the Customer and SalesInvoice tables. Alias the aggregated field as InvoiceIDCount. If a Customer has not ordered yet, then you should still see their name with a zero next to it.

\*/

SELECT

C.CustomerID

, C.FirstName

, C.LastName

, COUNT(InvoiceID) AS InvoiceIDCount

FROM Customer AS C

LEFT OUTER JOIN SalesInvoice AS SI

ON C.CustomerID = SI.CustomerID

GROUP BY C.CustomerID, C.FirstName, C.LastName

/\*

Skill Check 4: Make a slight modification to Skill Check 3 so that only Customers who have placed at least one order appear in the query (Hint: change the type of join). Notice CustomerID 5 does not appear in this result. (Figure below).

\*/

SELECT

C.CustomerID

, C.FirstName

, C.LastName

, COUNT(InvoiceID) AS InvoiceIDCount

FROM Customer AS C

LEFT OUTER JOIN SalesInvoice AS SI

ON C.CustomerID = SI.CustomerID

GROUP BY C.CustomerID, C.FirstName, C.LastName

HAVING COUNT(InvoiceID) > 0

/\* Lab 4.2 Filtering Aggregated Results; Skill Checks 1 and 2 \*/

/\*

Skill Check 1: Using the SalesInvoice table, write a query which groups on CustomerID and counts the number of orders (also called invoices) each customer has made. Return only the records where the CustomerID has ordered more than 7 times. Show the aggregated field as OrderCount. When done, the results should resemble the figure below.

\*/

SELECT

CustomerID

, COUNT(CustomerID) AS OrderCount

FROM SalesInvoice

GROUP BY CustomerID

HAVING COUNT(CustomerID) > 7

/\*

Skill Check 2: Query the SalesInvoiceDetail table and show just the ProductID and InvoiceID fields. Change the query to group on ProductID and count the InvoiceID field. Return only the records where the ProductID has been ordered more than 200 times.

\*/

SELECT

ProductID

, COUNT(InvoiceID) AS InvoiceCt

FROM SalesInvoiceDetail

GROUP BY ProductID

HAVING COUNT(InvoiceID) > 200

/\* Lab 4.3 Aggregation in Stored Procedures, Skill Check 1 \*/

/\*

Skill Check 1: Using what you’ve learned, create a stored procedure called GetCategoriesByProductCount. This will join the CurrentProducts and the SalesInvoiceDetail tables and show the total number of orders for each Category.

\*/

CREATE PROC GetCategoriesByProductCount AS

SELECT

CP.Category

, COUNT(SID.Quantity) AS ProductCt

FROM CurrentProducts AS CP

JOIN SalesInvoiceDetail AS SID

ON CP.ProductID = SID.ProductID

GROUP BY CP.Category

EXEC GetCategoriesByProductCount

/\* Volume 2: Lab 5.1 Finding Duplicates, Skill Checks 1 and 2 \*/

/\*

Skill Check 1: Go to the JProCo database and count all employees having multiple grants listed in the Grant table. Your result should resemble the figure below.

\*/

SELECT

E.FirstName

, E.LastName

, COUNT(G.EmpID)

FROM [Grant] AS G

JOIN Employee AS E

ON G.EmpID = E.EmpID

GROUP BY E.FirstName, E.LastName

HAVING COUNT(G.EmpID) > 1

/\*

Skill Check 2: Query the StateList table to find any duplicate records . List all duplicated StateID and StateName values you find. Title your aggregated field IDCount.

\*/

SELECT

StateID

, StateName

, COUNT(StateID) AS IDCount

FROM StateList

GROUP BY StateID, StateName

HAVING COUNT(StateID) > 1

/\* Lab 5.2 The Over the Clause, Skill Checks 1, 2, 3 and 4 \*/

/\*

Skill Check 1: Use the JProCo database and query the CurrentProducts table for the fields ProductName, RetailPrice and Category. Create an expression field that combines AVG() with an OVER() clause and call it AvgPrice. When you are done, your result should resemble the figure you see below.

\*/

SELECT

ProductName

, RetailPrice

, Category

, AVG(RetailPrice) OVER() AS AvgPrice

FROM CurrentProducts

/\*

Skill Check 2: Take the query from Skill Check 1 and add another expression field called AvgCatPrice that shows the average price for the Category for any given product. When you are done, your result will resemble the figure below.

\*/

SELECT

ProductName

, RetailPrice

, Category

, AVG(RetailPrice) OVER() AS AvgPrice

, AVG(RetailPrice) OVER(PARTITION BY Category) AS AvgCatPrice

FROM CurrentProducts

/\*

Skill Check 3: Use the JProCo database and query the CurrentProducts table. Show each distinct category and calculate the percentage (with decimals) of products for each Category. Since we have more LongTerm -Stay products, that category will represent the highest percentage of the total (Figure below).

\*/

SELECT

DISTINCT Category

, COUNT(\*) OVER(PARTITION BY Category) \* 100.0 / COUNT(\*) OVER() AS PctCategory

FROM CurrentProducts

/\*

Skill Check 4: Join the Location, Employee and Grant tables and display FirstName, LastName, GrantName, City and Amount. Add an expression field called CityTotal that compares each grant to the total amount in the same City (See the expected result in Figure below).

\*/

SELECT

emp.FirstName

, emp.LastName

, gr.GrantName

, loc.City

, gr.Amount

, SUM(Amount) OVER(PARTITION BY City) AS CityTotal

FROM Employee AS emp

INNER JOIN Location AS Loc

ON Emp.locationID = Loc.locationID

INNER JOIN [Grant] AS gr

ON emp.EmpID = gr.EmpID