Data Analysis Co-op Project (202205)

SQL Server Database Fundamentals System Requirements

v1.0 2022-06-16

Supported by DataTech Consulting Services Inc

[http://www.DataTechservice.com](http://www.datatechservice.com/)

|  |  |
| --- | --- |
| Name | Tasks |
| Amy | team lead：organize weekly meeting and keep meeting record  2.4.  [More Queries – 2, Employees-driven (C2 - C5)](https://docs.google.com/document/d/1fyR0vqXcuvxHgfYB_Kbdcut8qIl097Gn/edit" \l "heading=h.1t3h5sf)  2.6 More Queries – 4, Categories-driven (C2 - C5)  2.11.  [Add Quarterly-Based Sales Performance Tracking Report](https://docs.google.com/document/d/1fyR0vqXcuvxHgfYB_Kbdcut8qIl097Gn/edit" \l "heading=h.35nkun2)  2.16.  [Reporting with Percentile (2) - Employees-driven](https://docs.google.com/document/d/1fyR0vqXcuvxHgfYB_Kbdcut8qIl097Gn/edit" \l "heading=h.3whwml4)  2.18 Reporting with Percentile (4) - Categories-driven |
| Emma | 2.3 More Queries – 1, Customers-driven (C2 - C5)  2.5  More Queries – 3, Products-driven (C2 - C5)  2.10 Add Year-Based Sales Performance Tracking Reports (C5 - C8)  2.14 Key Performance Indicator (C7, C8)  2.15 Reporting with Percentile (1) - Customers-driven  2.17 Reporting with Percentile (3) - Products-driven |
| Sarah | 2.2 Data age  2.7 More Queries – 5, Shipping-driven (C2 - C5)  2.8 More Queries – 6, Supplier-driven (C2 - C5)  2.12 Add Monthly-Based Sales Performance Tracking Reports  2.19 Reporting with Percentile (5) - Shipping-driven |
| GZ | 2.1 Data Quantity (C1 - C3)  2.9 More Queries – 7, Supplier Reputation analysis (C2 - C5)  2.13 Add Daily-Based Sales Performance Tracking Reports  2.20 Reporting with Percentile (6) - Supplier-driven  2.21 Reporting with Percentile (7) - Supplier Reputation-driven |

Table of Contents

[Change History 3](#_bookmark0)

1. [Project Overview 4](#_bookmark1)
2. [Enhancement Summary 6](#_bookmark2)
   1. [Data Quantity (C1 - C3) 6](#_bookmark3)
   2. [Data Age (C1) 6](#_bookmark4)
   3. [More Queries – 1, Customers-driven (C2 - C5) 6](#_bookmark5)
   4. [More Queries – 2, Employees-driven (C2 - C5) 7](#_bookmark6)
   5. [More Queries – 3, Products-driven (C2 - C5) 7](#_bookmark7)
   6. [More Queries – 4, Categories-driven (C2 - C5) 8](#_bookmark8)
   7. [More Queries – 5, Shipping-driven (C2 - C5) 8](#_bookmark9)
   8. [More Queries – 6, Supplier-driven (C2 - C5) 8](#_bookmark10)
   9. [More Queries – 7, Supplier Reputation analysis (C2 - C5) 9](#_bookmark11)
   10. [Add Year-Based Sales Performance Tracking Reports (C5 - C8) 9](#_bookmark12)
   11. [Add Quarterly-Based Sales Performance Tracking Report 10](#_bookmark13)
   12. [Add Monthly-Based Sales Performance Tracking Reports 10](#_bookmark14)
   13. [Add Daily-Based Sales Performance Tracking Reports 10](#_bookmark15)
   14. [Key Performance Indicator (C7, C8) 11](#_bookmark16)
   15. [Reporting with Percentile (1) - Customers-driven 11](#_bookmark17)
   16. [Reporting with Percentile (2) - Employees-driven 11](#_bookmark18)
   17. [Reporting with Percentile (3) - Products-driven 11](#_bookmark19)
   18. [Reporting with Percentile (4) - Categories-driven 12](#_bookmark20)
   19. [Reporting with Percentile (5) - Shipping-driven 12](#_bookmark21)
   20. [Reporting with Percentile (6) - Supplier-driven 12](#_bookmark22)
   21. [Reporting with Percentile (7) - Supplier Reputation-driven 12](#_bookmark23)
3. [Offerings & Constraints 13](#_bookmark24)

# Change History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Reason for Change |
| 2022-06-08 | 0.1 | Jay L. | Initial Draft |
| 2022-06-12 | 0.2 | Jay L. | Updated 1, 2.1, 2.2, document reformatting |
| 2022-06-15 | 0.3 | Jay L. | Finalizing timeline |
| 2022-06-16 | 1.0 | Jay L. | Final – v1.0 |
|  |  |  |  |

# Project Overview

This project is designed to perform the upgrade of some major technologies for providing the technical support to the external users of DataTech Consulting Services, to improve their current analysis system to assist boosting their business. This document describes the detailed tasks required for this technology enhancement.

This requirement document is to request working on the following technology enhancement or migration, to modernize, optimize, and consolidate the existing version of the analysis platform based on SQL Server Database Fundamentals. With the project completion, the enhanced features running on the upgraded system design will offer a better designed, documented, improved, and standardized environment, which supports a new look of the technologies the system runs on, and offer a nicely-looking infrastructure of the new version of analysis.

Project Timeline

Week 1: 2022-05-29

Team build-up, Groups setup – Project started Week 2: 2022-06-05

Finalizing Groups and assigning Group Leads, System Requirements drafting - 5% completed

Week 3: 2022-06-12

System Requirements completed,

Coop environment setup and ready for use - 10% completed Week 4: 2022-06-19

System Requirements Study (1) - 15% completed Week 5: 2022-06-26

System Requirements Study (2) System Design (1) - 20% completed

Week 6: 2022-07-03

System Requirements Study (3) - completed

System Design (2) - 25% completed

Week 7: 2022-07-10

System Design (3) - completed Development started (1) - 30% completed

Week 8: 2022-07-17

Development in progress (2) - 40% completed Week 9: 2022-07-24

Development in progress (3) - 50% completed Week 10: 2022-07-31

Development in progress (4) - 60% completed Week 11: 2022-08-07

Development in progress (5) - 70% completed Week 12: 2022-08-14

Development (6) - completed Testing started (1) - 80% completed

Week 13: 2022-08-21

Testing started (2)

Final review and documentation started (1) - 90% completed Week 14: 2022-08-28

Testing (3) – completed

Final review and documentation started (2) - completed Work wrap-up and package submission, 100% completed

Final Meeting: 2022-09-04

Project Closing

# Enhancement Summary

The following list contains all required tasks, portions, technologies, and framework to support and offer a new look of the corresponding analysis.

## Data Quantity (C1 - C3)

Currently the process only uses a small set of sample data with 1.2 million rows. In the new version after this upgrade, we will offer ***50,000,000 (50M)*** level of rows as the dataset to show the 360-degree of data manipulation and processing. A simulation of the data to make up such level of amount of data would be considered.

For simulating the data as much real as possible, we need to give users for the Orders and Order-Details enlargement by matching Order to Order-Details in 1 to 10 relationships in average. That says, that if the Order-Details take 50M rows of data, roughly we should have 5M (million) orders generated in the Orders table.

As there is no such data yet, so to make this functionality developed and

tested, a simulation of the data to “make up” such level of amount of data would be considered, for testing the performance and enhancement purposes.

## Data Age (C1)

The new version will show the most recent 20 years of data, but counting back from the current year, back to 20 years from now, meaning the newly designed analysis should show the most up-to-date data. During the simulation as enlarging the data amount, this should be also handled. The project needs to show managing data as for ***May 31, 2022***.

## More Queries – 1, Customers-driven (C2 - C5)

Invoke more queries by combining more cross-referencing among more different tables of data.

Customers-driven statistical queries:

* Show Customers’ Shopping (Total purchase amount spent) based on the Customers’ Country, City
* Show Customers’ Shopping (Total purchase amount spent) based on the Customers’ Postal Code
* Show Customers’ Shopping (Total purchase amount spent) based on the Customers’ Area Code of their phone number
* Consider build a stored procedure to present the data, with the parameters passed in to drive how the report results

## More Queries – 2, Employees-driven (C2 - C5)

Invoke more queries by combining more cross-referencing among more different tables of data.

Employees-driven statistical queries:

* Show Employees’ Sales performance (Total amount sold) based on the Employees’ Country, City

select c.Country, sum(b.Salesamount) as SalesAmount

from Orders..NewOrders as a inner join (select OrderID, sum(UnitPrice \* Quantity \* (1-Discount) ) as Salesamount

from Orders..NewOrderDetails group by OrderID) as b

on a.OrderID = b.OrderID

inner join Orders..Employees as c

on a.EmployeeID = c.EmployeeID

group by c.Country

order by c.Country

select c.City, sum(b.Salesamount) as SalesAmount

from Orders..NewOrders as a inner join (select OrderID, sum(UnitPrice \* Quantity \* (1-Discount) ) as Salesamount

from Orders..NewOrderDetails group by OrderID) as b

on a.OrderID = b.OrderID

inner join Orders..Employees as c

on a.EmployeeID = c.EmployeeID

group by c.City

order by c.City

* Show Employees’ Sales performance (Total amount sold) based on the Employees’ Postal Code

select c.PostalCode, sum(b.Salesamount) as SalesAmount

from Orders..NewOrders as a inner join (select OrderID, sum(UnitPrice \* Quantity \* (1-Discount) ) as Salesamount

from Orders..NewOrderDetails group by OrderID) as b

on a.OrderID = b.OrderID

inner join Orders..Employees as c

on a.EmployeeID = c.EmployeeID

group by c.PostalCode

order by c.PostalCode

* Show Employees’ Sales performance (Total amount sold) based on the Employees’ Area Code of their phone number

--GROUP BY LEFT(HomePhone, 4)

select (LEFT(c.HomePhone, 4)) as areacode, sum(b.Salesamount) as SalesAmount

from Orders..NewOrders as a inner join (select OrderID, sum(UnitPrice \* Quantity \* (1-Discount) ) as Salesamount

from Orders..NewOrderDetails group by OrderID) as b

on a.OrderID = b.OrderID

inner join Orders..Employees as c

on a.EmployeeID = c.EmployeeID

group by LEFT(c.HomePhone, 4)

order by LEFT(c.HomePhone, 4)

* Show Employees’ Sales performance (Total amount sold) based on their Report-To Manager

select c.ReportsTo, sum(b.Salesamount) as SalesAmount

from Orders..NewOrders as a inner join (select OrderID, sum(UnitPrice \* Quantity \* (1-Discount) ) as Salesamount

from Orders..NewOrderDetails group by OrderID) as b

on a.OrderID = b.OrderID

inner join Orders..Employees as c

on a.EmployeeID = c.EmployeeID

group by c.ReportsTo

order by c.ReportsTo

* Consider build a stored procedure to present the data, with the parameters passed in to drive how the report results

​​Solution:

Employees’ Sales performance (Total amount sold) would involve three tables, which are Orders, Order Details and Employees - need to be joined together.

When select data, use [UnitPrice\*Quantity\*(1-Discount)] to get the ‘Total amount sold’.

Using group by to calculate  ‘Total amount sold’ according to Employees’ country, city, postal code and area code(GROUP BY LEFT(HomePhone, 4)) of their phone number.

Create stored procedure with parameters such as country, city, postal code and area code to drive the report results.

--Create Procedure

declare @sqle nvarchar(max) ,@prae varchar(30)

set @prae = 'c.PostalCode'

set @sqle = 'select ' + @prae + ', sum(b.Salesamount) as SalesAmount

from EnlargedOrders as a inner join (select OrderID, sum(UnitPrice \* Quantity \* (1-Discount) ) as Salesamount from EnlargedOrderDetail group by OrderID) as b

on a.OrderID = b.OrderID

inner join Employees as c

on a.EmployeeID = c.EmployeeID

GROUP BY ' + @prae

exec (@sqle)

## More Queries – 3, Products-driven (C2 - C5)

Invoke more queries by combining more cross-referencing among more different tables of data.

Products-driven statistical queries:

* Show Products’ Sales performance (Total amount sold) based on the Product Suppliers’ Country, City
* Show Products’ Sales performance (Total amount sold) based on the Product Suppliers’ Postal Code
* Show Products’ Sales performance (Total amount sold) based on the Product Suppliers’ Area Code of their phone number
* Consider build a stored procedure to present the data, with the parameters passed in to drive how the report results

## More Queries – 4, Categories-driven (C2 - C5)

Invoke more queries by combining more cross-referencing among more different tables of data.

Categories-driven statistical queries:

* Show Products’ Sales performance (Total amount sold) based on the Products’ Categories

select c.CategoryID, sum(b.Salesamount) as SalesAmount

from Orders..Products as a inner join (select ProductID, sum(UnitPrice \* Quantity \* (1-Discount) ) as Salesamount

from Orders..NewOrderDetails group by ProductID) as b

on a.ProductID = b.ProductID

inner join Orders..Categories as c

on a.CategoryID = c.CategoryID

group by c.CategoryID

order by c.CategoryID

* Consider build a stored procedure to present the data, with the parameters passed in to drive how the report results

Solution:

In order to get the ‘Total Amount’, Three tables which are Order Details, Categories, Products need to be joined together.

Using formula [UnitPrice\*Quantity\*(1-Discount)] to get the ‘Total Amount’.

Using groupby CategoryID, ProductID in the select statement to get Products’s sales performance based on products’ categories.

Creating Stored Procedure with parameters. Show the Products’ sales performance by passing in categoryID.

declare @sqlf nvarchar(max) ,@praf varchar(30)

set @praf = 'c.CategoryID'

set @sqlf = 'select ' + @praf + ', sum(b.Salesamount) as SalesAmount

from Orders..Products as a inner join (select ProductID, sum(UnitPrice \* Quantity \* (1-Discount) ) as Salesamount

from Orders..NewOrderDetails group by ProductID) as b

on a.ProductID = b.ProductID

inner join Orders..Categories as c

on a.CategoryID = c.CategoryID

GROUP BY ' + @praf

exec (@sqlf)

Solution: Calculate total amount sold for each product and then use a parameter that uses group by to display based on the product category

## More Queries – 5, Shipping-driven (C2 - C5)

Invoke more queries by combining more cross-referencing among more different tables of data.

Shipping-driven statistical queries:

* Show Products’ Sales performance (Total amount sold) based on the Area code of the phone number of the Shipping Companies
* Consider build a stored procedure to present the data, with the parameters passed in to drive how the report results

## More Queries – 6, Supplier-driven (C2 - C5)

Invoke more queries by combining more cross-referencing among more different tables of data.

Supplier-driven statistical queries:

* Show Suppliers’ Sales performance (Total amount sold) based on the supplier’s company over the ship-to-country.
* Consider build a stored procedure to present the data, with the parameters passed in to drive how the report results

## More Queries – 7, Supplier Reputation analysis (C2 - C5)

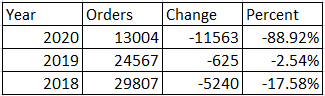
Invoke more queries by combining more cross-referencing among more different tables of data.

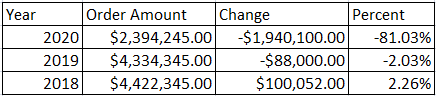
Supplier Reputation analysis queries:

* Show Suppliers’ products performance (Total products sold), list most products welcomed Suppliers, say top 100 suppliers.
* Consider build a stored procedure to present the data, with the parameters passed in to drive how the report results

## Add Year-Based Sales Performance Tracking Reports (C5 - C8)

Add two new reports to list:

* Over-year Total Orders with Comparison
* Over-year Total Order Amount



* Consider build a stored procedure to present the data, with the parameters (@years), passed in to drive how many years the report is calculated, from the current year, back to @years ago.
* The calculation should be able to auto-determine when the “last year” is.

For ex., if now is 2021-04-10, as it is the mid of the current year, so the report is to tell the sales performance as end of 2020. Because 2021 is not passed yet.

## Add Quarterly-Based Sales Performance Tracking Report

Similarly as the task above, this task is to track sales performance in year- quarter based report. You may think of the first column is “Year-Quarter”, and the list will be changed to Year-Quarter, as 2021-Q1, 2020-Q4, 2020-Q3, …

Note, similarly as year-based, your calculation should be able to auto-detect when the last year-quarter is. If now is 2021-04-10, then the last quarter is 2021-Q1; when moving into 2021-07-01, 2021-Q2 becomes the last one.

Solution:

Quarterly-Based Sales Performance Tracking Report

In order to present a quarterly-based sales performance tracking report, joining Orders and Order Details two tables.

‘Sales Performance/Orders Amount’ can be calculated by Sum (Unit Price\*Quantity\*(1-discount)). Each quarter sales performance can be generated by the table expression.

Using conditional clauses to return the user selected different Year-Quarter time frames.

Building a stored procedure with parameters to get the targeted dataset.

Solution:

Create a procedure and declare two parameters for start and end date.

Build up the column to show the “Year-Quarter” and get the Order number group by  “Year-Quarter”.

Use T-SQL function to calculate the order number change for different years. And calculate the percentage. The code would be:

Use Orders

go

--Orders quarterly based

if object\_id('SP\_year\_quarter\_Total\_Orders') is not null

drop  proc SP\_year\_quarter\_Total\_Orders

go

CREATE PROCEDURE SP\_year\_quarter\_Total\_Orders

 (@start\_year\_quarter varchar(100) ,@ending\_year\_quarter  varchar(100))

as

Begin

With part1 as

(select concat(year(orderdate),'-Q',

 datepart(QUARTER,orderdate))

[Year-Quarter], orderid

from NewOrders),

part2 as

(select [Year-Quarter], count(distinct orderid) Orders

from part1 group by [Year-Quarter])

select [Year-Quarter],Orders,lag(Orders)over(order by [Year-Quarter] desc) Temp,

(Orders-lag(Orders)over(order by [Year-Quarter] desc)) Change,

format((Orders-lag(Orders)over(order by [Year-Quarter] desc) \*1.0)/

lag(Orders)over(order by [Year-Quarter] desc),'P')  [Percent]

from part2

 where [Year-Quarter]  between  @start\_year\_quarter and @ending\_year\_quarter

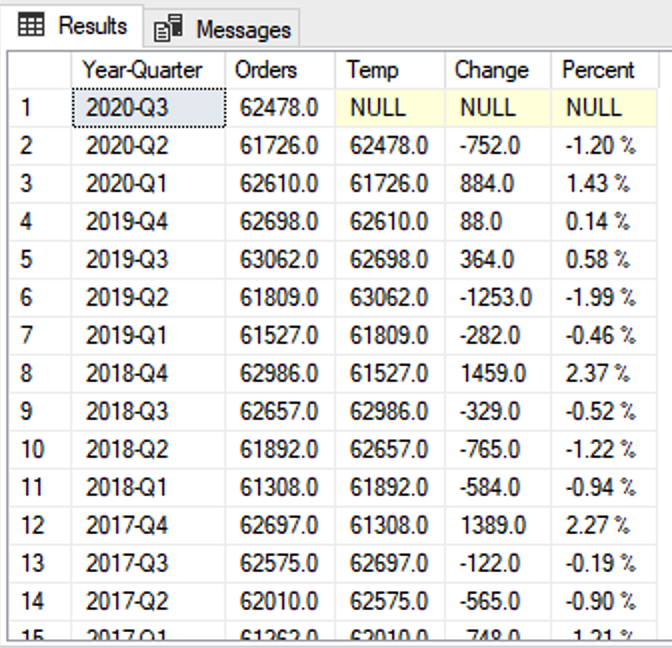
order by [Year-Quarter] desc

End

go

exec SP\_year\_quarter\_Total\_Orders @start\_year\_quarter='2015-Q1',

 @ending\_year\_quarter ='2016-Q3'



For calculate the Order Amount, Enlarged Order and Enlarged Order Details should be joined together. Other parts would be very similar. The code would be:

--Orders amount quarterly based

if object\_id('SP\_year\_quarter\_Sales\_Amount') is not null

Drop proc SP\_year\_quarter\_Sales\_Amount

go

CREATE PROCEDURE SP\_year\_quarter\_Sales\_Amount

(@start\_year\_quarter varchar(100) ,@ending\_year\_quarter varchar(100))

as

Begin

with part3 as

(select concat(year(orderdate),'-Q',

datepart(QUARTER,orderdate)) [Year-Quarter],

sum(Quantity \* UnitPrice \*(1-Discount)) [Order Sales Amount]

from NewOrders a

inner join NewOrderDetails b

on a.OrderID = b.OrderID

group by concat(year(orderdate), '-Q', datepart(QUARTER,orderdate))),

part4 as

(select [Year-Quarter],[Order Sales Amount],

lag([Order Sales Amount])over(order by [Year-Quarter]) lag\_yq\_OrderSalesAmount

from part3)

select [Year-Quarter],format([Order Sales Amount],'C', 'en-us') Temp,

format([Order Sales Amount]-lag\_yq\_OrderSalesAmount,'C', 'en-us') Change

,format(([Order Sales Amount]-lag\_yq\_OrderSalesAmount)/lag\_yq\_OrderSalesAmount,'P') [Percent]

from part4

where [Year-Quarter] between @start\_year\_quarter and @ending\_year\_quarter

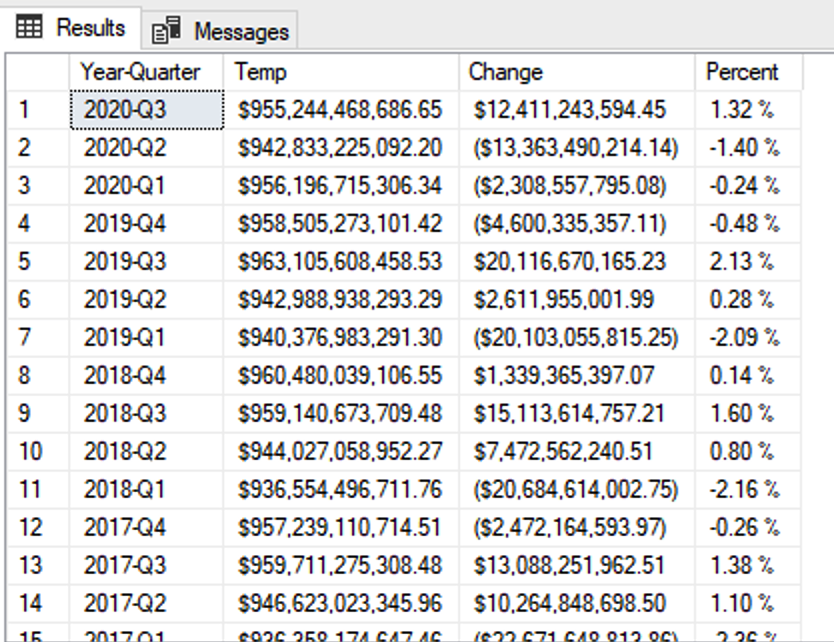
order by [Year-Quarter] desc

End

Go

exec SP\_year\_quarter\_Sales\_Amount @start\_year\_quarter='2015-Q1',

@ending\_year\_quarter ='2016-Q3'



## Add Monthly-Based Sales Performance Tracking Reports

Similarly as the task above, this task is to track sales performance in year- month based report. You may think of the first column is “Year-Month”, and the list will be changed to Year-Month, as 2021-03, 2021-02, 2021-01, …

Note, similarly as year-quarter-based, your calculation should be able to auto- detect when the last year-month is. If now is 2021-04-10, then the last month is 2021-03; when moving into 2021-05-01, 2021-04 becomes the last one.

## Add Daily-Based Sales Performance Tracking Reports

Similarly as the task above, this task is to track sales performance in daily based report. You may think of the first column is “YYYY-MM-DD”, and the list will be changed to dates, as 2021-04-10, 2021-04-09, 2021-04-08, …

The business users want the report to trace back up to 100 days before today.

## Key Performance Indicator (C7, C8)

Using SQL query or T-SQL, build a demo of Key Performance Indicator (KPI) data measurement. For ex., using last two years’ data of orders, combining Employees, to indicate annual sales increase, say total orders sold, and then show how many percent of employees having 15% or more increase, showing as Green; showing Yellow if it is between 15% and -15%, showing red if sales dropped below 15% (-15%+).

Of course, you can think of another example to complete this task, and it is totally open-minded. Additionally, you may consider building this as a report, and then create a stored procedure to get it done easily.

## Reporting with Percentile (1) - Customers-driven

For reporting task 2.3, add a new task to each of them, with a selection of number of percentile, i.e. 90 Percentile, so that the report only returns the data equal or above the given percentile.

## Reporting with Percentile (2) - Employees-driven

For reporting task 2.4, add a new task to each of them, with a selection of number of percentile, i.e. 90 Percentile, so that the report only returns the data equal or above the given percentile.

USE OrdersG2

GO

Select TOP(10) PERCENT dbo.Employees.Country, COUNT(dbo.Orders.EmployeeID) AS Total\_Amount\_Sold\_By\_Country from dbo.Employees

JOIN dbo.Orders

ON dbo.employees.EmployeeID = dbo.Orders.EmployeeID

GROUP BY dbo.Employees.Country

USE OrdersG2

GO

Select TOP(10) PERCENT dbo.Employees.City, COUNT(dbo.Orders.EmployeeID) AS Total\_Amount\_Sold\_By\_City from dbo.Employees

JOIN dbo.Orders

ON dbo.employees.EmployeeID = dbo.Orders.EmployeeID

GROUP BY dbo.Employees.City

USE OrdersG2

GO

Select TOP(10) PERCENT dbo.Employees.ReportsTo, COUNT(dbo.Orders.EmployeeID) AS Total\_Amount\_Sold\_By\_Report\_To\_Manage from dbo.Employees

JOIN dbo.Orders

ON dbo.employees.EmployeeID = dbo.Orders.EmployeeID

GROUP BY dbo.Employees.ReportsTo

USE OrdersG2

GO

Select TOP(10) PERCENT dbo.Employees.PostalCode, COUNT(dbo.Orders.EmployeeID) AS Total\_Amount\_Sold\_By\_Postal\_Code from dbo.Employees

JOIN dbo.Orders

ON dbo.employees.EmployeeID = dbo.Orders.EmployeeID

GROUP BY dbo.Employees.PostalCode

Solution: Simply take the number of data entries in total and find what 10% of the total number is equal

Sort the data by Employee sales performance

show only the first x number of data entries x being equal to 10% of the total number of entries.

## Reporting with Percentile (3) - Products-driven

For reporting task 2.5, add a new task to each of them, with a selection of number of percentile, i.e. 90 Percentile, so that the report only returns the data equal or above the given percentile.

## Reporting with Percentile (4) - Categories-driven

For reporting task 2.6, add a new task to each of them, with a selection of number of percentile, i.e. 90 Percentile, so that the report only returns the data equal or above the given percentile.

USE OrdersG2

GO

Select TOP(10) PERCENT dbo.Products.CategoryID, COUNT(dbo.Products.ProductID) AS Total\_Amount\_Sold\_By\_Product\_Category from dbo.Products

JOIN dbo.Orders

ON dbo.Products.ProductID = dbo.Orders.EmployeeID聽

GROUP BY dbo.Products.CategoryID

ORDER BY Total\_Amount\_Sold\_By\_Product\_Category

Solution: Simply take the number of data entries in total and find what 10% of the total number is equal

Sort the data by Product performance

show only the first x number of data entries x being equal to 10% of the total number of entries.

## Reporting with Percentile (5) - Shipping-driven

For reporting task 2.7, add a new task to each of them, with a selection of number of percentile, i.e. 90 Percentile, so that the report only returns the data equal or above the given percentile.

## Reporting with Percentile (6) - Supplier-driven

For reporting task 2.8, add a new task to each of them, with a selection of number of percentile, i.e. 90 Percentile, so that the report only returns the data equal or above the given percentile.

## Reporting with Percentile (7) - Supplier Reputation-driven

For reporting task 2.9, add a new task to each of them, with a selection of number of percentile, i.e. 90 Percentile, so that the report only returns the data equal or above the given percentile.

# Offerings & Constraints

* DataTech Consulting Services Inc. (DataTech) is the vendor cooperating and coordinating the projects, and as the end of the projects, DataTech offers reference check for all co-op project resources.
* All participating resources must finish all their tasks with their groups from the beginning until the end, and when and only when your name appears in the last version of the registry list at the project closing time, DataTech will then offer the reference service.
* You can ask for a short leave for vacation or any other reasons, by telling your group lead or members, and they are fine to cover you, or wait for your availability for catching up the progress designed by your group.
* With absence up to TWO weeks of unavailability when reporting your status update, you are still eligible to continue your projects, but after THREE times of missing status update, you will automatically be removed from the projects for the current round. Of course, you are still allowed to come back for the future rounds of projects.