A red and white logo with a bird

Description automatically generated

Wroclaw University of Technology

and

Telecommunications

**Data Warehouses Report 3**

Members:

Khalid Muzaffar(269553)

Mert Nuri Kodzhaaslan(260312)

**Task 1 – Simple Querying**

**1. Report without using windowed functions:**

SELECT

p.FirstName + ', ' + p.LastName AS SalesPerson,

e.BusinessEntityID AS EmployeeID,

YEAR(soh.OrderDate) AS Year,

SUM(sod.LineTotal) AS SubTotal,

COUNT(DISTINCT soh.SalesOrderID) AS NumberOfOrders

FROM

Sales.SalesOrderHeader soh

JOIN

Sales.SalesOrderDetail sod

ON

soh.SalesOrderID = sod.SalesOrderID

JOIN

Sales.SalesPerson sp

ON

soh.SalesPersonID = sp.BusinessEntityID

JOIN

HumanResources.Employee e

ON

sp.BusinessEntityID = e.BusinessEntityID

JOIN

Person.Person p

ON

e.BusinessEntityID = p.BusinessEntityID

GROUP BY

e.BusinessEntityID, p.FirstName, p.LastName, YEAR(soh.OrderDate)

**2. Report using windowed functions:**

SELECT DISTINCT

p.FirstName + ', ' + p.LastName AS SalesPerson,

e.BusinessEntityID AS EmployeeID,

YEAR(soh.OrderDate) AS Year,

SUM(sod.LineTotal) OVER (PARTITION BY e.BusinessEntityID, YEAR(soh.OrderDate)) AS SubTotal,

COUNT(soh.SalesOrderID) OVER (PARTITION BY e.BusinessEntityID, YEAR(soh.OrderDate)) AS NumberOfOrders

FROM

Sales.SalesOrderHeader soh

JOIN

Sales.SalesOrderDetail sod

ON

soh.SalesOrderID = sod.SalesOrderID

JOIN

Sales.SalesPerson sp

ON

soh.SalesPersonID = sp.BusinessEntityID

JOIN

HumanResources.Employee e

ON

sp.BusinessEntityID = e.BusinessEntityID

JOIN

Person.Person p

ON

e.BusinessEntityID = p.BusinessEntityID

**3. Report using CTE (Common Table Expression):**

WITH YearlySalesData AS (

SELECT

sp.BusinessEntityID,

YEAR(soh.OrderDate) AS Year,

SUM(sod.LineTotal) AS SubTotal,

COUNT(DISTINCT soh.SalesOrderID) AS NumberOfOrders

FROM

Sales.SalesOrderHeader soh

JOIN

Sales.SalesOrderDetail sod

ON

soh.SalesOrderID = sod.SalesOrderID

JOIN

Sales.SalesPerson sp

ON

soh.SalesPersonID = sp.BusinessEntityID

GROUP BY

sp.BusinessEntityID, YEAR(soh.OrderDate)

)

SELECT

p.FirstName + ', ' + p.LastName AS SalesPerson,

e.BusinessEntityID AS EmployeeID,

ysd.Year,

ysd.SubTotal,

ysd.NumberOfOrders

FROM

YearlySalesData ysd

JOIN

HumanResources.Employee e

ON

ysd.BusinessEntityID = e.BusinessEntityID

JOIN

Person.Person p

ON

e.BusinessEntityID = p.BusinessEntityID

**Task 2 – Grouping sets**

-- Using window functions to calculate total sales and number of orders

WITH SalesData AS (

    SELECT

        p.FirstName + ' ' + p.LastName AS SalesPerson,

        sp.BusinessEntityID,

        YEAR(soh.OrderDate) AS Year,

        sod.LineTotal,

        COUNT(\*) OVER (PARTITION BY sp.BusinessEntityID, YEAR(soh.OrderDate)) AS NumberOfOrders,

        SUM(sod.LineTotal) OVER (PARTITION BY sp.BusinessEntityID, YEAR(soh.OrderDate)) AS SubTotal

    FROM Sales.SalesOrderHeader AS soh

    INNER JOIN Sales.SalesOrderDetail AS sod ON soh.SalesOrderID = sod.SalesOrderID

    INNER JOIN Sales.SalesPerson AS sp ON soh.SalesPersonID = sp.BusinessEntityID

    INNER JOIN Person.Person AS p ON sp.BusinessEntityID = p.BusinessEntityID

)

-- Now using GROUPING SETS to get the totals by Year and by EmployeeID

SELECT

    CASE

        WHEN GROUPING(SalesPerson) = 1 THEN 'Total'

        ELSE SalesPerson

    END AS SalesPerson,

    CASE

        WHEN GROUPING(BusinessEntityID) = 1 THEN NULL

        ELSE BusinessEntityID

    END AS EmployeeID,

    CASE

        WHEN GROUPING(Year) = 1 THEN 'Total'

        ELSE CAST(Year AS VARCHAR(4))

    END AS Year,

    SUM(LineTotal) AS SubTotal,

    SUM(NumberOfOrders) AS TotalNumberOfOrders

FROM SalesData

GROUP BY GROUPING SETS (

    (Year, BusinessEntityID, SalesPerson),

    (Year),

    (BusinessEntityID)

)

ORDER BY Year DESC, SalesPerson;

**Task 3 – Pivot Tables**

**1)**

A screenshot of a computer

Description automatically generated

2)

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

3)

4) A screenshot of a computer

Description automatically generated

**Task 4 – Naïve approach**

**1)**

1: Identify Relevant Tables and Attributes

The main tables related to products in the AdventureWorks database are usually:

* Production.Product: Contains details about each product.
* Production.ProductSubcategory: Contains the subcategories for products.
* Production.ProductCategory: Contains the categories for products.
* Sales.SalesOrderDetail: Contains details about each order, including the product sold.

Attributes that are likely relevant for your analysis:

* From Production.Product: Name, ProductNumber, Color, StandardCost, ListPrice, Weight, ProductSubcategoryID.
* From Production.ProductSubcategory: Name, ProductCategoryID.
* From Production.ProductCategory: Name.

2)

SELECT

p.Name AS ProductName,

pc.Name AS CategoryName,

psc.Name AS SubcategoryName,

p.ListPrice AS CataloguePrice,

p.Color,

p.Weight

FROM Production.Product p

JOIN Production.ProductSubcategory psc ON p.ProductSubcategoryID = psc.ProductSubcategoryID

JOIN Production.ProductCategory pc ON psc.ProductCategoryID = pc.ProductCategoryID

WHERE p.SellEndDate IS NULL OR p.SellEndDate > GETDATE() -- Only include products that are still for sale

**3: Justify Attribute Selection**

When selecting attributes for analysis, you'll need to justify why you included or excluded certain attributes:

* **Product Name**: Necessary to identify products.
* **Category/Subcategory Name**: Important for grouping products and analyzing sales by categories.
* **Catalogue Price**: Essential for calculating sales value and profit.
* **Color**: You are specifically interested in comparing sales by color.
* **Weight**: This could be relevant if shipping costs are a factor in sales performance.

Attributes you might choose to exclude:

* **Product Number**: Might not be necessary for the analysis unless you're tracking stock or specific models.
* **Standard Cost**: If you're looking at revenue rather than profit, this might not be necessary.
* **Sell Start/End Date**: In this case, you’re only looking at completed orders, so you don’t need products not yet for sale or those that are no longer available.

**2)**

1: Identify Relevant Tables and Attributes

From a product perspective, the main tables related to orders in the AdventureWorks database are:

* Sales.SalesOrderHeader: This table usually includes the header information for each sales order, such as order date, customer ID, and total due.
* Sales.SalesOrderDetail: This table typically holds line-level details for each sales order, including the product ID, order quantity, and line total.
* Production.Product: Contains details about each product that is sold, such as name, product number, color, list price, and standard cost.

Attributes from these tables that may be useful for analyzing sales performance could include:

* From Sales.SalesOrderHeader:
  + SalesOrderID (to uniquely identify orders)
  + OrderDate (to analyze sales trends over time)
  + TotalDue (to measure the sales amount)
* From Sales.SalesOrderDetail:
  + OrderQty (to measure sales volume)
  + UnitPrice (to measure revenue)
  + LineTotal (also to measure revenue)
* From Production.Product:
  + StandardCost (to calculate the cost of goods sold and thus the profit)
  + ListPrice (to analyze pricing strategy)
  + Other attributes such as Color, Size, or Weight, depending on the specifics of your analysis.

2: Determine the Need for Order Information

b. You typically don't need information about all orders; you can focus on:

* Completed Orders: Only consider orders that have been completed (shipped and invoiced).
* Relevant Time Period: Limit the analysis to a specific period if historical or forecast data is not required.
* Geographical or Demographical Segmentation: You may only be interested in orders from certain locations or demographics.

3: Identify Performance Measures

c. Potential performance measures to analyze the sales process could include:

* Sales Amount: This can be measured using the TotalDue from Sales.SalesOrderHeader.
* Sales Volume: This refers to the quantity of products sold and can be measured using OrderQty from Sales.SalesOrderDetail.
* Profit: Calculated as the difference between the revenue (from LineTotal) and the cost of goods sold (which could be estimated using StandardCost from Production.Product).

4: Justification of Attribute Selection

d. Justifications for selecting these attributes might include:

* Sales Amount and Volume: These are direct measures of sales performance.
* Profit: This is a key indicator of financial health and efficiency.
* Order and Product Details: Specific product attributes like Color or Size may be chosen based on the hypothesis that such characteristics could influence sales performance (e.g., do red products sell better?).

**3)**

SELECT

    P.ProductID,

    P.Name AS ProductName,

    PC.Name AS CategoryName,

    PSC.Name AS SubcategoryName,

    P.Color,

    P.Weight,

    SOD.OrderQty,

    SOD.LineTotal AS SalesAmount,

    DATEPART(year, SOH.OrderDate) AS OrderYear,

    DATEPART(month, SOH.OrderDate) AS OrderMonth,

    DATEPART(day, SOH.OrderDate) AS OrderDay,

    SOH.SalesOrderID,

    (SOD.LineTotal - (P.StandardCost \* SOD.OrderQty)) AS Profit -- Calculating profit

FROM Sales.SalesOrderDetail SOD

INNER JOIN Sales.SalesOrderHeader SOH ON SOD.SalesOrderID = SOH.SalesOrderID

INNER JOIN Production.Product P ON SOD.ProductID = P.ProductID

LEFT JOIN Production.ProductSubcategory PSC ON P.ProductSubcategoryID = PSC.ProductSubcategoryID

LEFT JOIN Production.ProductCategory PC ON PSC.ProductCategoryID = PC.ProductCategoryID

WHERE SOH.Status = 5 -- Assuming status 5 means completed orders; adjust as necessary

AND SOH.OrderDate BETWEEN '2013-01-01' AND '2013-12-31' -- Adjust the date range as needed

ORDER BY SOH.OrderDate;

**5)**

**1 : Information Needs for Basic Analysis**

**a. Identifying Information Needs:**

1. **Sales Performance by Product Attributes: Understanding how specific product features (like color or category) impact sales can help in inventory and marketing decisions. For instance, determining if products with certain attributes (e.g., "red" products) perform better in sales.**
2. **Customer Segmentation and Sales Performance: Analyzing sales data based on customer types (individual vs. corporate) can inform targeted marketing strategies and sales approaches.**

**2: Identifying Tables and Attributes for Customer Analysis**

b. **Customer-Related Data**:

* **Tables**:
  + **Sales.Customer**: Contains customer information. You might need to join this table with others to get detailed demographics or classification.
  + **Person.Person**: Provides more detailed information about individual customers.
  + **Sales.Store**: Information about store customers (if B2B sales are involved).
* **Attributes**:
  + **Customer Type**: There might not be a direct 'Customer Type' field in AdventureWorks. You may classify customers based on their relation to other tables (e.g., linked to **Person.Person** for individuals or **Sales.Store** for businesses).
* **Analysis**:
  + **Total Customers**: Run a query to count distinct customers. You might not need detailed information on all, but understanding the total can help contextualize sales data.
  + **Different Types of Customers**: Introduce a calculated attribute or use an existing distinction to differentiate between customer types for targeted analysis.

**3: Identifying Order Attributes from a Customer Perspective**

c. **Order-Related Data**:

* **Tables**:
  + **Sales.SalesOrderHeader** and **Sales.SalesOrderDetail**: Essential for order information, including product and quantity.
  + **Production.Product**: Product details like color which are crucial for the given analysis question.
* **Performance Measures**:
  + **Sales Volume and Amount**: Quantities and prices from order details can measure the volume and value of sales.
  + **Profit**: Calculated from the sales amount minus the cost associated with sold products. Requires data on product cost (**Production.Product**) and sales price (**Sales.SalesOrderDetail**).

**SQL Query for Data Preparation**

SELECT

P.ProductID,

P.Name AS ProductName,

P.Color,

PC.Name AS CategoryName,

PSC.Name AS SubcategoryName,

C.CustomerID,

CASE

WHEN PP.BusinessEntityID IS NOT NULL THEN 'Individual'

ELSE 'Business'

END AS CustomerType,

SOH.OrderDate,

SOD.OrderQty,

SOD.LineTotal AS SalesAmount,

(SOD.LineTotal - (P.StandardCost \* SOD.OrderQty)) AS Profit

FROM Sales.SalesOrderHeader SOH

JOIN Sales.SalesOrderDetail SOD ON SOH.SalesOrderID = SOD.SalesOrderID

JOIN Production.Product P ON SOD.ProductID = P.ProductID

LEFT JOIN Production.ProductSubcategory PSC ON P.ProductSubcategoryID = PSC.ProductSubcategoryID

LEFT JOIN Production.ProductCategory PC ON PSC.ProductCategoryID = PC.ProductCategoryID

LEFT JOIN Sales.Customer C ON SOH.CustomerID = C.CustomerID

LEFT JOIN Person.Person PP ON C.CustomerID = PP.BusinessEntityID

WHERE SOH.Status = 5 -- Assuming 5 means completed

AND P.Color = 'Red' -- For the specific question about red products

**2) prepare query**

**a. Customer Details**

CREATE VIEW CustomerDetails AS

SELECT

C.CustomerID,

CASE

WHEN P.BusinessEntityID IS NOT NULL THEN 'Individual'

ELSE 'Company'

END AS CustomerType,

P.FirstName,

P.MiddleName,

P.LastName

FROM Sales.Customer AS C

LEFT JOIN Person.Person AS P ON C.CustomerID = P.BusinessEntityID**;**

**b. Product Details**

CREATE VIEW ProductDetails AS

SELECT

P.ProductID,

P.Name AS ProductName,

PC.Name AS CategoryName,

PSC.Name AS SubcategoryName,

P.Color,

P.Weight,

P.ListPrice AS Price,

P.StandardCost,

(P.ListPrice - P.StandardCost) AS ProfitMargin

FROM Production.Product AS P

LEFT JOIN Production.ProductSubcategory AS PSC ON P.ProductSubcategoryID = PSC.ProductSubcategoryID

LEFT JOIN Production.ProductCategory AS PC ON PSC.ProductCategoryID = PC.ProductCategoryID;

**c. Time Details**

CREATE VIEW TimeDetails AS

SELECT

OrderDate AS FullDate,

DATEPART(day, OrderDate) AS Day,

DATEPART(month, OrderDate) AS Month,

DATEPART(year, OrderDate) AS Year

FROM Sales.SalesOrderHeader

GROUP BY OrderDate;

**d. Order Details**

CREATE VIEW OrderDetails AS

SELECT

SOH.SalesOrderID,

SOH.CustomerID,

SOD.ProductID,

SOH.OrderDate,

SOD.OrderQty AS Quantity,

SOD.LineTotal AS OrderValue

FROM Sales.SalesOrderDetail AS SOD

JOIN Sales.SalesOrderHeader AS SOH ON SOD.SalesOrderID = SOH.SalesOrderID;