Mini Project DE

DSLS 2023

**By Ichfan Kurniawan**

# CONTENT

[CONTENT 2](#_Toc125296902)

[2A INTERMEDIATE QUERIES 3](#_Toc125296903)

[2B CASE STUDIES 13](#_Toc125296904)

[1. PRODUCT ANALYSIS 13](#_Toc125296905)

[1.1. Background & Finding 13](#_Toc125296906)

[1.2. The Process (SQL Query, Flowchart, Visualization) 14](#_Toc125296907)

[2. CUSTOMER ANALYSIS – RFM 39](#_Toc125296908)

[2.1. Background & Finding 39](#_Toc125296909)

[2.2. The Process (SQL Query, Flowchart, Visualization) 39](#_Toc125296910)

[3. COHORT ANALYSIS – RETENTION (MONTHLY) 43](#_Toc125296911)

[3.1. Background & Finding 43](#_Toc125296912)

[3.2. The Process (SQL Query, Flowchart, Visualization) 43](#_Toc125296913)

# 2A INTERMEDIATE QUERIES

Untuk mereview kembali kemampuan dasar query, Anda diminta untuk dapat mengerjakan soal ini terlebih dahulu :

1. Tulis query untuk mendapatkan jumlah customer tiap bulan yang melakukan order pada tahun 1997.
2. Tulis query untuk mendapatkan nama employee yang termasuk Sales Representative.
3. Tulis query untuk mendapatkan top 5 nama produk yang quantitynya paling banyak diorder pada bulan Januari 1997.
4. Tulis query untuk mendapatkan nama company yang melakukan order Chai pada bulan Juni 1997.
5. Tulis query untuk mendapatkan jumlah OrderID yang pernah melakukan sales (unit\_price dikali quantity) <=100, 100<x<=250, 250<x<=500, dan >500.
6. Tulis query untuk mendapatkan Company name yang melakukan sales di atas 500 pada tahun 1997.
7. Tulis query untuk mendapatkan nama produk yang merupakan Top 5 sales tertinggi tiap bulan di tahun 1997.
8. Buatlah view untuk melihat Order Details yang berisi OrderID, ProductID, ProductName, UnitPrice, Quantity, Discount, Harga setelah diskon.
9. Buatlah procedure Invoice untuk memanggil CustomerID, CustomerName, OrderID, OrderDate, RequiredDate, ShippedDate jika terdapat inputan CustomerID tertentu.

--1. Tulis query untuk mendapatkan jumlah customer tiap bulan yang melakukan order pada tahun 1997.

SELECT

DATEPART(MONTH, OrderDate) as month

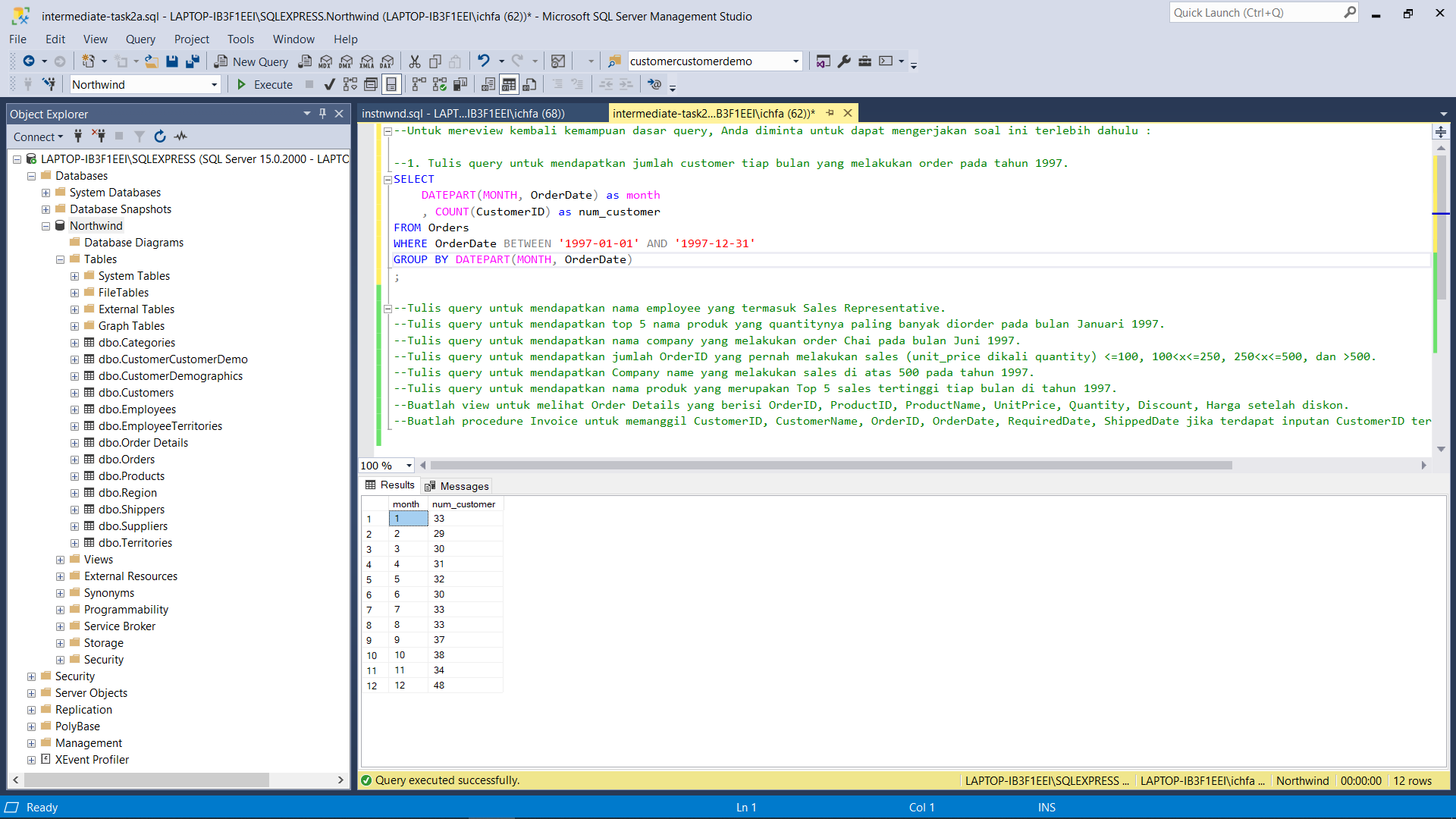
, COUNT(CustomerID) as num\_customer

FROM Orders

WHERE OrderDate BETWEEN '1997-01-01' AND '1997-12-31'

GROUP BY DATEPART(MONTH, OrderDate)

;



--2. Tulis query untuk mendapatkan nama employee yang termasuk Sales Representative.

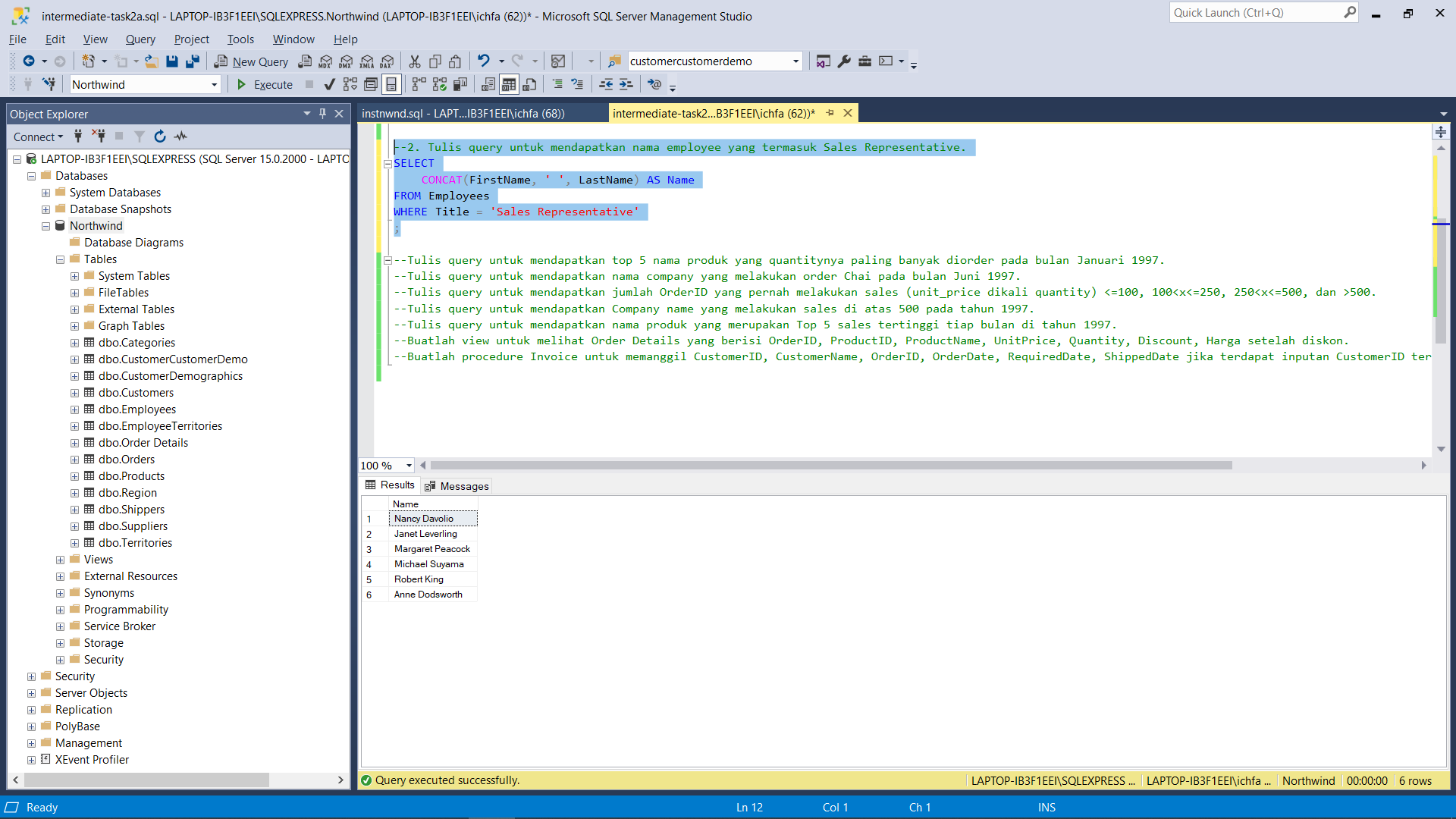
SELECT

CONCAT(FirstName, ' ', LastName) AS Name

FROM Employees

WHERE Title = 'Sales Representative'

;



--3. Tulis query untuk mendapatkan top 5 nama produk yang quantitynya paling banyak diorder pada bulan Januari 1997.

SELECT

p.ProductName

, SUM(d.Quantity) as tot\_quantity

FROM Orders o

JOIN [Order Details] d ON o.OrderID = d.OrderID

JOIN Products p ON d.ProductID = p.ProductID

WHERE OrderDate BETWEEN '1997-01-01' AND '1997-01-31'

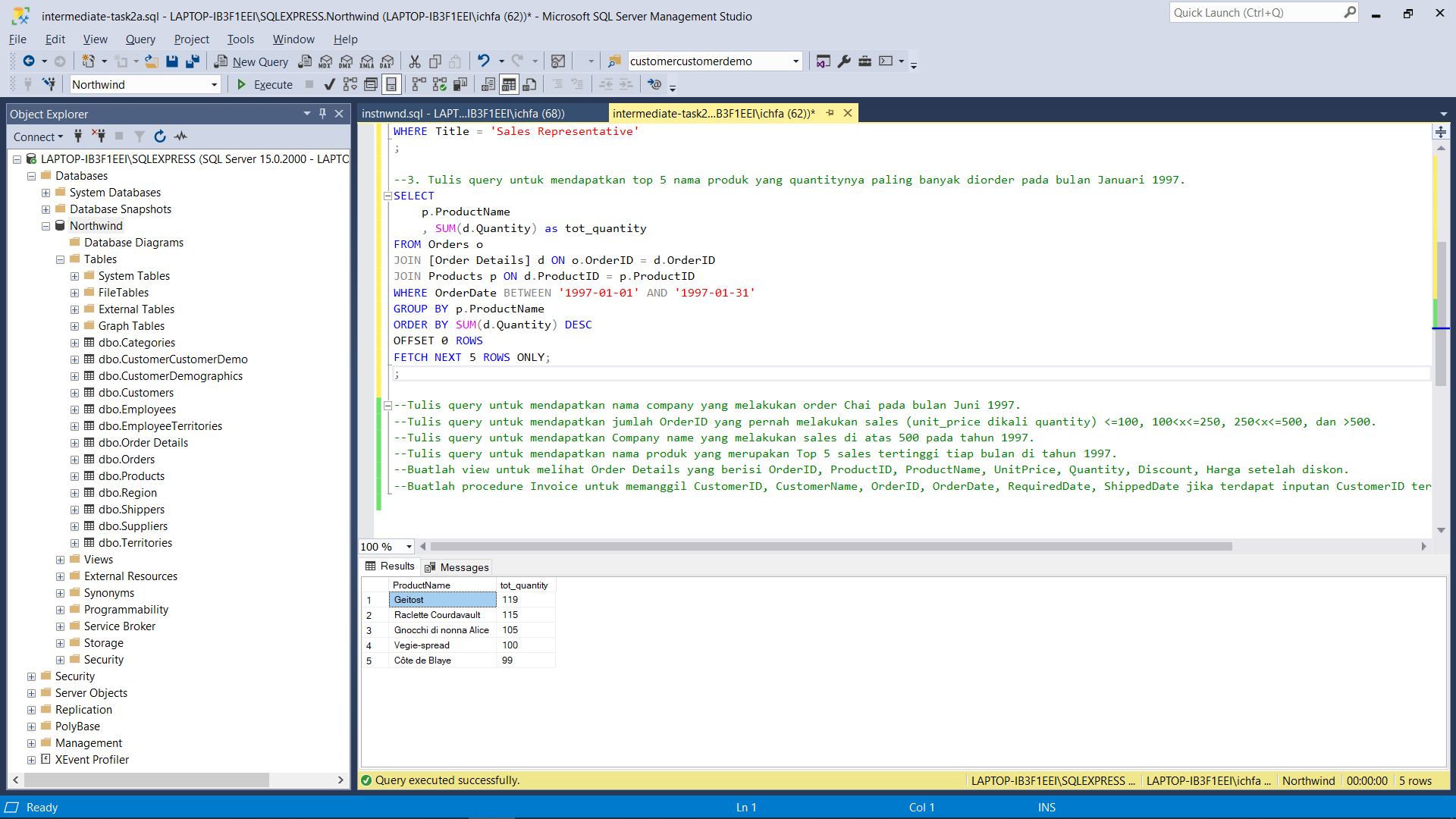
GROUP BY p.ProductName

ORDER BY SUM(d.Quantity) DESC

OFFSET 0 ROWS

FETCH NEXT 5 ROWS ONLY;

;



--4. Tulis query untuk mendapatkan nama company yang melakukan order Chai pada bulan Juni 1997.

SELECT

DISTINCT c.CompanyName

FROM Orders o

JOIN [Order Details] d ON o.OrderID = d.OrderID

JOIN Products p ON d.ProductID = p.ProductID

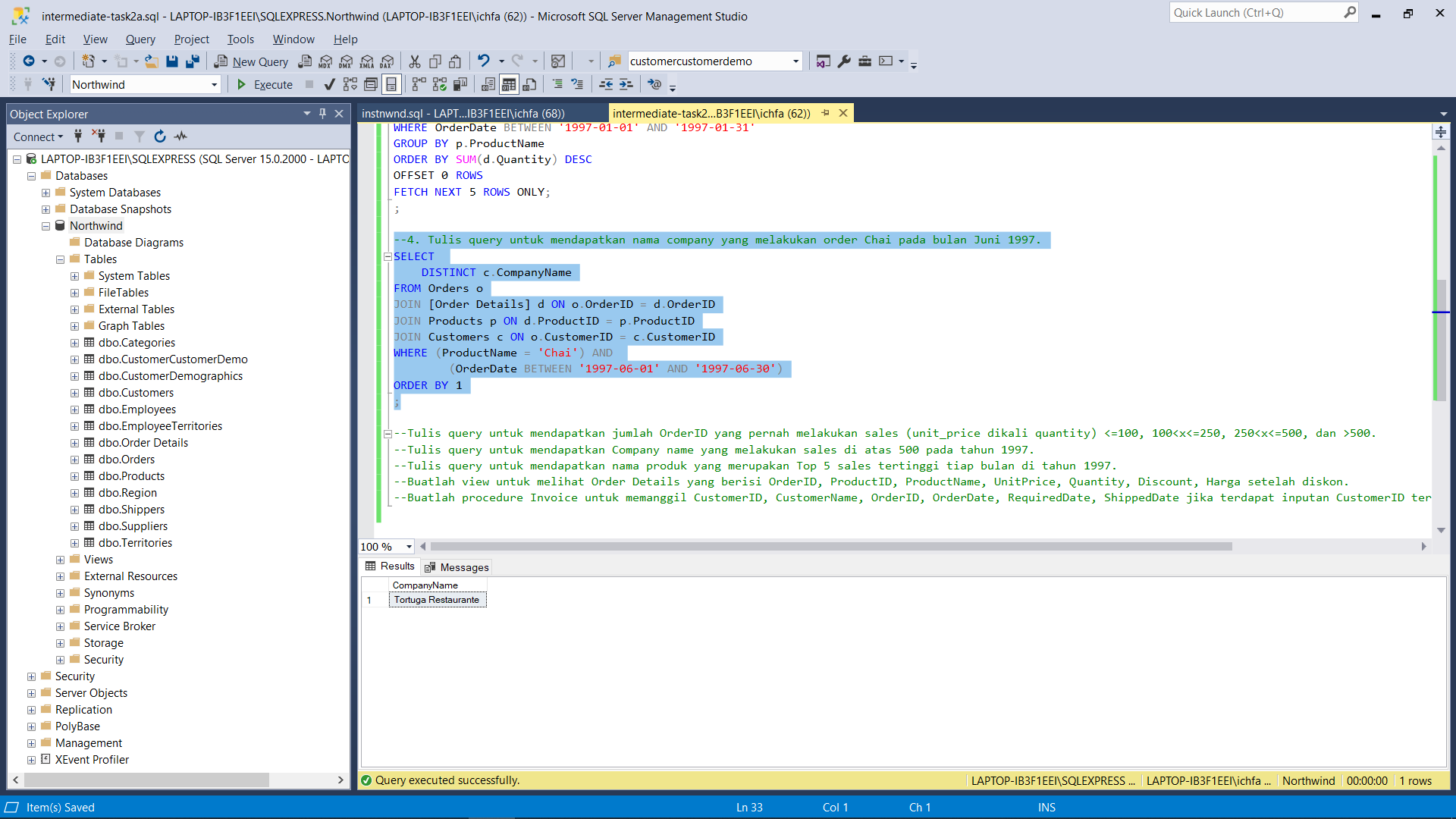
JOIN Customers c ON o.CustomerID = c.CustomerID

WHERE (ProductName = 'Chai') AND

(OrderDate BETWEEN '1997-06-01' AND '1997-06-30')

ORDER BY 1

;



--5. Tulis query untuk mendapatkan jumlah OrderID yang pernah melakukan sales (unit\_price dikali quantity) <=100, 100<x<=250, 250<x<=500, dan >500.

WITH sales\_table(OrderID, sales) AS(

SELECT

OrderID

, SUM(UnitPrice\*Quantity)

FROM [Order Details]

GROUP BY OrderID)

SELECT

COUNT(CASE WHEN sales<=100 THEN OrderID ELSE NULL END) AS 'sales<=100'

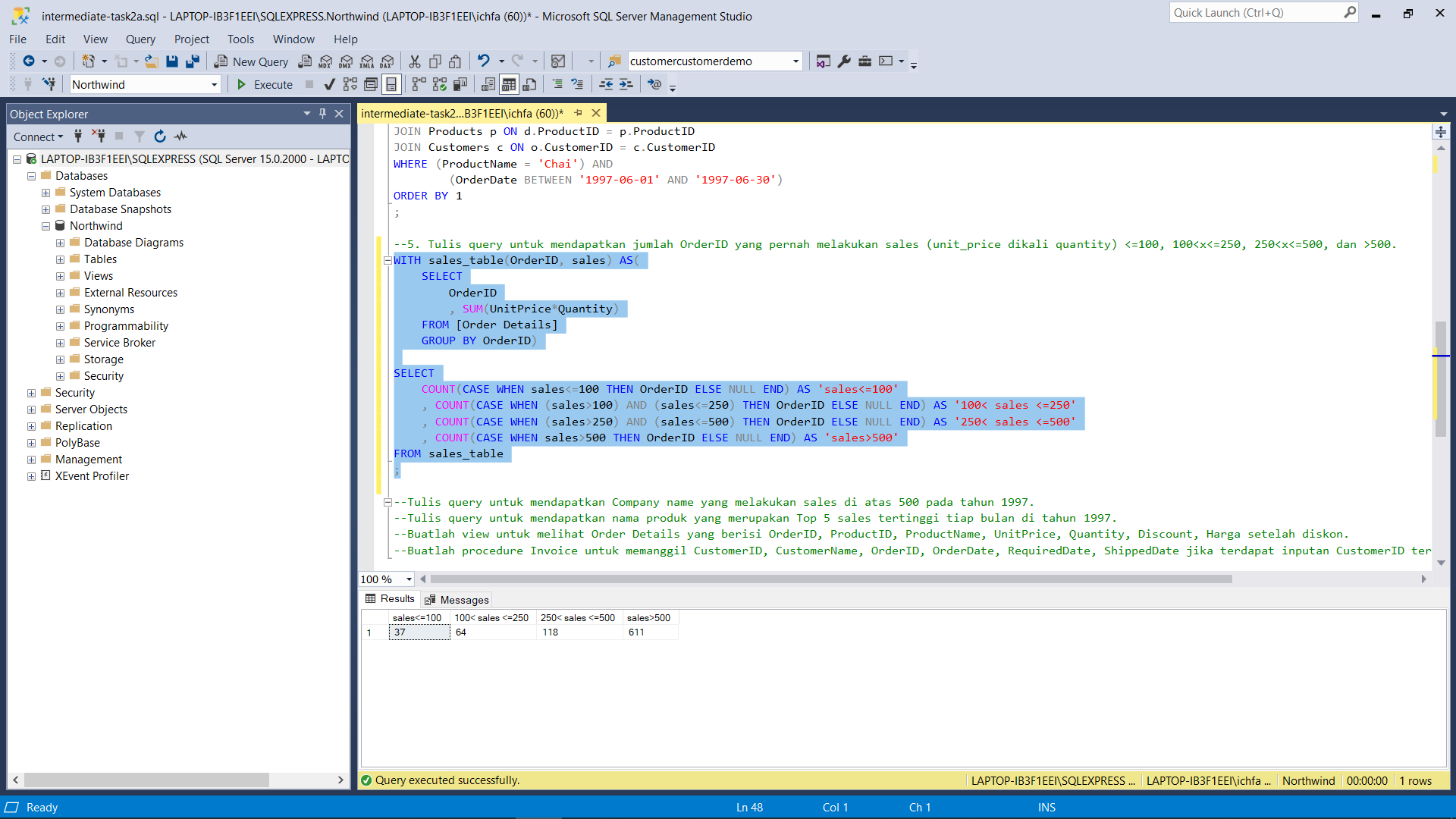
, COUNT(CASE WHEN (sales>100) AND (sales<=250) THEN OrderID ELSE NULL END) AS '100< sales <=250'

, COUNT(CASE WHEN (sales>250) AND (sales<=500) THEN OrderID ELSE NULL END) AS '250< sales <=500'

, COUNT(CASE WHEN sales>500 THEN OrderID ELSE NULL END) AS 'sales>500'

FROM sales\_table

;



--6. Tulis query untuk mendapatkan Company name yang melakukan sales di atas 500 pada tahun 1997.

WITH sales\_table(OrderID, sales) AS(

SELECT

OrderID

, SUM(UnitPrice\*Quantity)

FROM [Order Details]

GROUP BY OrderID)

SELECT

DISTINCT CompanyName

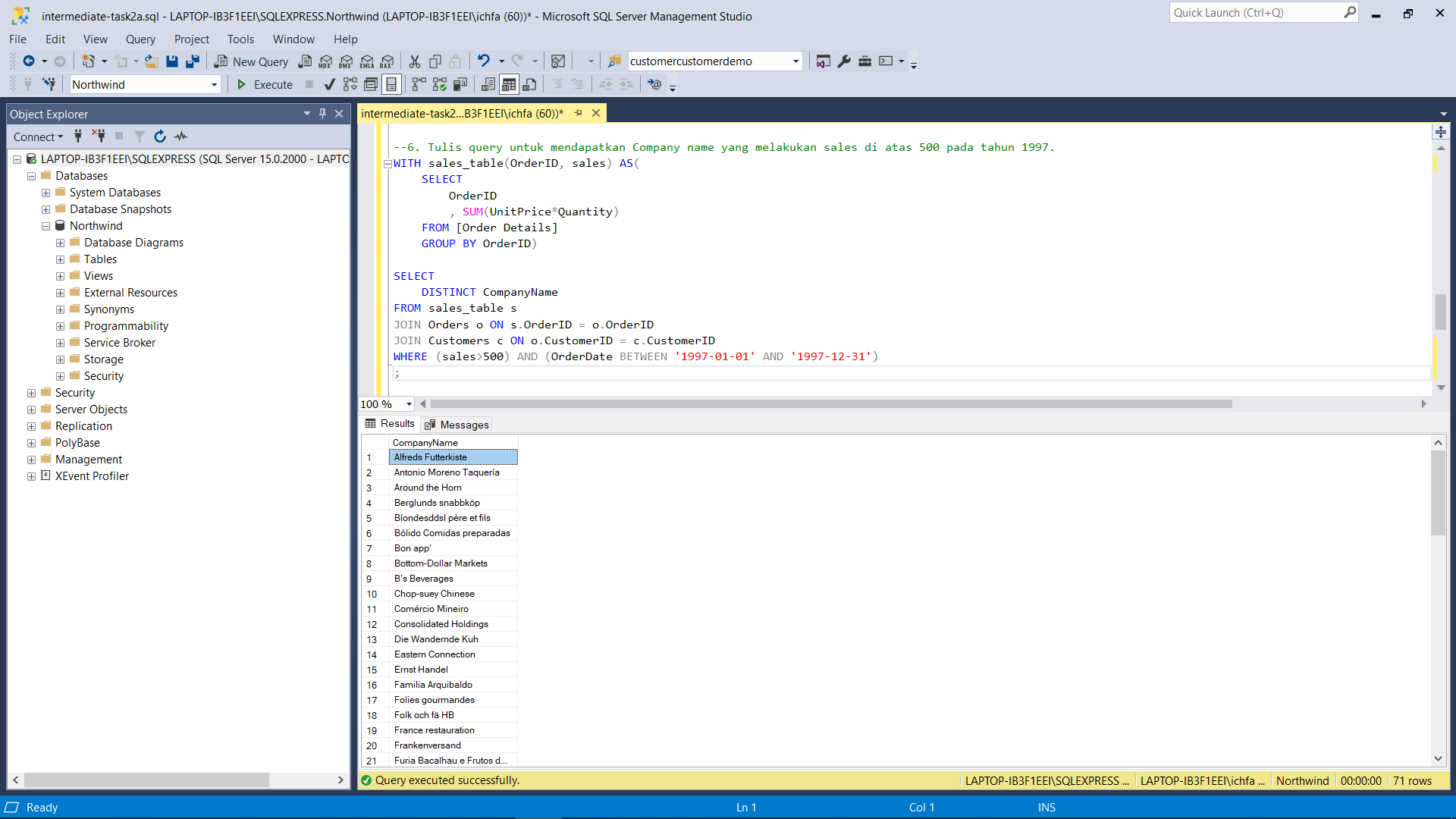
FROM sales\_table s

JOIN Orders o ON s.OrderID = o.OrderID

JOIN Customers c ON o.CustomerID = c.CustomerID

WHERE (sales>500) AND (OrderDate BETWEEN '1997-01-01' AND '1997-12-31')

;



--7. Tulis query untuk mendapatkan nama produk yang merupakan Top 5 sales tertinggi tiap bulan di tahun 1997.

-- month | product | rank | (output looks like)

WITH sales\_product(month, ProductName, sales, ranking) AS(

SELECT

MONTH(o.OrderDate)

, ProductName

, SUM(d.UnitPrice\*d.Quantity)

, ROW\_NUMBER() OVER(PARTITION BY MONTH(o.OrderDate) ORDER BY SUM(d.UnitPrice\*d.Quantity) DESC)

FROM Orders o

JOIN [Order Details] d ON o.OrderID = d.OrderID

JOIN Products p ON d.ProductID = p.ProductID

WHERE OrderDate BETWEEN '1997-01-01' AND '1997-12-31'

GROUP BY MONTH(o.OrderDate), ProductName)

SELECT \*

FROM sales\_product

WHERE ranking <= 5

ORDER BY 'month', ranking

;



--8. Buatlah view untuk melihat Order Details yang berisi OrderID, ProductID, ProductName, UnitPrice, Quantity, Discount, Harga setelah diskon.

USE Northwind;

-- creating the view

CREATE VIEW v\_order\_details(OrderID, ProductID, ProductName, UnitPrice, Quantity, Discount, discounted\_price)

AS

SELECT

o.OrderID

, o.ProductID

, ProductName

, o.UnitPrice

, o.Quantity

, o.Discount

, (1.0 - o.Discount)\*o.UnitPrice

FROM [Order Details] o

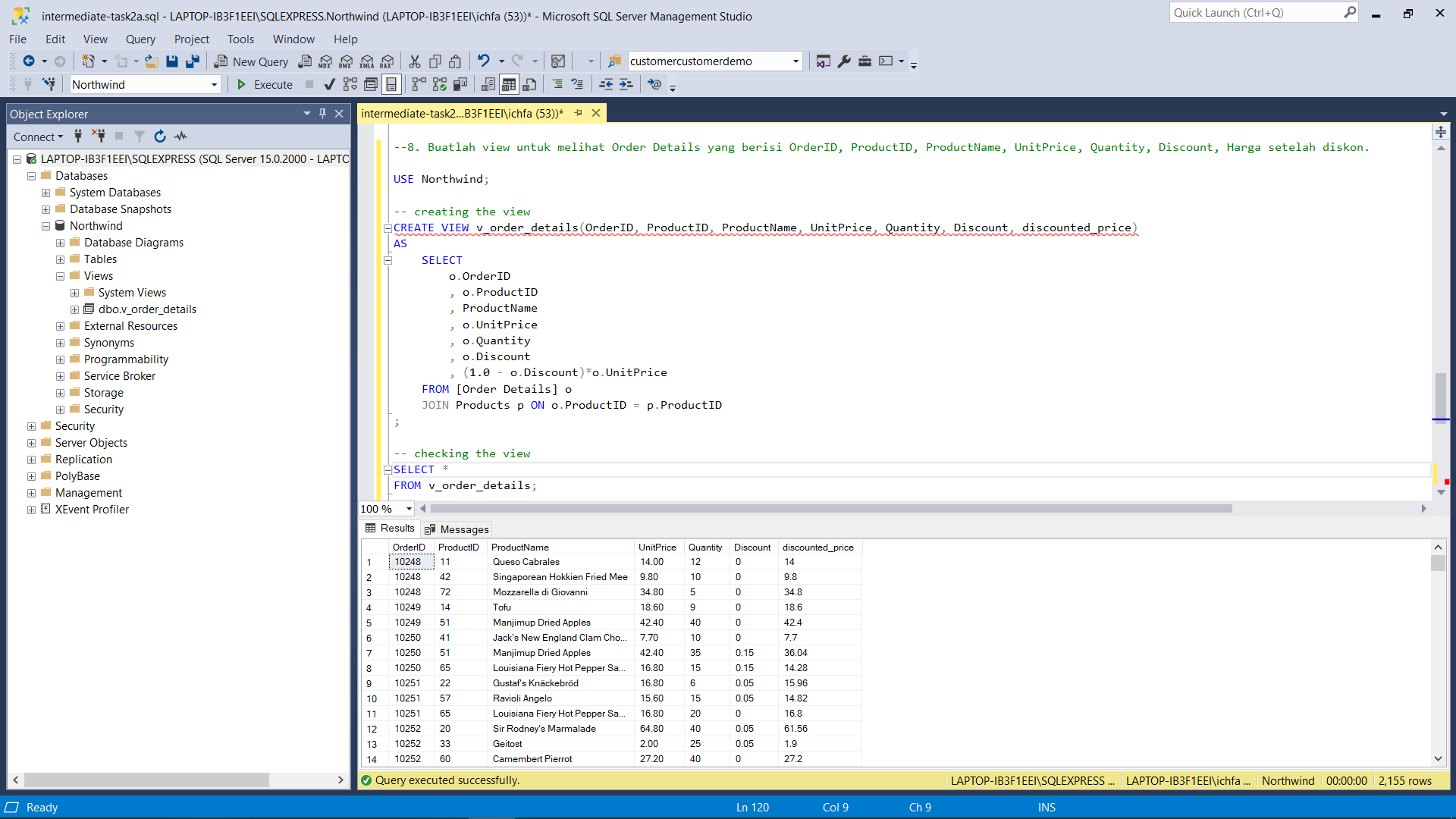
JOIN Products p ON o.ProductID = p.ProductID

;

-- checking the view

SELECT \*

FROM v\_order\_details;



-- 9. Buatlah procedure Invoice untuk memanggil CustomerID, CustomerName, OrderID, OrderDate, RequiredDate, ShippedDate jika terdapat inputan CustomerID tertentu.

-- Create the procedure

CREATE PROCEDURE pr\_invoice (@cust\_id AS nchar(5))

AS

BEGIN

SELECT

o.CustomerID,

c.ContactName,

o.OrderID,

o.OrderDate,

o.RequiredDate,

o.ShippedDate

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID

WHERE o.CustomerID = @cust\_id

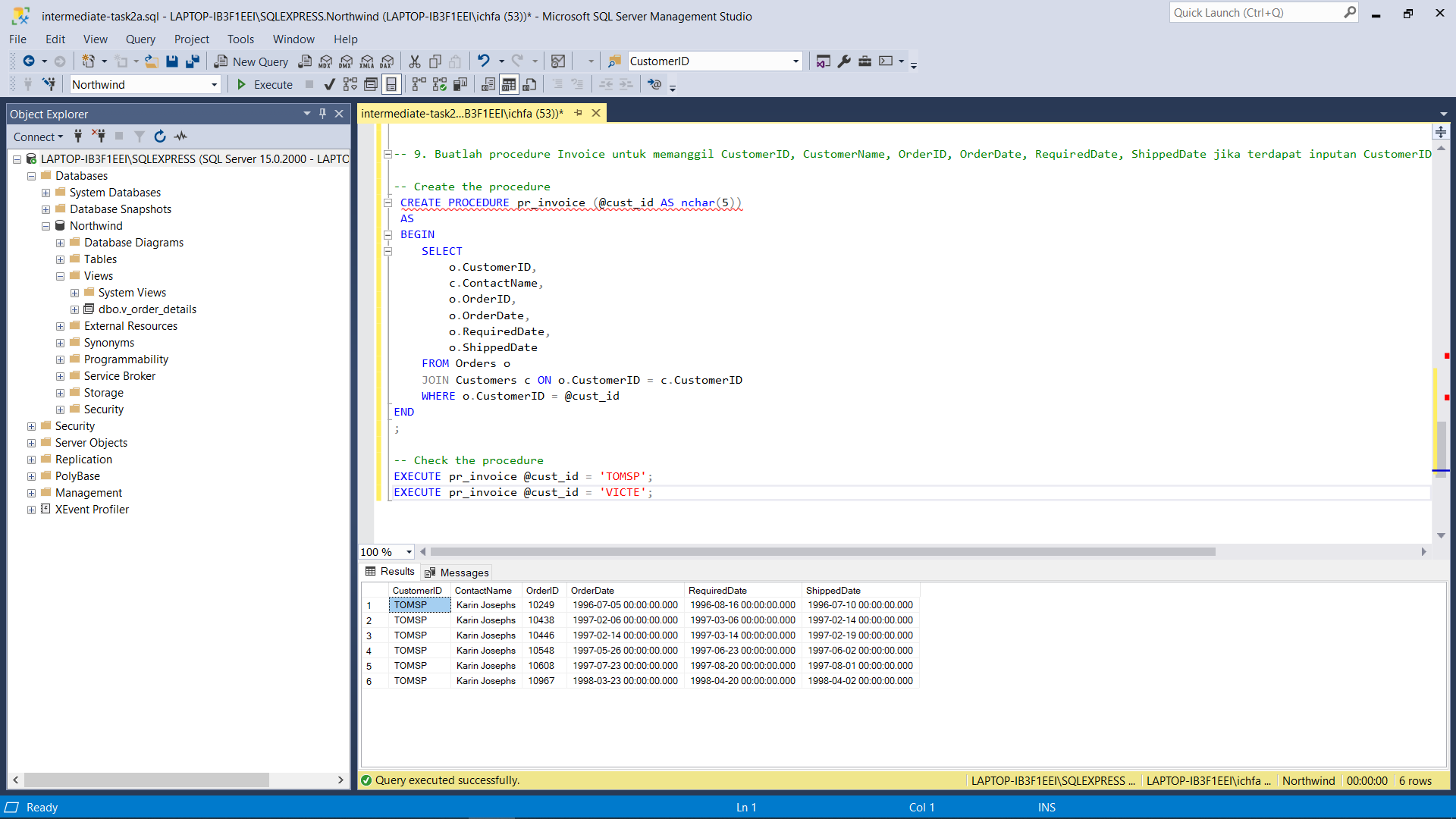
END

;

-- Check the procedure

EXECUTE pr\_invoice @cust\_id = 'TOMSP';

EXECUTE pr\_invoice @cust\_id = 'VICTE';



# 2B CASE STUDIES

## PRODUCT ANALYSIS

### Background & Finding

In this case study, the store tries to find insights from the overall trend of the business & particular trends that can be anticipated, repeated, & eventually improved to boost the revenue of the business.

Based on the analysis, we found some meaningful insights below:

1. The overall trend of business is good & increasing over the time period of late 1996 to early 1998 as shown in Table 1.2. a. In Addition, the increase in sales was driven by the increase of number quantities in the purchase. Because the only period that capture the full year phenomena is in 1997, from now on the analysis will be focussing in the 1997 period only.
2. For focusing our effort using target marketing in the 1997 period, we could find
   * 1. the Pareto of the most contributed company name in the sales as shown in Table 1.2. b .
     2. the Pareto of the most contributed product name in the sales as shown in Table 1.2. c
     3. the Pareto of the most contributed category name in the sales as shown in Table 1.2. d
     4. the Pareto of the most contributed territory name in the sales as shown in Table 1.2. e
     5. the Pareto of the most contributed region name in the sales as shown in Table 1.2. f
3. We find that eastern region is the most contributor to our sales, we could deep down dive into eastern region only below:
   * 1. the Pareto of the most contributed territory name in the sales of Eastern Region as shown in Table 1.2. g
     2. the Pareto of the most contributed category name in the sales of Eastern Region as shown in Table 1.2. h
     3. the Pareto of the most contributed product name in the sales of Eastern Region as shown in Table 1.2. i
4. We notice that in the period from 1997/11/01 to 1998/04/30, there was spike of sales, we could deep down dive into it only below:
   * 1. the Pareto of the most contributed region name in the sales of the spike period is the Eastern Region as shown in Table 1.2. j. From now on, lets only focus on Eastern region only in the spike period.
     2. the Pareto of the most contributed territory name in the sales of Eastern Region in the spike period as shown in Table 1.2. k
     3. the Pareto of the most contributed category name in the sales of Eastern Region in the spike period as shown in Table 1.2. l
     4. the Pareto of the most contributed product name in the sales of Eastern Region in the spike period as shown in Table 1.2. m

### The Process (SQL Query, Flowchart, Visualization)

Table 1.2. a. The process to get finding

|  |
| --- |
| Finding 1.2.a |
| SELECT  YEAR(o.OrderDate) AS 'year'  , MONTH(o.OrderDate) AS 'month'  , SUM(od.UnitPrice \* od.Quantity) AS 'sum\_sales'  , ROUND(100.0\*(SUM(od.UnitPrice \* od.Quantity) - LAG(SUM(od.UnitPrice \* od.Quantity), 1) OVER(ORDER BY YEAR(o.OrderDate), MONTH(o.OrderDate))) /  LAG(SUM(od.UnitPrice \* od.Quantity), 1) OVER(ORDER BY YEAR(o.OrderDate), MONTH(o.OrderDate)), 2) AS '%\_chg\_sum\_sales'    , AVG(od.UnitPrice \* od.Quantity) AS 'avg\_sales'  , COUNT(od.UnitPrice \* od.Quantity) AS 'vol\_sales'  , SUM(od.Quantity) AS 'sum\_qty'  , ROUND(100.0\*(SUM(od.Quantity) - LAG(SUM(od.Quantity), 1) OVER(ORDER BY YEAR(o.OrderDate), MONTH(o.OrderDate))) /  LAG(SUM(od.Quantity), 1) OVER(ORDER BY YEAR(o.OrderDate), MONTH(o.OrderDate)), 2) AS '%\_chg\_sum\_qty'  , AVG(od.UnitPrice) AS 'avg\_u\_price'  , ROUND(100.0\*(AVG(od.UnitPrice) - LAG(AVG(od.UnitPrice), 1) OVER(ORDER BY YEAR(o.OrderDate), MONTH(o.OrderDate))) /  LAG(AVG(od.UnitPrice), 1) OVER(ORDER BY YEAR(o.OrderDate), MONTH(o.OrderDate)), 2) AS '%\_chg\_avg\_u\_price'  FROM Orders o  JOIN [Order Details] od ON o.OrderID = od.OrderID  GROUP BY YEAR(o.OrderDate), MONTH(o.OrderDate)  ORDER BY YEAR(o.OrderDate), MONTH(o.OrderDate)  ; |
|  |
|  |
|  |
|  |

Table 1.2. b . The process to get finding

|  |
| --- |
| Finding 1.2.b |
| WITH company\_sales AS  (SELECT  c.CompanyName  , SUM(od.UnitPrice\*od.Quantity) as 'sum\_sales'  FROM Orders o  JOIN Customers c ON o.CustomerID = c.CustomerID  JOIN [Order Details] od ON o.OrderID = od.OrderID  WHERE o.OrderDate BETWEEN '1997-01-01' AND '1997-12-31'  GROUP BY c.CompanyName)  SELECT  \*  , 100.0 \* sum\_sales / (SELECT SUM(sum\_sales) FROM company\_sales) AS '%\_sales'  , 100.0 \* SUM(sum\_sales) OVER(ORDER BY sum\_sales DESC ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) / (SELECT SUM(sum\_sales) FROM company\_sales) as '%\_cumsum\_sales'  FROM company\_sales  ; |
|  |
|  |
|  |

Table 1.2. c The process to get finding

|  |
| --- |
| Finding 1.2.c |
| WITH product\_sales AS  (SELECT  p.ProductName  , SUM(od.UnitPrice\*od.Quantity) as 'sum\_sales'  FROM Orders o  JOIN [Order Details] od ON o.OrderID = od.OrderID  JOIN Products p ON od.ProductID = p.ProductID  WHERE o.OrderDate BETWEEN '1997-01-01' AND '1997-12-31'  GROUP BY p.ProductName)  SELECT  \*  , 100.0 \* sum\_sales / (SELECT SUM(sum\_sales) FROM product\_sales) AS '%\_sales'  , 100.0 \* SUM(sum\_sales) OVER(ORDER BY sum\_sales DESC ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) / (SELECT SUM(sum\_sales) FROM product\_sales) as '%\_cumsum\_sales'  FROM product\_sales; |
|  |
|  |
|  |

Table 1.2. d The process to get finding

|  |
| --- |
| Finding 1.2.d |
| WITH category\_sales AS  (SELECT  c.CategoryName  , SUM(od.UnitPrice\*od.Quantity) as 'sum\_sales'  FROM Orders o  JOIN [Order Details] od ON o.OrderID = od.OrderID  JOIN Products p ON od.ProductID = p.ProductID  JOIN Categories c ON p.CategoryID = c.CategoryID  WHERE o.OrderDate BETWEEN '1997-01-01' AND '1997-12-31'  GROUP BY c.CategoryName)  SELECT  \*  , 100.0 \* sum\_sales / (SELECT SUM(sum\_sales) FROM category\_sales) AS '%\_sales'  , 100.0 \* SUM(sum\_sales) OVER(ORDER BY sum\_sales DESC ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) / (SELECT SUM(sum\_sales) FROM category\_sales) as '%\_cumsum\_sales'  FROM category\_sales  ; |
|  |
|  |
|  |

Table 1.2. e The process to get finding

|  |
| --- |
| Finding 1.2.e |
| WITH teritory\_sales AS  (SELECT  t.TerritoryDescription  , SUM(od.UnitPrice\*od.Quantity) as 'sum\_sales'  FROM Orders o  JOIN [Order Details] od ON o.OrderID = od.OrderID  JOIN Employees e ON e.EmployeeID = o.EmployeeID  JOIN EmployeeTerritories et ON et.EmployeeID = o.EmployeeID  JOIN Territories t ON et.TerritoryID = t.TerritoryID  WHERE o.OrderDate BETWEEN '1997-01-01' AND '1997-12-31'  GROUP BY t.TerritoryDescription)  SELECT  \*  , 100.0 \* sum\_sales / (SELECT SUM(sum\_sales) FROM teritory\_sales) AS '%\_sales'  , 100.0 \* SUM(sum\_sales) OVER(ORDER BY sum\_sales DESC ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) / (SELECT SUM(sum\_sales) FROM teritory\_sales) as '%\_cumsum\_sales'  FROM teritory\_sales  ; |
|  |
|  |
|  |

Table 1.2. f. The process to get finding

|  |
| --- |
| Finding 1.2.e |
| WITH region\_sales AS  (SELECT  r.RegionDescription  , SUM(od.UnitPrice\*od.Quantity) as 'sum\_sales'  FROM Orders o  JOIN [Order Details] od ON o.OrderID = od.OrderID  JOIN Employees e ON e.EmployeeID = o.EmployeeID  JOIN EmployeeTerritories et ON et.EmployeeID = o.EmployeeID  JOIN Territories t ON et.TerritoryID = t.TerritoryID  JOIN Region r ON r.RegionID = t.RegionID  WHERE o.OrderDate BETWEEN '1997-01-01' AND '1997-12-31'  GROUP BY r.RegionDescription)  SELECT  \*  , 100.0 \* sum\_sales / (SELECT SUM(sum\_sales) FROM region\_sales) AS '%\_sales'  , 100.0 \* SUM(sum\_sales) OVER(ORDER BY sum\_sales DESC ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) / (SELECT SUM(sum\_sales) FROM region\_sales) as '%\_cumsum\_sales'  FROM region\_sales  ; |
|  |
|  |
|  |

Table 1.2. g

|  |
| --- |
| Finding 1.2.g |
| WITH territory\_sales AS  (SELECT  t.TerritoryDescription  , SUM(od.UnitPrice\*od.Quantity) as 'sum\_sales'  FROM Orders o  JOIN [Order Details] od ON o.OrderID = od.OrderID  JOIN Employees e ON e.EmployeeID = o.EmployeeID  JOIN EmployeeTerritories et ON et.EmployeeID = o.EmployeeID  JOIN Territories t ON et.TerritoryID = t.TerritoryID  JOIN Region r ON r.RegionID = t.RegionID  WHERE (o.OrderDate BETWEEN '1997-01-01' AND '1997-12-31') AND (RegionDescription = 'Eastern')  GROUP BY t.TerritoryDescription)  SELECT  \*  , 100.0 \* sum\_sales / (SELECT SUM(sum\_sales) FROM territory\_sales) AS '%\_sales'  , 100.0 \* SUM(sum\_sales) OVER(ORDER BY sum\_sales DESC ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) / (SELECT SUM(sum\_sales) FROM territory\_sales) as '%\_cumsum\_sales'  FROM territory\_sales  ; |
|  |
|  |

Table 1.2. h

|  |
| --- |
| Finding 1.2.h |
| WITH category\_sales AS  (SELECT  c.CategoryName  , SUM(od.UnitPrice\*od.Quantity) as 'sum\_sales'  FROM Orders o  JOIN [Order Details] od ON o.OrderID = od.OrderID  JOIN Products p ON od.ProductID = p.ProductID  JOIN Categories c ON p.CategoryID = c.CategoryID  JOIN Employees e ON e.EmployeeID = o.EmployeeID  JOIN EmployeeTerritories et ON et.EmployeeID = o.EmployeeID  JOIN Territories t ON et.TerritoryID = t.TerritoryID  JOIN Region r ON r.RegionID = t.RegionID  WHERE (o.OrderDate BETWEEN '1997-01-01' AND '1997-12-31') AND (RegionDescription = 'Eastern')  GROUP BY c.CategoryName)  SELECT  \*  , 100.0 \* sum\_sales / (SELECT SUM(sum\_sales) FROM category\_sales) AS '%\_sales'  , 100.0 \* SUM(sum\_sales) OVER(ORDER BY sum\_sales DESC ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) / (SELECT SUM(sum\_sales) FROM category\_sales) as '%\_cumsum\_sales'  FROM category\_sales  ; |
|  |
|  |
|  |

Table 1.2. i

|  |
| --- |
| Finding 1.2.i |
| WITH product\_sales AS  (SELECT  p.ProductName  , SUM(od.UnitPrice\*od.Quantity) as 'sum\_sales'  FROM Orders o  JOIN [Order Details] od ON o.OrderID = od.OrderID  JOIN Products p ON od.ProductID = p.ProductID  JOIN Categories c ON p.CategoryID = c.CategoryID  JOIN EmployeeTerritories et ON et.EmployeeID = o.EmployeeID  JOIN Territories t ON et.TerritoryID = t.TerritoryID  JOIN Region r ON r.RegionID = t.RegionID  WHERE (o.OrderDate BETWEEN '1997-01-01' AND '1997-12-31') AND (RegionDescription = 'Eastern')  GROUP BY p.ProductName)  SELECT  \*  , 100.0 \* sum\_sales / (SELECT SUM(sum\_sales) FROM product\_sales) AS '%\_sales'  , 100.0 \* SUM(sum\_sales) OVER(ORDER BY sum\_sales DESC ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) / (SELECT SUM(sum\_sales) FROM product\_sales) as '%\_cumsum\_sales'  FROM product\_sales  ; |
|  |
|  |
|  |

Table 1.2. j

|  |
| --- |
| Finding 1.2.j |
| WITH region\_sales AS  (SELECT  r.RegionDescription  , SUM(od.UnitPrice\*od.Quantity) as 'sum\_sales'  FROM Orders o  JOIN [Order Details] od ON o.OrderID = od.OrderID  JOIN EmployeeTerritories et ON et.EmployeeID = o.EmployeeID  JOIN Territories t ON et.TerritoryID = t.TerritoryID  JOIN Region r ON r.RegionID = t.RegionID  WHERE o.OrderDate BETWEEN '1997-11-01' AND '1998-04-30'  GROUP BY r.RegionDescription)  SELECT  \*  , 100.0 \* sum\_sales / (SELECT SUM(sum\_sales) FROM region\_sales) AS '%\_sales'  , 100.0 \* SUM(sum\_sales) OVER(ORDER BY sum\_sales DESC ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) / (SELECT SUM(sum\_sales) FROM region\_sales) as '%\_cumsum\_sales'  FROM region\_sales  ; |
|  |
|  |
|  |

Table 1.2. k

|  |
| --- |
| Finding 1.2.k |
| WITH territory\_sales AS  (SELECT  t.TerritoryDescription  , SUM(od.UnitPrice\*od.Quantity) as 'sum\_sales'  FROM Orders o  JOIN [Order Details] od ON o.OrderID = od.OrderID  JOIN Employees e ON e.EmployeeID = o.EmployeeID  JOIN EmployeeTerritories et ON et.EmployeeID = o.EmployeeID  JOIN Territories t ON et.TerritoryID = t.TerritoryID  JOIN Region r ON r.RegionID = t.RegionID  WHERE (o.OrderDate BETWEEN '1997-11-01' AND '1998-04-30') AND (RegionDescription = 'Eastern')  GROUP BY t.TerritoryDescription)  SELECT  \*  , 100.0 \* sum\_sales / (SELECT SUM(sum\_sales) FROM territory\_sales) AS '%\_sales'  , 100.0 \* SUM(sum\_sales) OVER(ORDER BY sum\_sales DESC ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) / (SELECT SUM(sum\_sales) FROM territory\_sales) as '%\_cumsum\_sales'  FROM territory\_sales  ; |
|  |
|  |

Table 1.2. l

|  |
| --- |
| Finding 1.2.l |
| -- pareto category name on sales in eastern region  -- category name in eastern with the highest sales on Nov 97 to Apr 98 period  WITH category\_sales AS  (SELECT  c.CategoryName  , SUM(od.UnitPrice\*od.Quantity) as 'sum\_sales'  FROM Orders o  JOIN [Order Details] od ON o.OrderID = od.OrderID  JOIN Products p ON od.ProductID = p.ProductID  JOIN Categories c ON p.CategoryID = c.CategoryID  JOIN Employees e ON e.EmployeeID = o.EmployeeID  JOIN EmployeeTerritories et ON et.EmployeeID = o.EmployeeID  JOIN Territories t ON et.TerritoryID = t.TerritoryID  JOIN Region r ON r.RegionID = t.RegionID  WHERE (o.OrderDate BETWEEN '1997-11-01' AND '1998-04-30') AND (RegionDescription = 'Eastern')  GROUP BY c.CategoryName)  SELECT  \*  , 100.0 \* sum\_sales / (SELECT SUM(sum\_sales) FROM category\_sales) AS '%\_sales'  , 100.0 \* SUM(sum\_sales) OVER(ORDER BY sum\_sales DESC ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) / (SELECT SUM(sum\_sales) FROM category\_sales) as '%\_cumsum\_sales'  FROM category\_sales  ; |
|  |
|  |
|  |
|  |

Table 1.2. m

|  |
| --- |
| Finding 1.2.l |
| WITH product\_sales AS  (SELECT  p.ProductName  , SUM(od.UnitPrice\*od.Quantity) as 'sum\_sales'  FROM Orders o  JOIN [Order Details] od ON o.OrderID = od.OrderID  JOIN Products p ON od.ProductID = p.ProductID  JOIN Categories c ON p.CategoryID = c.CategoryID  JOIN EmployeeTerritories et ON et.EmployeeID = o.EmployeeID  JOIN Territories t ON et.TerritoryID = t.TerritoryID  JOIN Region r ON r.RegionID = t.RegionID  WHERE (o.OrderDate BETWEEN '1997-11-01' AND '1998-04-30') AND (RegionDescription = 'Eastern')  GROUP BY p.ProductName)  SELECT  \*  , 100.0 \* sum\_sales / (SELECT SUM(sum\_sales) FROM product\_sales) AS '%\_sales'  , 100.0 \* SUM(sum\_sales) OVER(ORDER BY sum\_sales DESC ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) / (SELECT SUM(sum\_sales) FROM product\_sales) as '%\_cumsum\_sales'  FROM product\_sales  ; |
|  |
|  |
|  |

## CUSTOMER ANALYSIS – RFM

### Background & Finding

In this case study, the store tries to assess the quality of their customers. By doing this analysis, we could obtain insights of overall customers based on their segment. The quality of the customers is defined by the RFM metric (Recency, Frequency, Monetary). Based on the RFM, we could group the customers into different segments. This approach is useful to gain information of how ‘good’ the quality of our customer & help to identified what kinds of marketing approach to be applied to partifular customers segment.

Based on the analysis, we found some meaningful insights below:

1. The proportion & average monetary of each segment are shown in the Table 2.2. a
2. Based on the finding in the a, we could prioritize our marketing strategies on the champion & loyal customers segments because of two reasons: the proportion of customer from these two segments only is more than 40% and the average monetary is the highest after the cant lose them segment.
3. After prioritizing the champion & loyal customers, we could shift to the next segment in the queue, i.e. cant lose them, need attention, at risk, and potential loyalist segment
4. Further analysis could be applied to deep dive into specific region and recent time period, so that the marketing strategies could be more precise & on point.

### The Process (SQL Query, Flowchart, Visualization)

Table 2.2. a

|  |
| --- |
| Finding 2.2.a |
| -- 1. rfm score for each customer id  WITH  base\_rfm\_score AS  (SELECT  o.CustomerID  , COUNT(DISTINCT o.OrderID) as 'frequency\_value'  , DATEDIFF(DAY, MAX(o.OrderDate), '1998-01-07') as 'recency\_value'  , SUM((1-od.Discount)\*(od.UnitPrice\*od.Quantity)) as 'monetary\_value'  , NTILE(5) OVER(ORDER BY COUNT(DISTINCT o.OrderID) ASC) as 'frequency\_score'  , NTILE(5) OVER(ORDER BY DATEDIFF(DAY, MAX(o.OrderDate), '1998-01-07') DESC) as 'recency\_score'  , NTILE(5) OVER(ORDER BY SUM((1-od.Discount)\*(od.UnitPrice\*od.Quantity)) ASC) as 'monetary\_score'  FROM Orders o  JOIN [Order Details] od ON o.OrderID = od.OrderID  WHERE o.OrderDate BETWEEN '1997-01-01' AND '1997-12-31'  GROUP BY o.CustomerID),  -- 2. from the rfm score, group each customer to rfm segment  rfm\_segment\_table AS  (SELECT  \*  , (monetary\_score + recency\_score + frequency\_score)/3 as 'rfm\_score'  , CASE WHEN (recency\_score = 5) AND ((frequency\_score=5) OR (frequency\_score=4)) THEN 'champion'  WHEN ((recency\_score = 3) OR (recency\_score = 4)) AND ((frequency\_score=5) OR (frequency\_score=4)) THEN 'loyal customer'  WHEN ((recency\_score = 1) OR (recency\_score = 2)) AND (frequency\_score=5) THEN 'cant lose them'  WHEN ((recency\_score = 5) OR (recency\_score = 4)) AND ((frequency\_score=3) OR (frequency\_score=2)) THEN 'potential loyalist'  WHEN (recency\_score = 3) AND (frequency\_score=3) THEN 'need attention'  WHEN ((recency\_score = 1) OR (recency\_score = 2)) AND ((frequency\_score=3) OR (frequency\_score=4)) THEN 'at risk'  WHEN (recency\_score = 5) AND (frequency\_score=1) THEN 'new customer'  WHEN (recency\_score = 4) AND (frequency\_score=1) THEN 'promising'  WHEN (recency\_score = 3) AND ((frequency\_score=1) OR (frequency\_score=2)) THEN 'about to sleep'  ELSE 'hibernating' END AS 'rfm\_segment'  FROM base\_rfm\_score)  -- 3. for each customer segment, calcuate its proportion & avg monetary value  SELECT  rfm\_segment  , 100.0\*COUNT(CustomerID)/(SELECT COUNT(\*) FROM rfm\_segment\_table) AS '%\_segment'  , AVG(monetary\_value) AS 'avg\_monetary'  FROM rfm\_segment\_table  GROUP BY rfm\_segment  ORDER BY 100.0\*COUNT(CustomerID)/(SELECT COUNT(\*) FROM rfm\_segment\_table) DESC  ; |
|  |
|  |
|  |

## COHORT ANALYSIS – RETENTION (MONTHLY)

### Background & Finding

In this case study, the store tries to assess the monthly retention rate of different cohorts. This analysis is useful to obtain the information of customer engagement to the company’s products. This analysis was done in the full 1997 period.

Based on the analysis, we found some meaningful insights below:

1. The retention rate analysis is presented in the Table 3.2. a
2. This retention analysis could be combine to the Product Analysis to get more comprehensive view of business health condition.
3. The context should be employed more in this analysis to assess whether the result below is acceptable or not.
4. If the result is not acceptable, further analysis, e.g. root cause analysis, should be utilized to do counter action.

### The Process (SQL Query, Flowchart, Visualization)

Table 3.2. a

|  |
| --- |
| Finding 3.2.a |
| USE Northwind;  WITH  -- 1. for each customer id, find their first time order (month)  first\_buy AS(  SELECT  o.CustomerID  , DATEPART(MONTH, MIN(o.OrderDate)) first\_time\_buy  FROM Orders o  WHERE o.OrderDate BETWEEN '1997-01-01' AND '1997-12-31'  GROUP BY o.CustomerID),  -- 2. for each customer id, find all of their time order (month)  next\_purchase AS(  SELECT  o.CustomerID  , DATEPART(MONTH, o.OrderDate) - first\_time\_buy AS buy\_interval  FROM Orders o  JOIN first\_buy f ON o.CustomerID = f.CustomerID  WHERE o.OrderDate BETWEEN '1997-01-01' AND '1997-12-31'),  -- 3. for each first time buy (month), calculate the number of total distinct customer  initial\_user AS(  SELECT  first\_time\_buy  , COUNT(DISTINCT CustomerID) AS users  FROM first\_buy  GROUP BY first\_time\_buy),  -- 4. calculate the retention for each first time buy (month) & for each buy interval (month)  retention AS(  SELECT  f.first\_time\_buy  , buy\_interval  , COUNT(DISTINCT n.CustomerID) AS users\_transacting  FROM first\_buy f  JOIN next\_purchase n ON f.CustomerID = n.CustomerID  WHERE buy\_interval IS NOT NULL  GROUP BY f.first\_time\_buy, buy\_interval)  -- 5. put it all together, convert the retention into percentage  SELECT  r.first\_time\_buy,  i.users ,  r.buy\_interval,  r.users\_transacting,  100.0\*r.users\_transacting/i.users AS '%\_user\_transacting'  FROM retention r  LEFT JOIN initial\_user i ON r.first\_time\_buy = i.first\_time\_buy  ORDER BY r.first\_time\_buy, r.buy\_interval  ; |
|  |
|  |
|  |