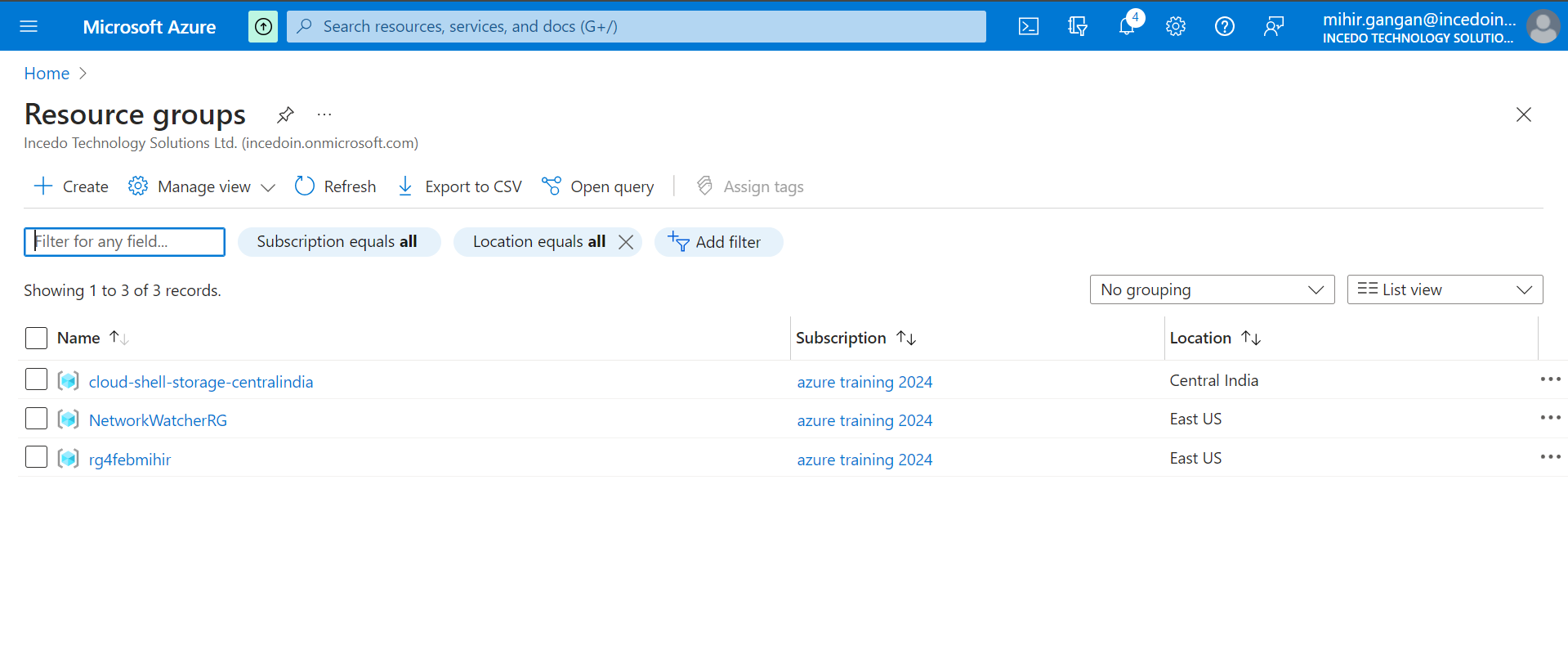
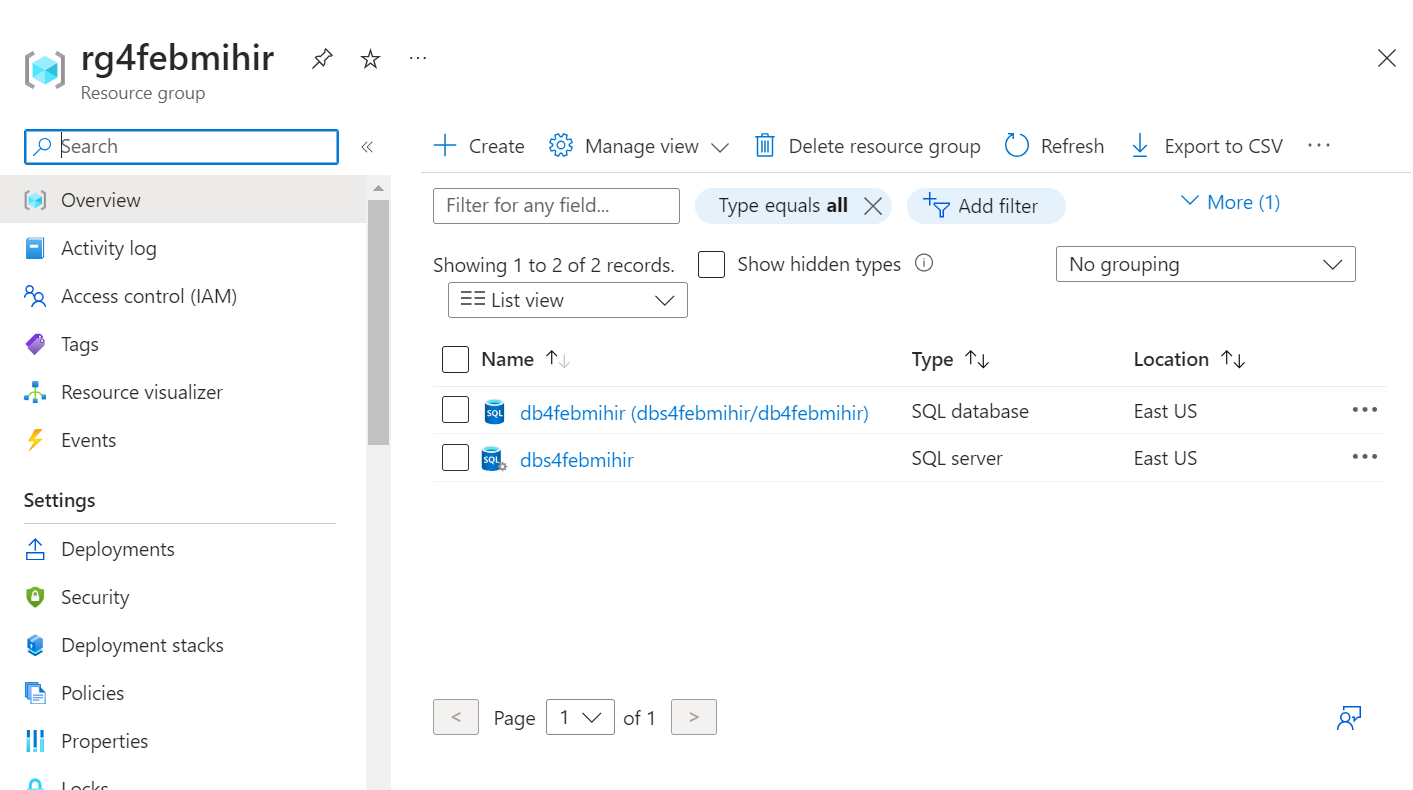
## 

## Module 1:

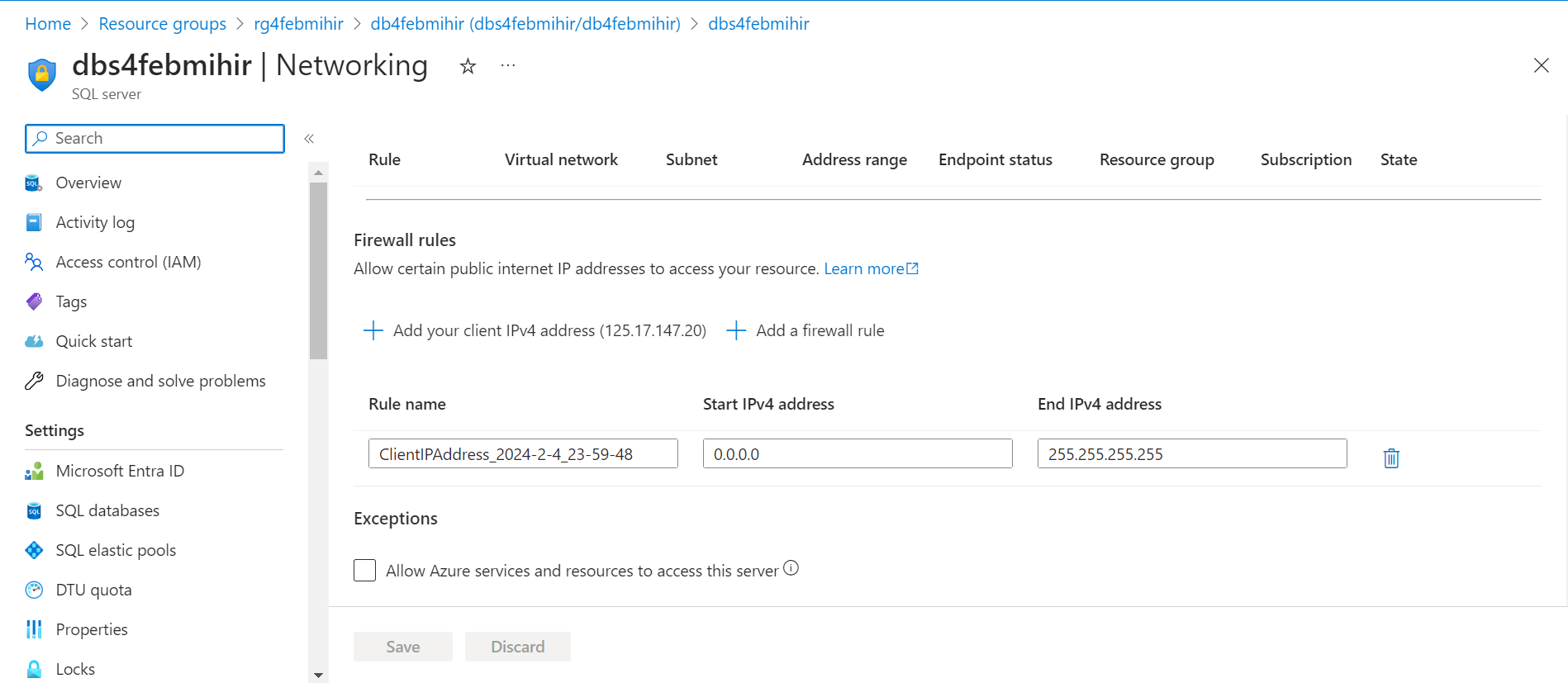
**Create resource group**



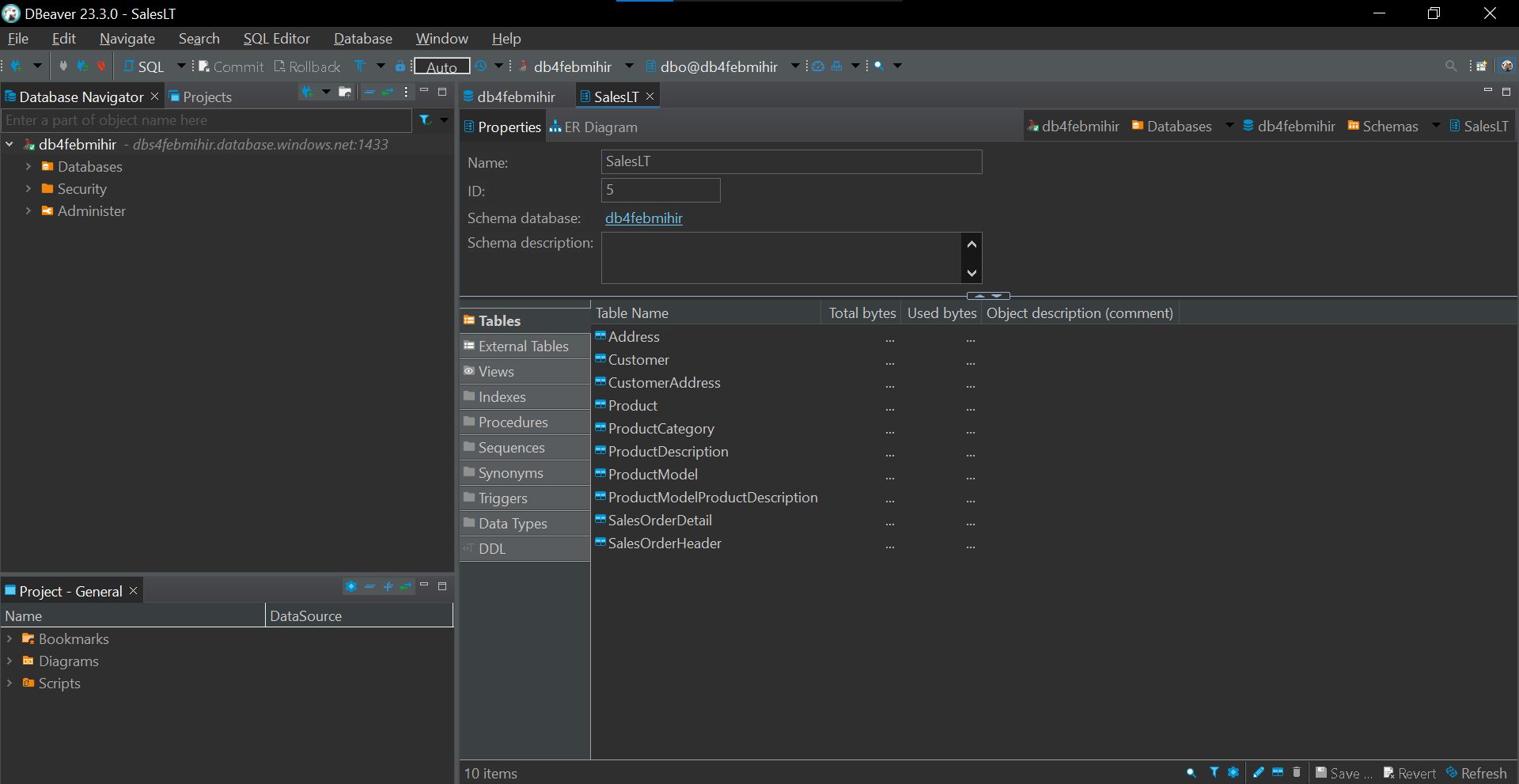
**Create sql server and sql database (sample adventureworks)**



**Change firewall settings to allow access to database from dbeaver**



**Open dbeaver and connect database**

****

## 

## Module 2 :

**1. Retrieve a list of customers along with their total order amounts**

SELECT

Customer.CustomerID,

Customer.FirstName,

Customer.LastName,

SUM(SalesOrderHeader.TotalDue) AS TotalOrderAmount

FROM

SalesLT.Customer

JOIN

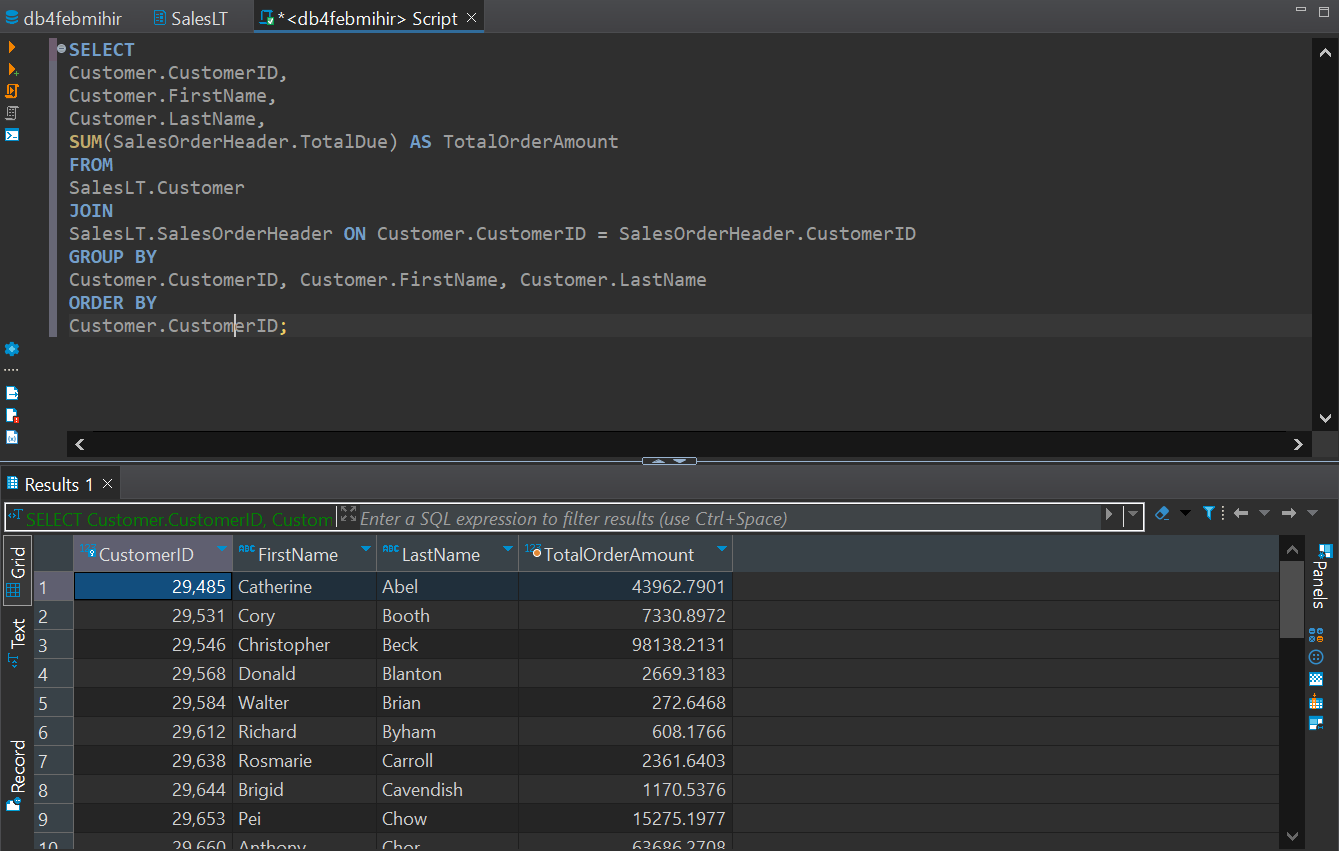
SalesLT.SalesOrderHeader ON Customer.CustomerID = SalesOrderHeader.CustomerID

GROUP BY

Customer.CustomerID, Customer.FirstName, Customer.LastName

ORDER BY

Customer.CustomerID;



**2. Display product information along with the number of units sold for each**

**product.**

SELECT

p.ProductID,

p.Name AS ProductName,

p.ProductNumber,

p.Color,

SUM(od.OrderQty) AS TotalUnitsSold

FROM

SalesLT.Product p

JOIN

SalesLT.SalesOrderDetail od ON p.ProductID = od.ProductID

JOIN

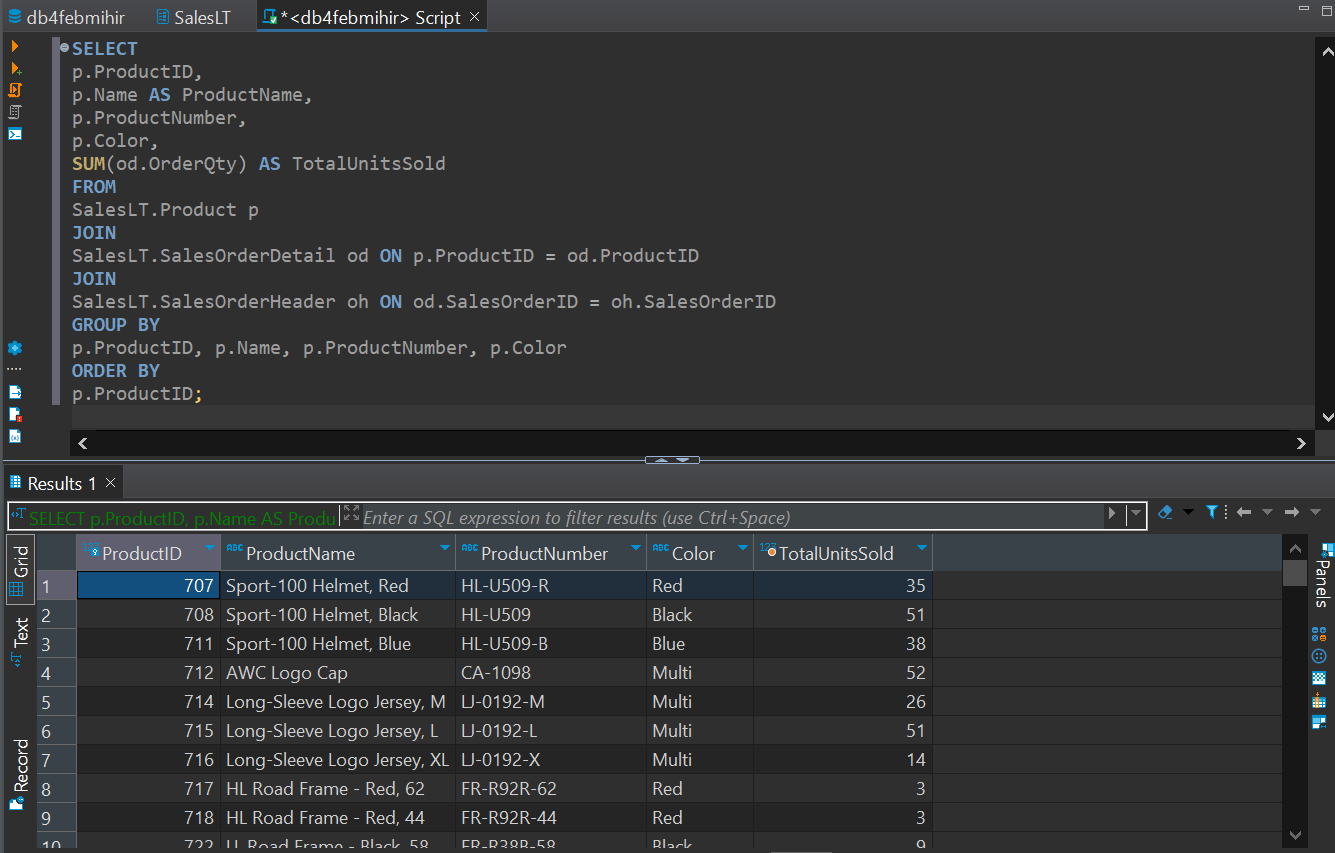
SalesLT.SalesOrderHeader oh ON od.SalesOrderID = oh.SalesOrderID

GROUP BY

p.ProductID, p.Name, p.ProductNumber, p.Color

ORDER BY

p.ProductID;



**3. Find employees who have the same manager** - Data insufficient

**4. List all customers who have never placed an order.**

SELECT

c.CustomerID,

c.FirstName,

c.LastName

FROM

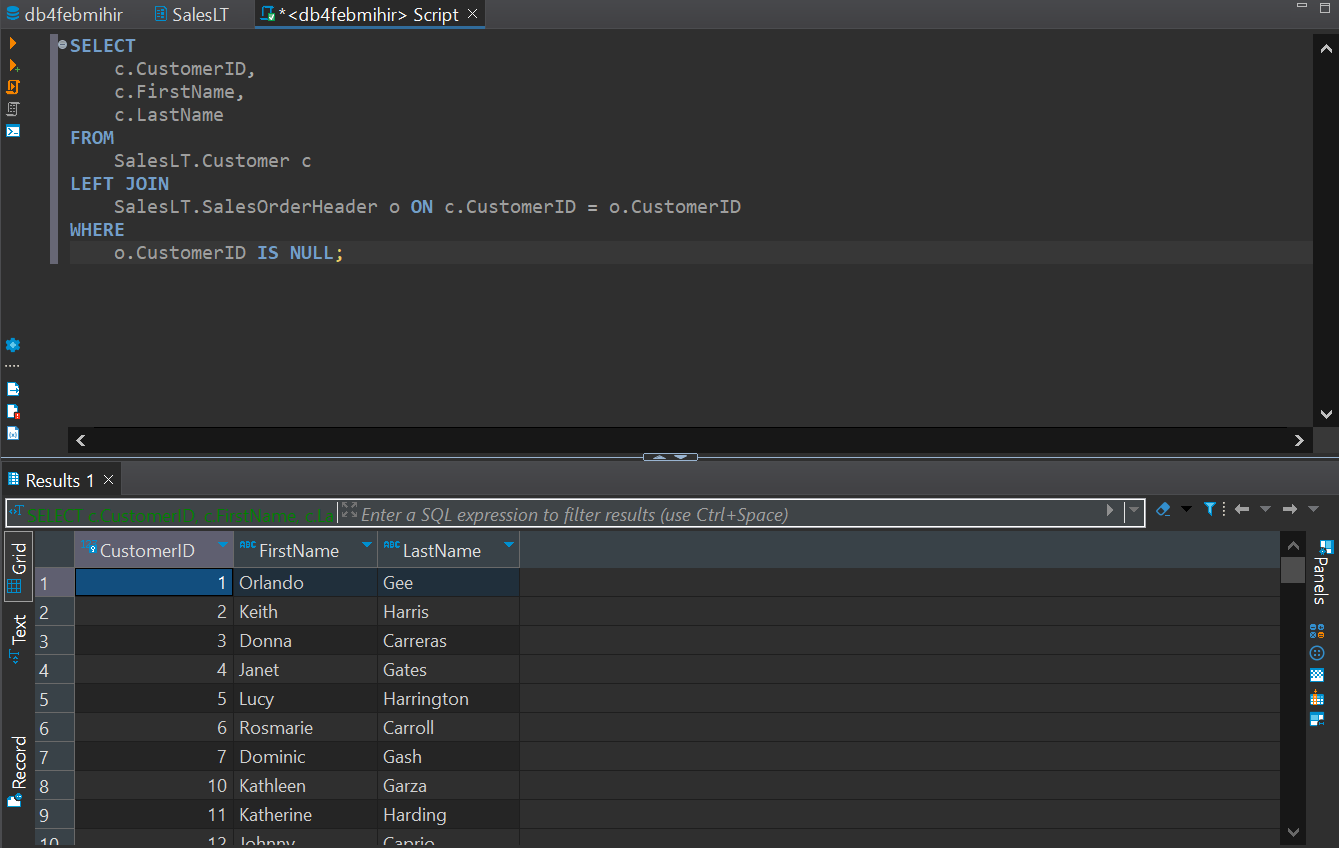
SalesLT.Customer c

LEFT JOIN

SalesLT.SalesOrderHeader o ON c.CustomerID = o.CustomerID

WHERE

o.CustomerID IS NULL;



**5. Retrieve the total sales amount for each product category.**

SELECT

pc.ProductCategoryID,

pc.Name AS CategoryName,

SUM(od.OrderQty \* od.UnitPrice) AS TotalsalesAmount

FROM

SalesLT.ProductCategory pc

JOIN

SalesLT.Product p ON pc.ProductCategoryID = p.ProductCategoryID

JOIN

SalesLT.SalesOrderDetail od ON p.ProductID = od.ProductID

JOIN

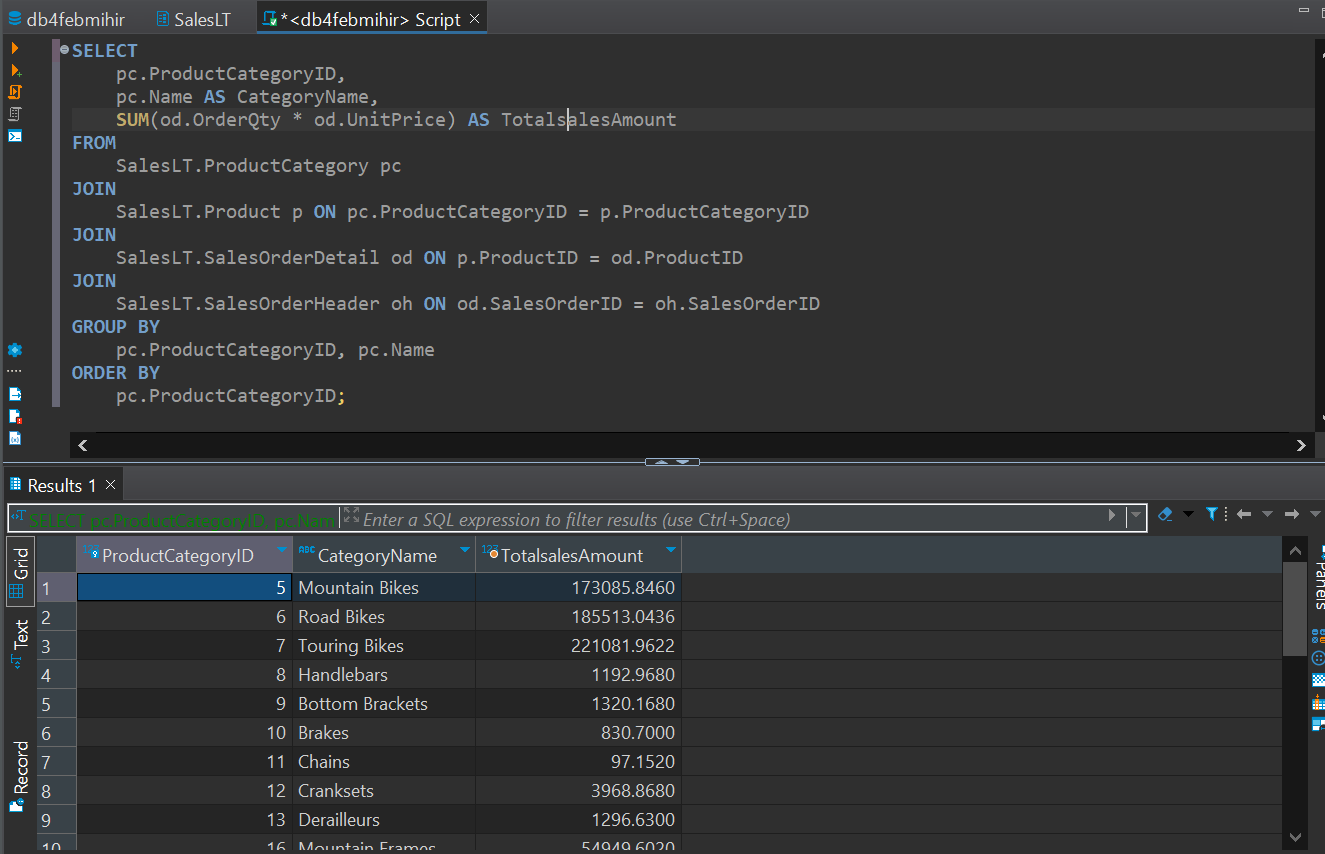
SalesLT.SalesOrderHeader oh ON od.SalesOrderID = oh.SalesOrderID

GROUP BY

pc.ProductCategoryID, pc.Name

ORDER BY

pc.ProductCategoryID;



**6. Display the names of employees and their direct managers** - Data insufficient.

**7. Show the order details with product names for a specific customer**

SELECT

oh.SalesOrderID,

od.ProductID,

p.Name AS ProductName,

od.OrderQty,

od.UnitPrice,

od.LineTotal

FROM

SalesLT.Customer c

JOIN

SalesLT.SalesOrderHeader oh ON c.CustomerID = oh.CustomerID

JOIN

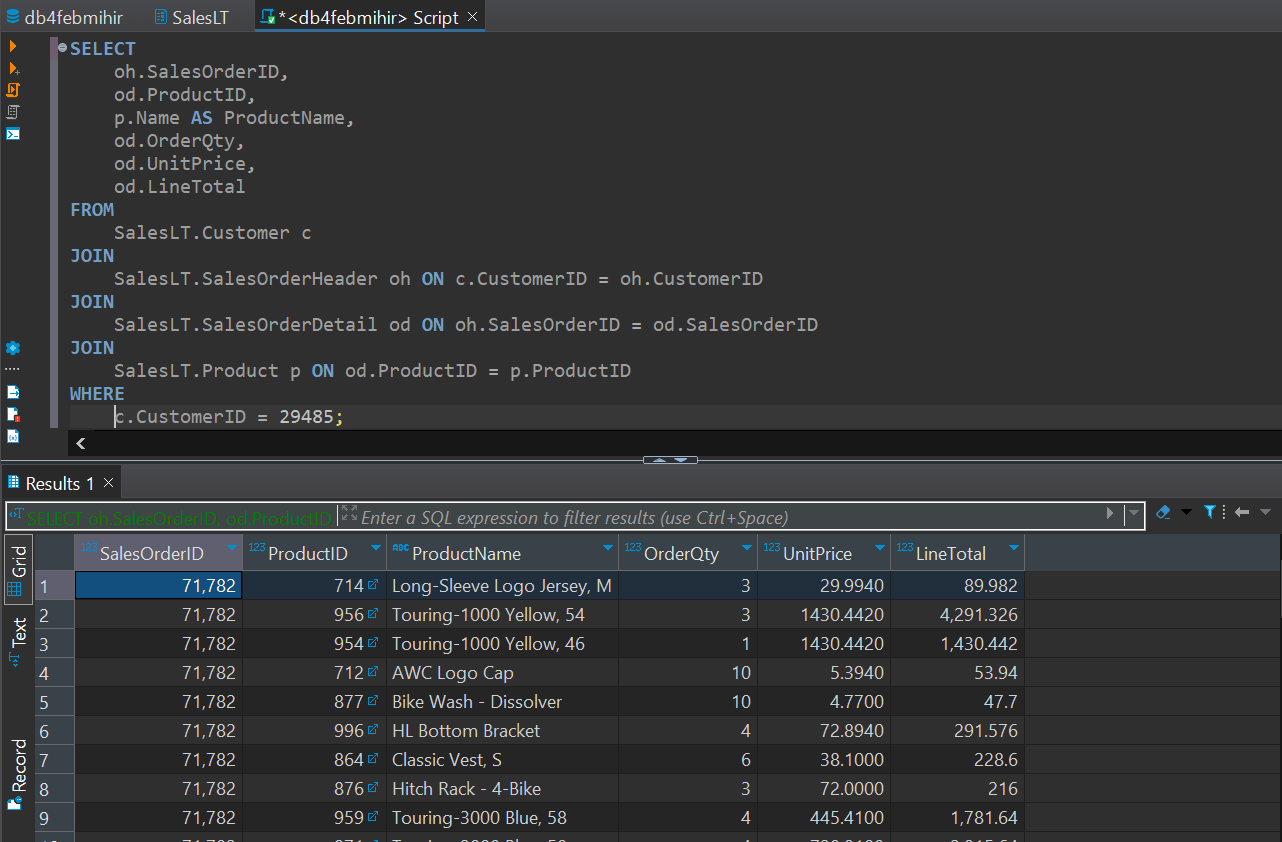
SalesLT.SalesOrderDetail od ON oh.SalesOrderID = od.SalesOrderID

JOIN

SalesLT.Product p ON od.ProductID = p.ProductID

WHERE

c.CustomerID = 29485;



**8. List customers who have made purchases in the last 30 days.**

SELECT DISTINCT

c.CustomerID,

c.FirstName,

c.LastName

FROM

SalesLT.Customer c

JOIN

SalesLT.SalesOrderHeader oh ON c.CustomerID = oh.CustomerID

WHERE

oh.OrderDate >= DATEADD(day, -30, GETDATE());

**9. Find employees who do not have any direct reports** - Data insufficient

**10. Retrieve all products along with their average selling prices**

SELECT

p.ProductID,

p.Name AS ProductName,

AVG(od.UnitPrice) AS AverageSellingPrice

FROM

SalesLT.Product p

JOIN

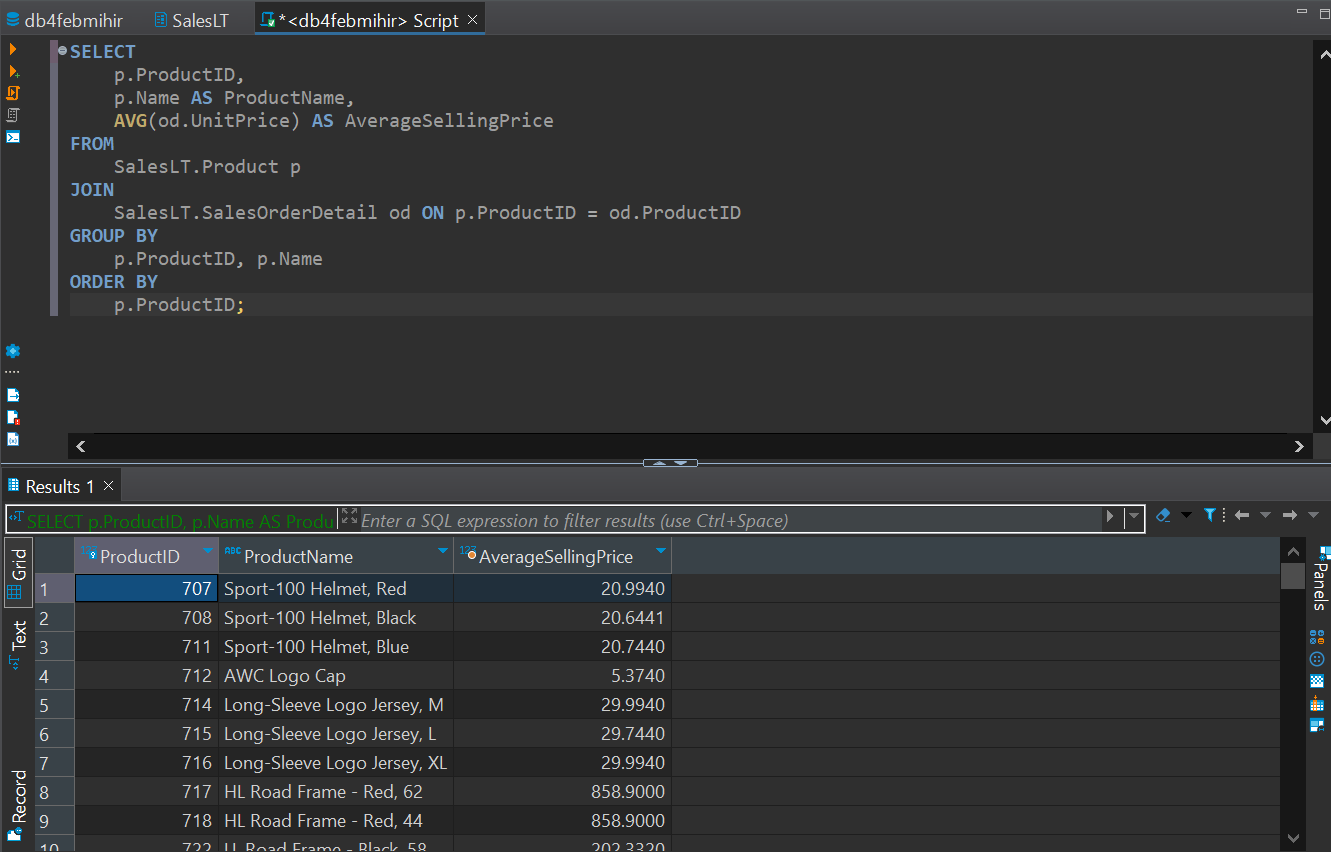
SalesLT.SalesOrderDetail od ON p.ProductID = od.ProductID

GROUP BY

p.ProductID, p.Name

ORDER BY

p.ProductID;



**11. Find the order with the highest total amount.**

SELECT TOP 1

oh.SalesOrderID,

oh.OrderDate,

SUM(od.LineTotal) AS TotalAmount

FROM

SalesLT.SalesOrderHeader oh

JOIN

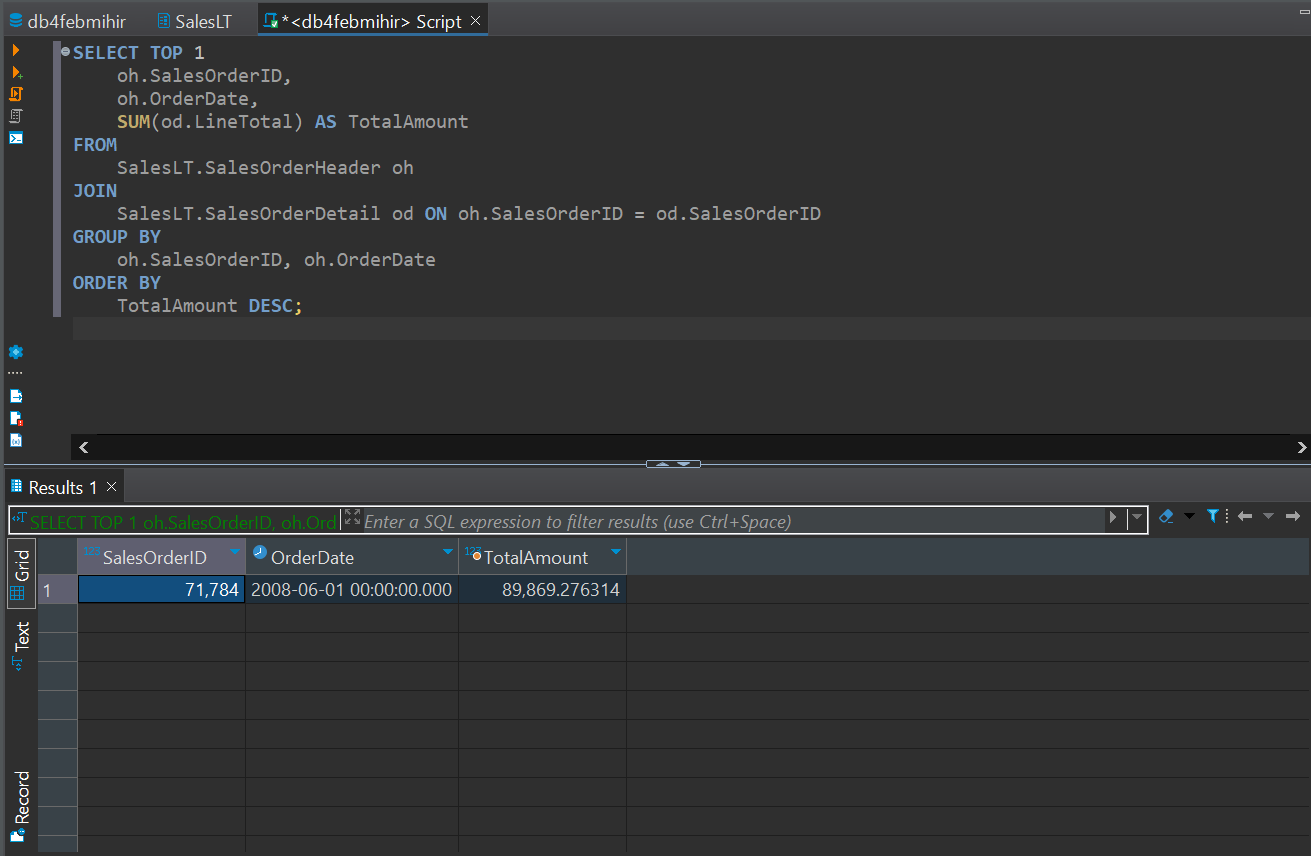
SalesLT.SalesOrderDetail od ON oh.SalesOrderID = od.SalesOrderID

GROUP BY

oh.SalesOrderID, oh.OrderDate

ORDER BY

TotalAmount DESC;



**12. Display customers who have placed orders with a total amount greater than the**

**average**

WITH CustomerOrderTotals AS (

SELECT

c.CustomerID,

c.FirstName,

c.LastName,

SUM(od.LineTotal) AS TotalAmount

FROM

SalesLT.Customer c

JOIN

SalesLT.SalesOrderHeader oh ON c.CustomerID = oh.CustomerID

JOIN

SalesLT.SalesOrderDetail od ON oh.SalesOrderID = od.SalesOrderID

GROUP BY

c.CustomerID, c.FirstName, c.LastName

)

SELECT

CustomerID,

FirstName,

LastName,

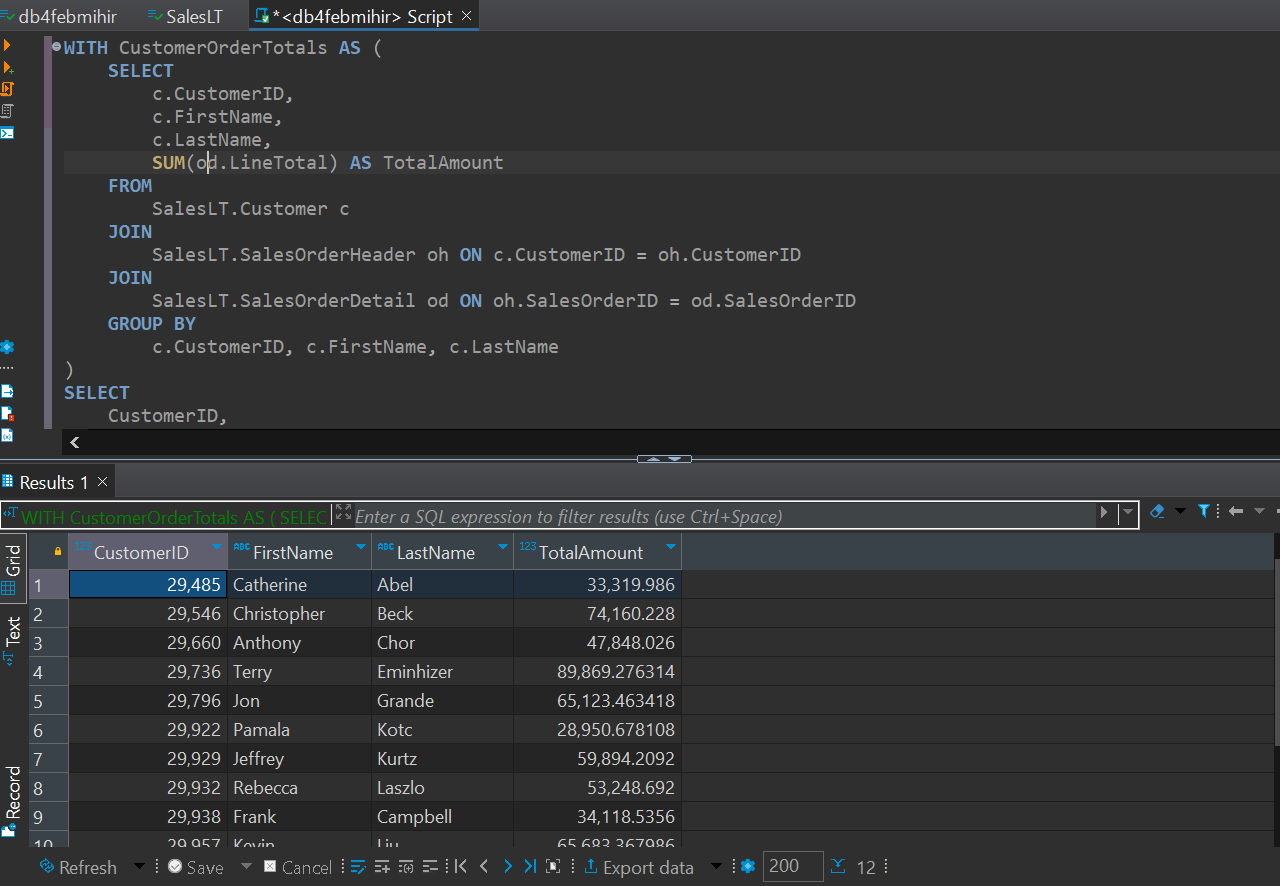
TotalAmount

FROM

CustomerOrderTotals

WHERE

TotalAmount > (SELECT AVG(TotalAmount) FROM CustomerOrderTotals);



**13. List products with prices higher than the average product price.**

WITH ProductPrices AS (

SELECT

ProductID,

Name AS ProductName,

ListPrice

FROM

SalesLT.Product

)

SELECT

ProductID,

ProductName,

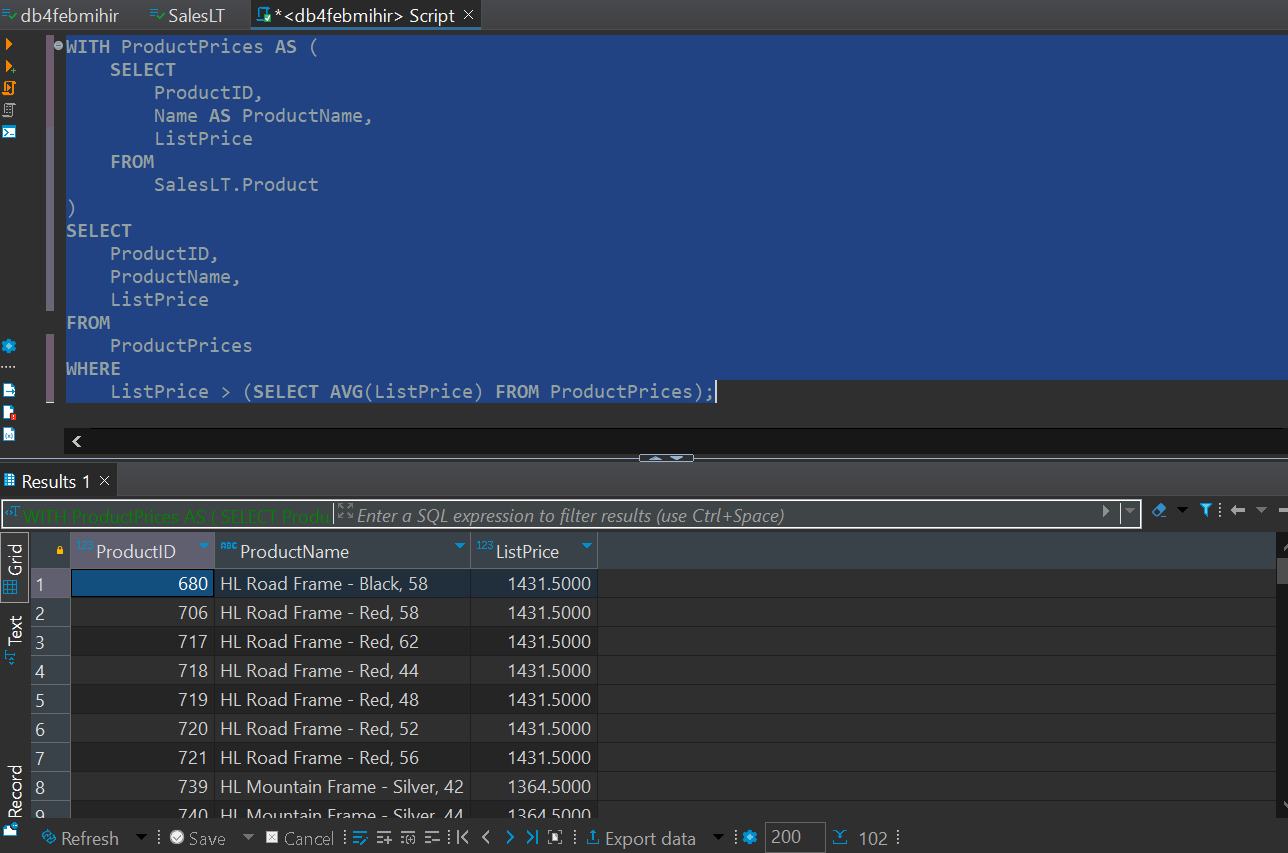
ListPrice

FROM

ProductPrices

WHERE

ListPrice > (SELECT AVG(ListPrice) FROM ProductPrices);



**14. Retrieve orders placed by employees who have a specific job title.** - Data insufficient

**15. Display customers who have placed orders for a specific product category**

SELECT

c.CustomerID,

c.FirstName,

c.LastName

FROM

SalesLT.Customer c

JOIN

SalesLT.SalesOrderHeader oh ON c.CustomerID = oh.CustomerID

JOIN

SalesLT.SalesOrderDetail od ON oh.SalesOrderID = od.SalesOrderID

JOIN

SalesLT.Product p ON od.ProductID = p.ProductID

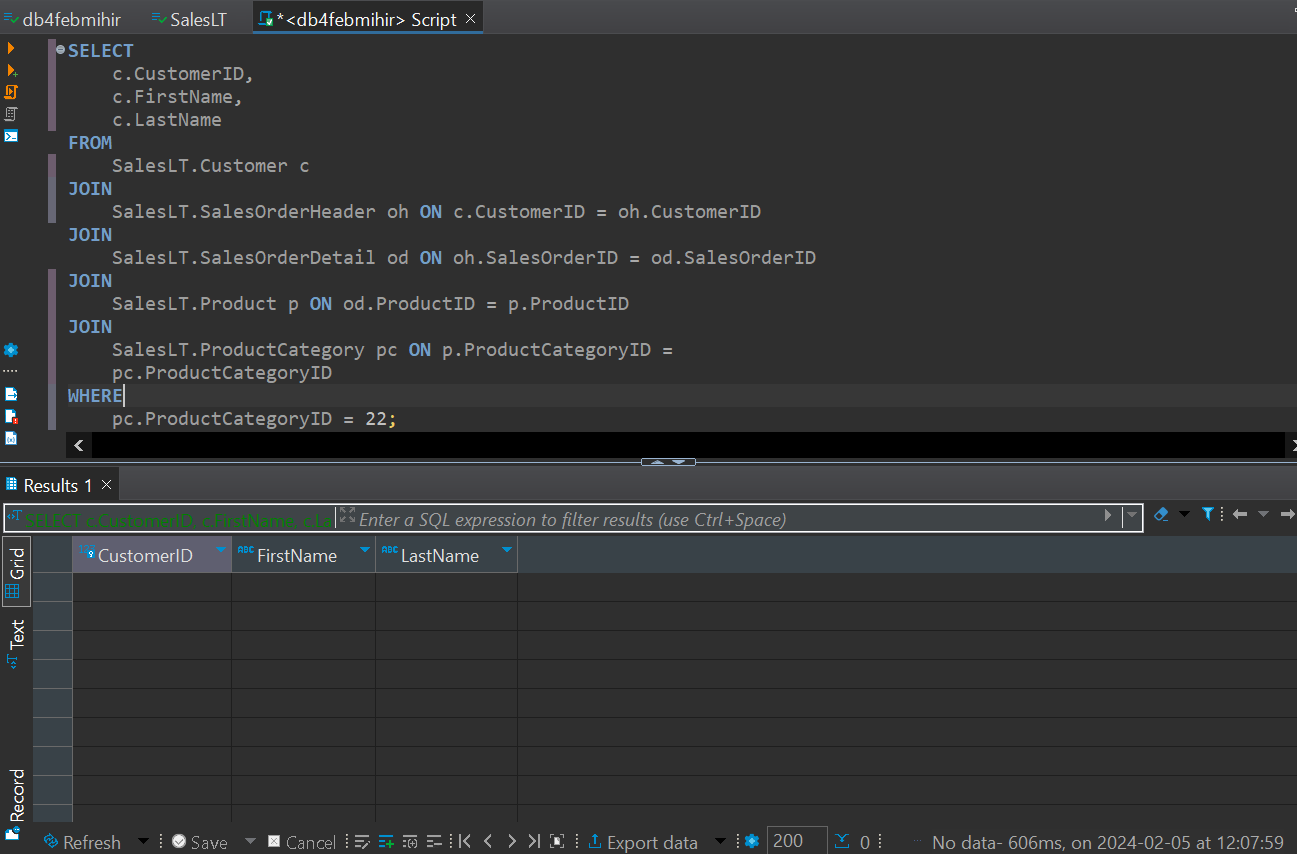
JOIN

SalesLT.ProductCategory pc ON p.ProductCategoryID =

pc.ProductCategoryID

WHERE

pc.ProductCategoryID = 22;



**16. Find employees with salaries greater than the average salary in their department** - Data insufficient

**17. List customers who have placed orders before a specific date.**

SELECT DISTINCT

c.CustomerID,

c.FirstName,

c.LastName

FROM

SalesLT.Customer c

JOIN

SalesLT.SalesOrderHeader oh ON c.CustomerID = oh.CustomerID

WHERE

oh.OrderDate < 2008-07-02

18.Retrieve the order with the highest quantity of a specific product.

SELECT TOP 1

od.SalesOrderID,

od.ProductID,

p.Name AS ProductName,

SUM(od.OrderQty) AS TotalQuantity

FROM

SalesLT.SalesOrderDetail od

JOIN

SalesLT.Product p ON od.ProductID = p.ProductID

WHERE

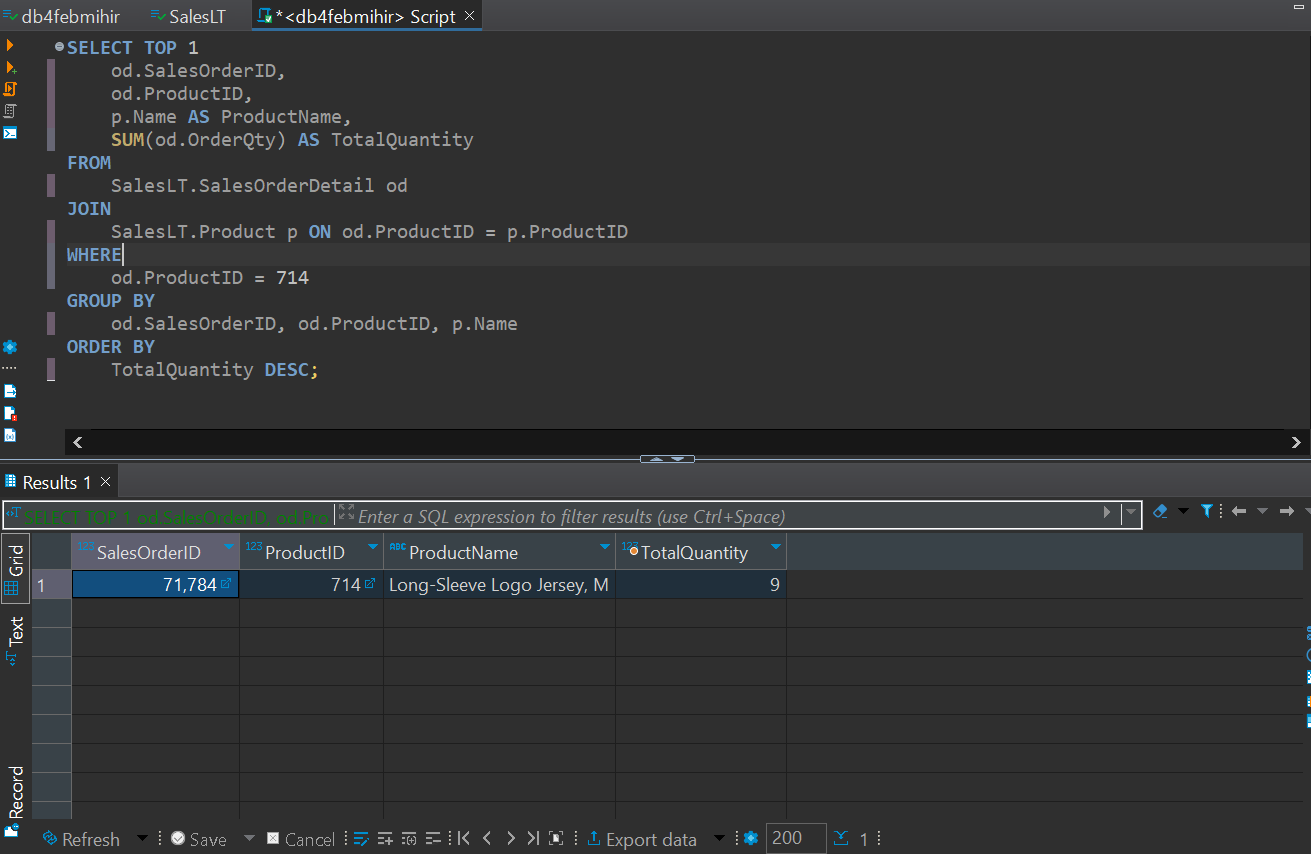
od.ProductID = 714

GROUP BY

od.SalesOrderID, od.ProductID, p.Name

ORDER BY

TotalQuantity DESC;



**19. Display products with prices lower than the lowest product price in a specific category.**

WITH ProductPrices AS (

SELECT

p.ProductID,

p.Name AS ProductName,

p.ListPrice,

pc.ProductCategoryID

FROM

SalesLT.Product p

JOIN

SalesLT.ProductCategory pc ON p.ProductCategoryID =

pc.ProductCategoryID

WHERE

pc.ProductCategoryID = 11

)

SELECT

ProductID,

ProductName,

ListPrice

FROM

ProductPrices

WHERE

ListPrice < (SELECT MIN(ListPrice) FROM ProductPrices);

**20. Find employees who have the same job title as their manager** - Data insufficient

**21. Combine results from two queries to get a list of unique customer and employee names** - Data insufficient

**22. Retrieve product names that are common in two different product categories.**

SELECT

p.Name AS ProductName

FROM

SalesLT.Product p

JOIN

SalesLT.ProductCategory pc1 ON p.ProductCategoryID = pc1.ProductCategoryID

JOIN

SalesLT.ProductCategory pc2 ON p.ProductCategoryID = pc2.ProductCategoryID

WHERE

pc1.ProductCategoryID <> pc2.ProductCategoryID;

**23. Display the names of employees and customers in a single result set** - Data insufficient

**24. List products that are in stock or have been discontinued** - Data insufficient

**25. Combine the results of two queries to find unique products ordered by a specific customer**

SELECT DISTINCT

p.ProductID,

p.Name AS ProductName

FROM

SalesLT.Product p

JOIN

SalesLT.SalesOrderDetail od ON p.ProductID = od.ProductID

JOIN

SalesLT.SalesOrderHeader oh ON od.SalesOrderID = oh.SalesOrderID

WHERE

oh.CustomerID = 29485

UNION

SELECT DISTINCT

p.ProductID,

p.Name AS ProductName

FROM

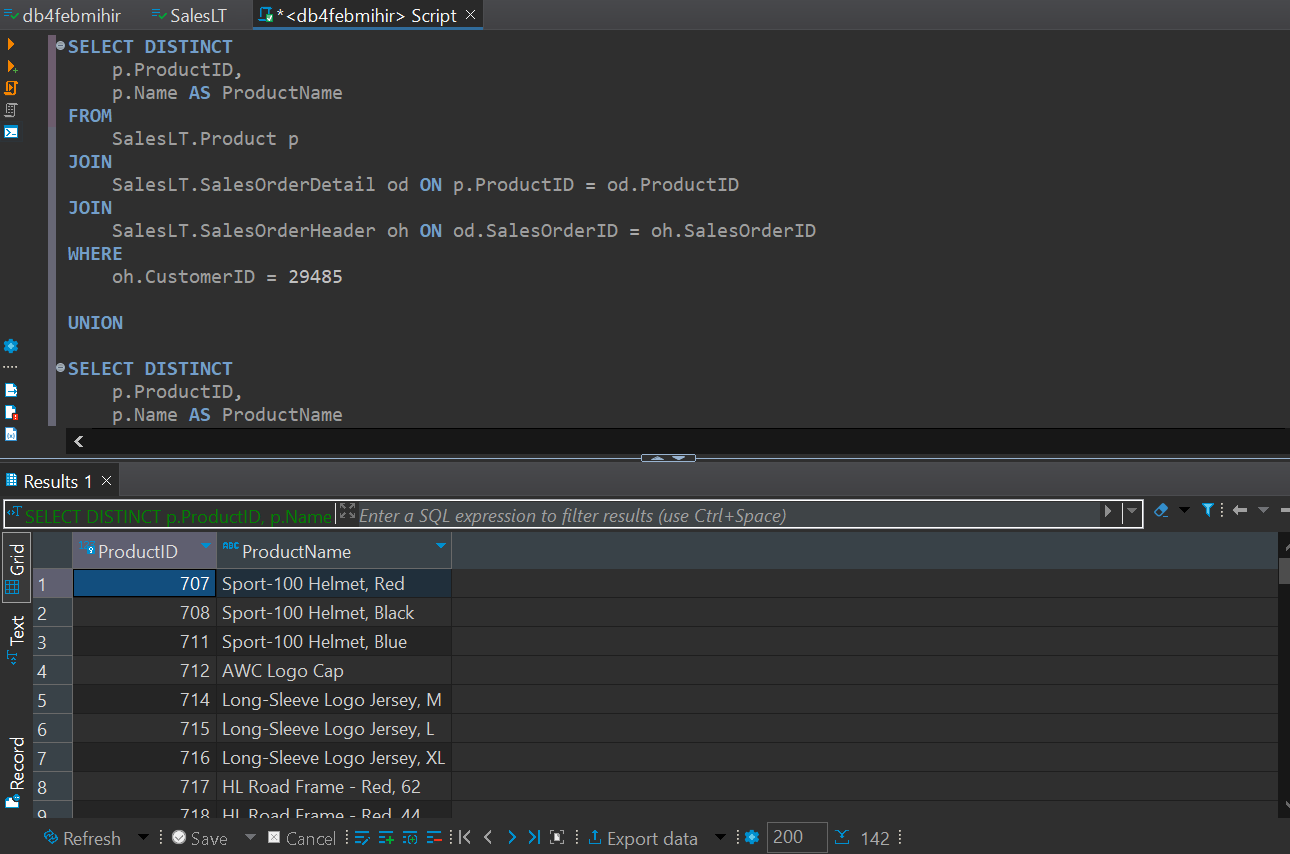
SalesLT.Product p

JOIN

SalesLT.SalesOrderDetail od ON p.ProductID = od.ProductID

ORDER BY

ProductID;



**26.Retrieve orders placed by customers and employees in a single result set** - Data insufficient

**27.Display products that are either in a specific category or have a specific safety stock level** - Data insufficient

**28.List customers who have placed orders and employees who have direct reports in a single result set** - Data insufficient

**29.Retrieve products that are in stock in one location and out of stock in another** - Data insufficient

**30.Combine information about employees who are managers and employees who have managers** - Data insufficient

**INTERMEDIATE**

**31. Retrieve a list of customers along with the names of the products they have purchased.**

SELECT

c.CustomerID,

c.FirstName,

c.LastName,

p.Name AS ProductName

FROM

SalesLT.Customer c

JOIN

SalesLT.SalesOrderHeader oh ON c.CustomerID = oh.CustomerID

JOIN

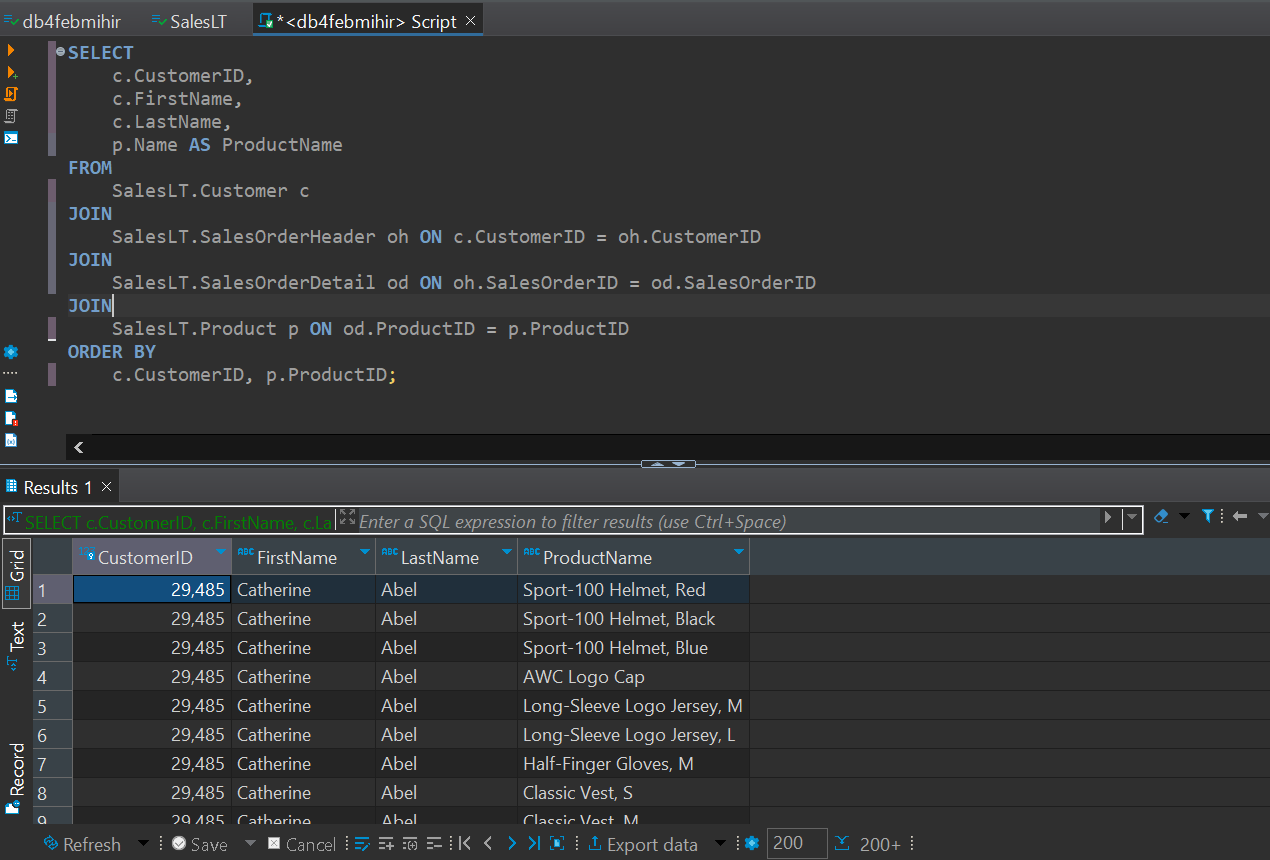
SalesLT.SalesOrderDetail od ON oh.SalesOrderID = od.SalesOrderID

JOIN

SalesLT.Product p ON od.ProductID = p.ProductID

ORDER BY

c.CustomerID, p.ProductID;



**32. Display employees who have the same manager, including indirect reports** - Data insufficient

**33. Find orders with multiple products and display the product names.**

SELECT

oh.SalesOrderID,

COUNT(DISTINCT od.ProductID) AS NumberOfProducts,

STRING\_AGG(p.Name, ', ') AS ProductNames

FROM

SalesLT.SalesOrderHeader oh

JOIN

SalesLT.SalesOrderDetail od ON oh.SalesOrderID = od.SalesOrderID

JOIN

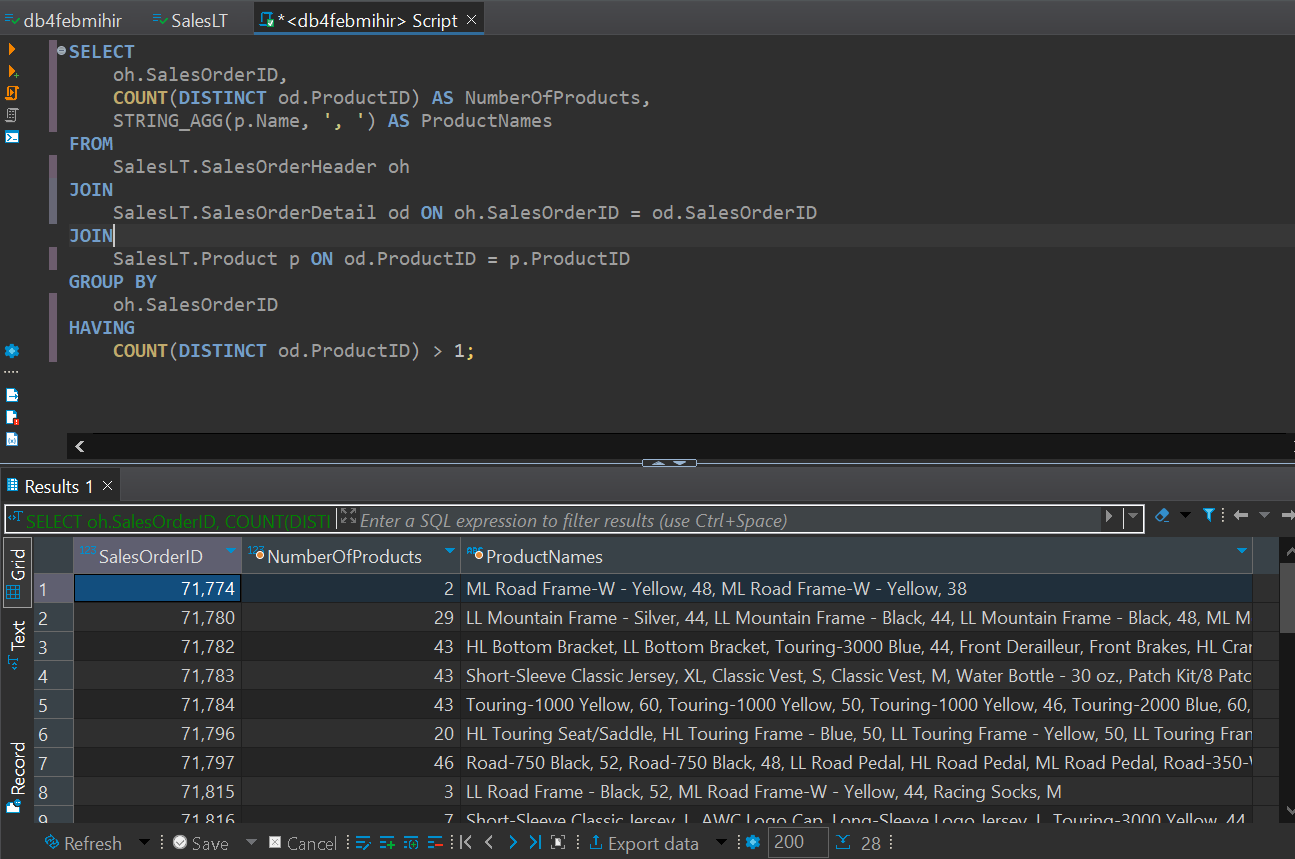
SalesLT.Product p ON od.ProductID = p.ProductID

GROUP BY

oh.SalesOrderID

HAVING

COUNT(DISTINCT od.ProductID) > 1;



**34. List customers along with the names of the salespeople who handled their orders -** Data insufficient

**35. Retrieve a list of products along with the names of suppliers -** Data insufficient

**36. Display customers who have placed orders and the products they have purchased, including product details**.

SELECT

c.CustomerID,

c.FirstName,

c.LastName,

oh.SalesOrderID,

p.ProductID,

p.Name AS ProductName,

od.OrderQty,

od.UnitPrice

FROM

SalesLT.Customer c

JOIN

SalesLT.SalesOrderHeader oh ON c.CustomerID = oh.CustomerID

JOIN

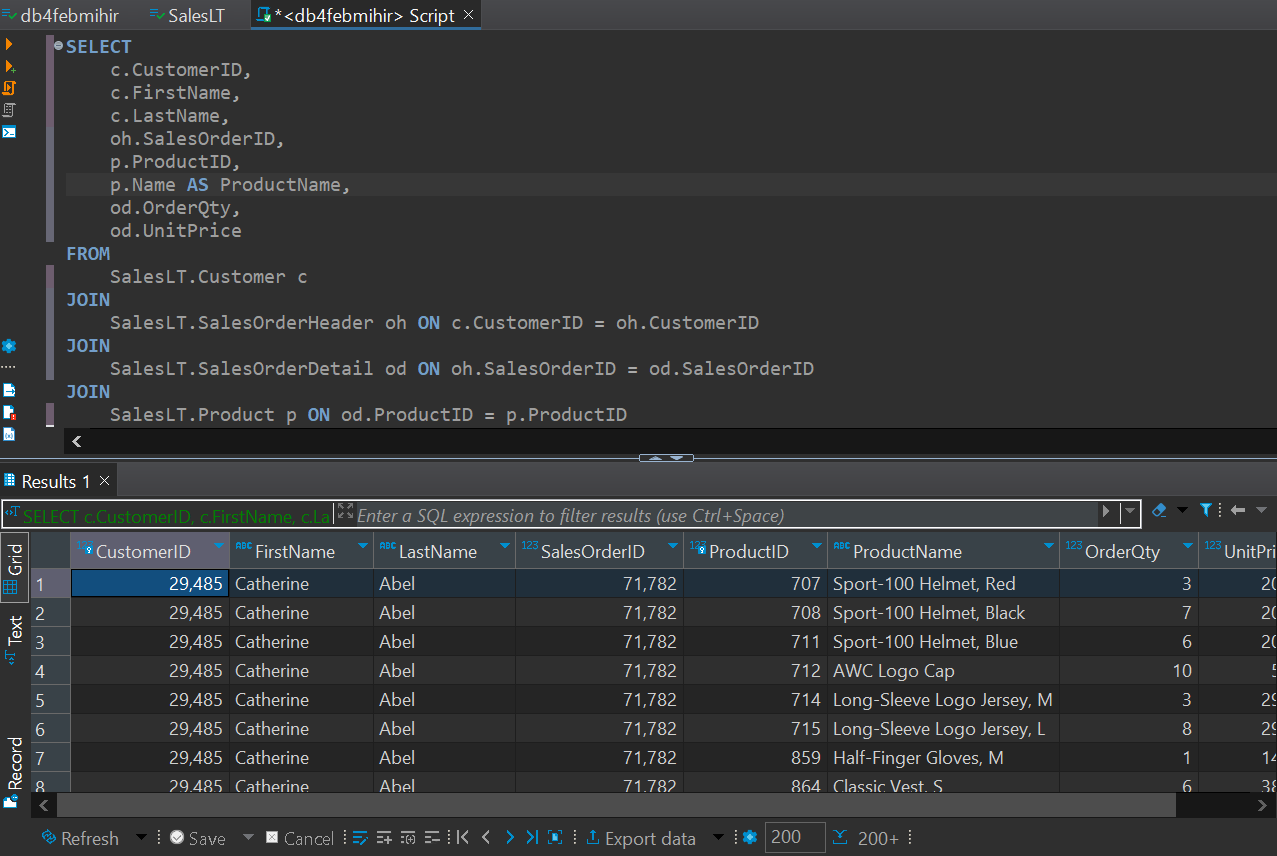
SalesLT.SalesOrderDetail od ON oh.SalesOrderID = od.SalesOrderID

JOIN

SalesLT.Product p ON od.ProductID = p.ProductID

ORDER BY

c.CustomerID, oh.SalesOrderID, p.ProductID;



**37. Find orders where multiple employees were involved, showing the employee names** -Data insufficient

**38. List products that have similar names but belong to different categories -** Data insufficient

**39. Retrieve a list of employees along with their training courses and training dates** - Data insufficient

**40. Display customers who have placed orders and the total quantity of each product ordered.**

SELECT

c.CustomerID,

c.FirstName,

c.LastName,

p.ProductID,

p.Name AS ProductName,

SUM(od.OrderQty) AS TotalQuantity

FROM

SalesLT.Customer c

JOIN

SalesLT.SalesOrderHeader oh ON c.CustomerID = oh.CustomerID

JOIN

SalesLT.SalesOrderDetail od ON oh.SalesOrderID = od.SalesOrderID

JOIN

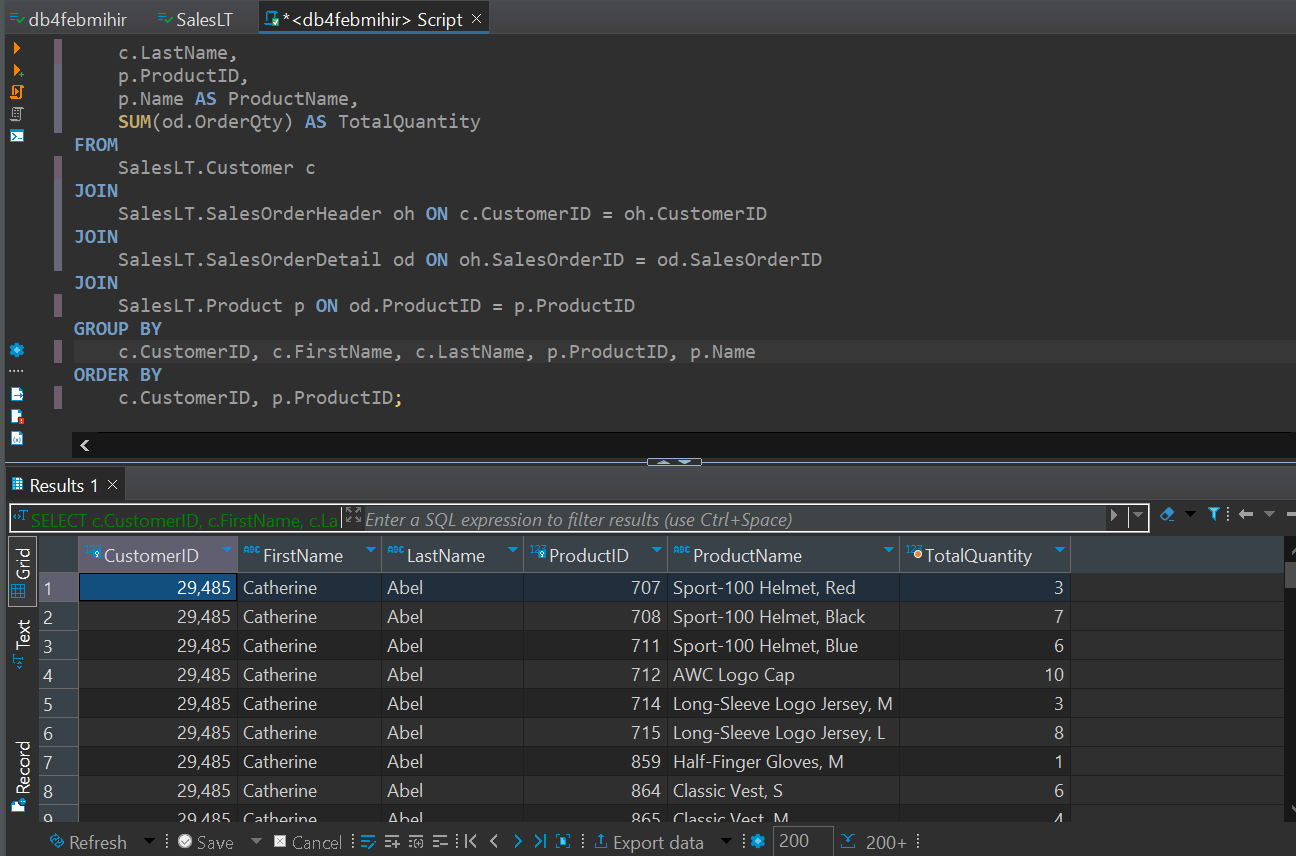
SalesLT.Product p ON od.ProductID = p.ProductID

GROUP BY

c.CustomerID, c.FirstName, c.LastName, p.ProductID, p.Name

ORDER BY

c.CustomerID, p.ProductID;



**41. Find customers who have made more purchases than the average number of purchases.**

WITH CustomerPurchaseCounts AS (

SELECT

c.CustomerID,

COUNT(DISTINCT oh.SalesOrderID) AS PurchaseCount

FROM

SalesLT.Customer c

JOIN

SalesLT.SalesOrderHeader oh ON c.CustomerID = oh.CustomerID

JOIN

SalesLT.SalesOrderDetail od ON oh.SalesOrderID = od.SalesOrderID

GROUP BY

c.CustomerID

)

SELECT

c.CustomerID,

c.FirstName,

c.LastName,

c.EmailAddress,

c.Phone,

c.CompanyName

FROM

SalesLT.Customer c

JOIN

(

SELECT

CustomerID

FROM

CustomerPurchaseCounts

WHERE

PurchaseCount > (SELECT AVG(PurchaseCount) FROM CustomerPurchaseCounts)

) pc ON c.CustomerID = pc.CustomerID;

**42. Display products that have been ordered more than the average number of times.**

WITH ProductOrderCounts AS (

SELECT

p.ProductID,

p.Name AS ProductName,

COUNT(od.SalesOrderID) AS OrderCount

FROM

SalesLT.Product p

JOIN

SalesLT.SalesOrderDetail od ON p.ProductID = od.ProductID

GROUP BY

p.ProductID, p.Name

)

SELECT

ProductID,

ProductName,

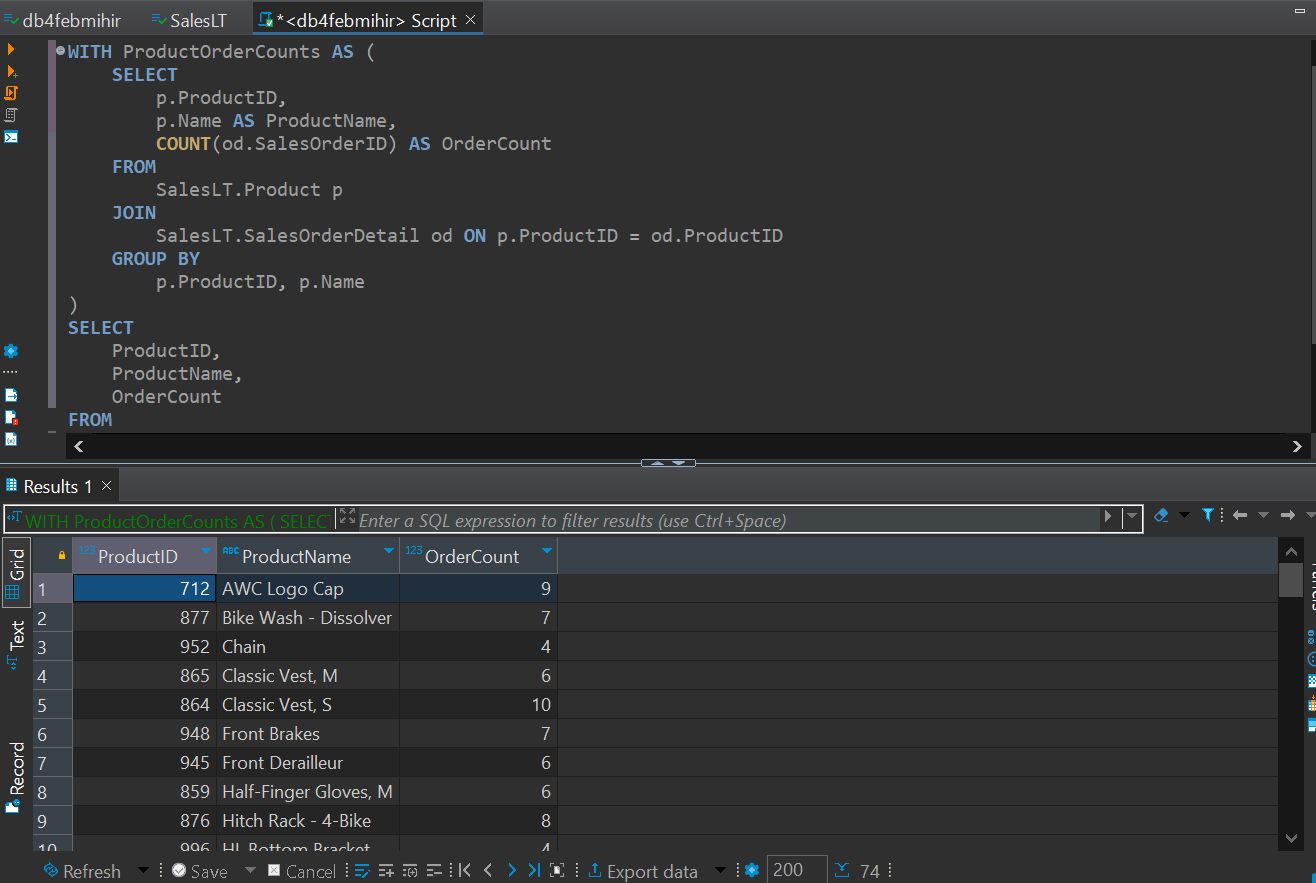
OrderCount

FROM

ProductOrderCounts

WHERE

OrderCount > (SELECT AVG(OrderCount) FROM ProductOrderCounts);



**43. Retrieve orders placed by employees who have completed a specific training course** - Data insufficient

**44. List employees who have a higher salary than at least one employee in another department** - Data insufficient

**45.Display products that have not been ordered in the last 60 days -** Data insufficient

**46. Find employees who have the same job title as the employee with the highest salary -** Data insufficient

**47. List customers who have placed orders with a total amount greater than the total amount of a specific order** - Data insufficient

**48. Retrieve products that have been ordered by customers with the same shipping address.**

WITH CustomerShippingAddresses AS (

SELECT

ca.CustomerID,

ca.AddressID,

a.AddressLine1,

a.AddressLine2,

a.City,

a.StateProvince,

a.CountryRegion,

a.PostalCode

FROM

SalesLT.CustomerAddress ca

JOIN

SalesLT.Address a ON ca.AddressID = a.AddressID

)

SELECT

p.ProductID,

p.Name AS ProductName,

od.SalesOrderID,

csa.CustomerID,

csa.AddressID,

csa.AddressLine1,

csa.AddressLine2,

csa.City,

csa.StateProvince,

csa.CountryRegion,

csa.PostalCode

FROM

SalesLT.Product p

JOIN

SalesLT.SalesOrderDetail od ON p.ProductID = od.ProductID

JOIN

SalesLT.SalesOrderHeader oh ON od.SalesOrderID = oh.SalesOrderID

JOIN

CustomerShippingAddresses csa ON oh.CustomerID = csa.CustomerID

WHERE

oh.ShipToAddressID IN (

SELECT

ShipToAddressID

FROM

SalesLT.SalesOrderHeader

GROUP BY

ShipToAddressID

HAVING

COUNT(DISTINCT CustomerID) > 1

);

**49. Display orders with quantities higher than the average quantity for a specific product.**

WITH AvgQuantityPerProduct AS (

SELECT

od.ProductID,

AVG(od.OrderQty) AS AvgQuantity

FROM

SalesLT.SalesOrderDetail od

GROUP BY

od.ProductID

)

SELECT

od.SalesOrderID,

od.ProductID,

p.Name AS ProductName,

od.OrderQty

FROM

SalesLT.SalesOrderDetail od

JOIN

SalesLT.Product p ON od.ProductID = p.ProductID

JOIN

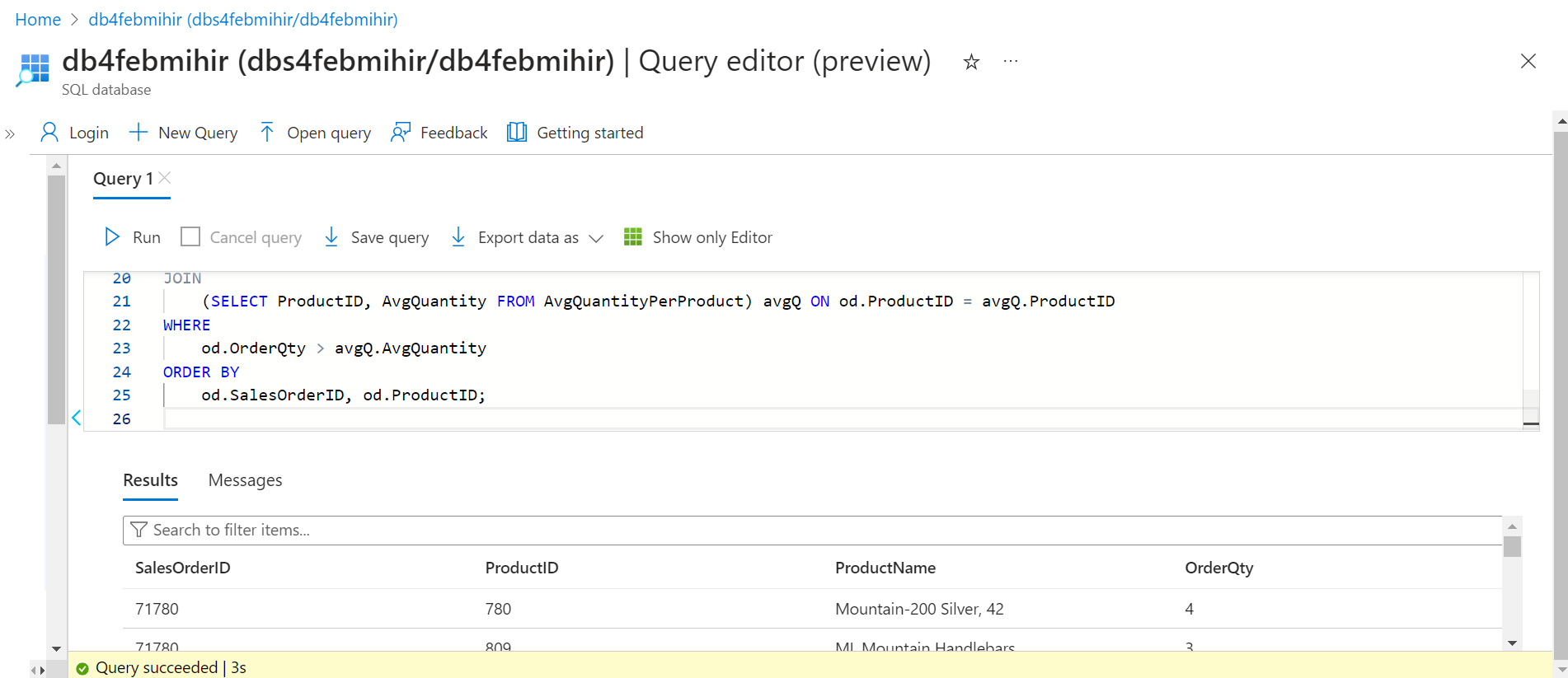
(SELECT ProductID, AvgQuantity FROM AvgQuantityPerProduct) avgQ ON od.ProductID = avgQ.ProductID

WHERE

od.OrderQty > avgQ.AvgQuantity

ORDER BY

od.SalesOrderID, od.ProductID;



**50. Find customers who have placed orders for products that have not been ordered by any other customer.**

WITH ProductsOrderedByCustomers AS (

SELECT DISTINCT

od.ProductID

FROM

SalesLT.SalesOrderDetail od

)

SELECT

c.CustomerID,

c.FirstName,

c.LastName,

oh.SalesOrderID,

p.ProductID,

p.Name AS ProductName

FROM

SalesLT.Customer c

JOIN

SalesLT.SalesOrderHeader oh ON c.CustomerID = oh.CustomerID

JOIN

SalesLT.SalesOrderDetail od ON oh.SalesOrderID = od.SalesOrderID

JOIN

SalesLT.Product p ON od.ProductID = p.ProductID

WHERE

p.ProductID NOT IN (SELECT ProductID FROM ProductsOrderedByCustomers)

ORDER BY

c.CustomerID, oh.SalesOrderID, p.ProductID;

