

Module 1:

Create resource group

Microsoft Azure

Search resources, services, and docs (G+/)

Home >

Resource groups

Incedo Technology Solutions Ltd. (incedo.onmicrosoft.com)

+ Create Manage view Refresh Export to CSV Open query Assign tags

Filter for any field... Subscription equals all Location equals all Add filter

Showing 1 to 3 of 3 records.

No grouping List view

Name	Subscription	Location
cloud-shell-storage-centralindia	azure training 2024	Central India
NetworkWatcherRG	azure training 2024	East US
rg4febmihir	azure training 2024	East US

Create sql server and sql database (sample adventureworks)

rg4febmihir

Resource group

Search

+ Create Manage view Delete resource group Refresh Export to CSV

Filter for any field... Type equals all Add filter More (1)

Showing 1 to 2 of 2 records. Show hidden types

No grouping


List view

Name	Type	Location
db4febmihir (dbs4febmihir/db4febmihir)	SQL database	East US
dbs4febmihir	SQL server	East US

Page 1 of 1

Change firewall settings to allow access to database from dbeaver

Home > Resource groups > rg4febmihir > db4febmihir (dbs4febmihir/db4febmihir) > dbs4febmihir

**dbs4febmihir** | Networking

SQL server

Search

Overview

Activity log

Access control (IAM)

Tags

Quick start

Diagnose and solve problems

Settings

Microsoft Entra ID

SQL databases

SQL elastic pools

DTU quota

Properties

Locks

Rule

Virtual network

Subnet

Address range

Endpoint status

Resource group

Subscription

State

Firewall rules

Allow certain public internet IP addresses to access your resource. [Learn more](#)

+ Add your client IPv4 address (125.17.147.20) + Add a firewall rule

Rule name	Start IPv4 address	End IPv4 address
ClientIPAddress_2024-2-4_23-59-48	0.0.0.0	255.255.255.255

Exceptions

☐ Allow Azure services and resources to access this server

Save

Discard

Open dbeaver and connect database

DBeaver 23.3.0 - SalesLT

File Edit Navigate Search SQL Editor Database Window Help

Auto db4febmihir dbo@db4febmihir

Database Navigator x Projects x SalesLT x

Enter a part of object name here

db4febmihir - dbs4febmihir.database.windows.net:1433

Databases

Security

Administer

Project - General x

Name DataSource

Bookmarks

Diagrams

Scripts

Properties ER Diagram

Name: SalesLT

ID: 5

Schema database: db4febmihir

Schema description:

Tables

Table Name Total bytes Used bytes Object description (comment)

Address

Customer

CustomerAddress

Product

ProductCategory

ProductDescription

ProductModel

ProductModelProductDescription

SalesOrderDetail

SalesOrderHeader

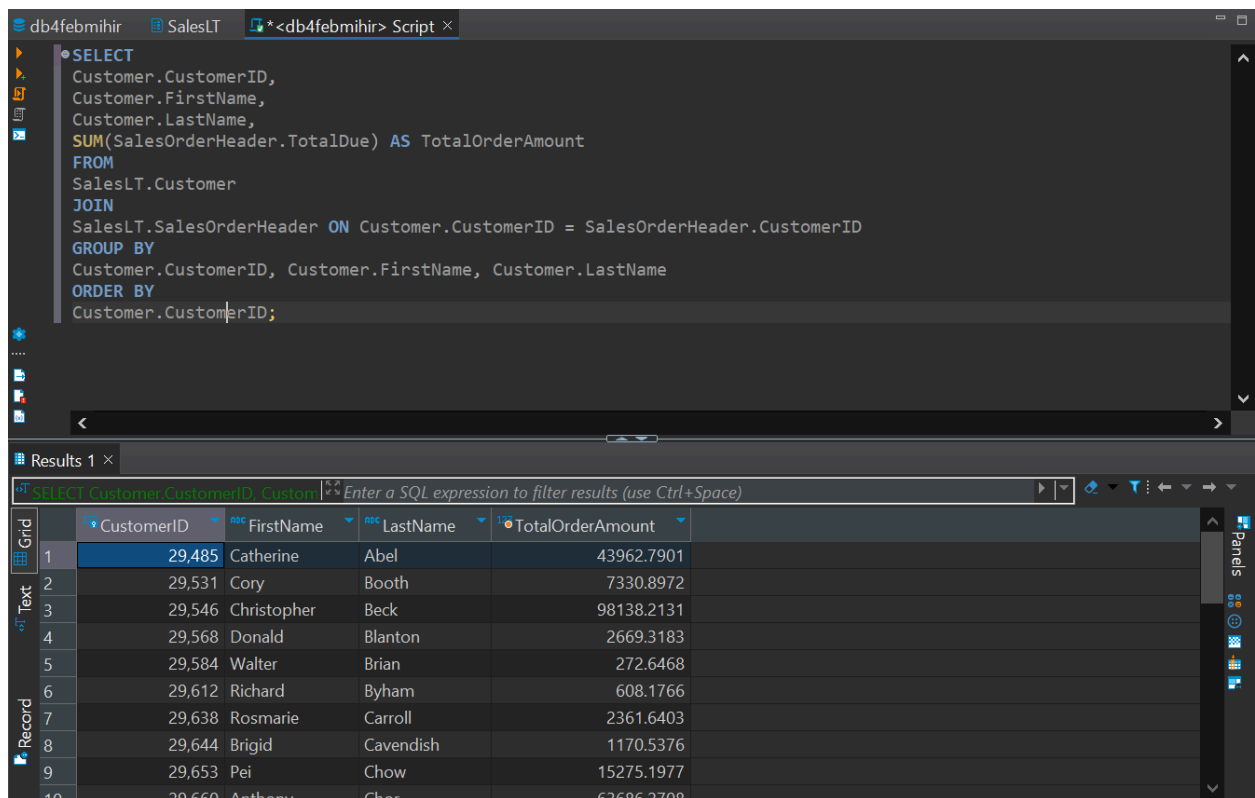
10 items

Save Revert Refresh

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;
```



The screenshot shows a SQL Server Enterprise Manager interface. The top pane displays a SQL query in a script editor. The bottom pane shows the results of the query in a grid format. The query is a SELECT statement that joins the SalesLT.Customer table with the SalesLT.SalesOrderHeader table, grouping by CustomerID, FirstName, and LastName, and ordering by CustomerID. The results grid shows 10 rows of data, including CustomerID, FirstName, LastName, and TotalOrderAmount.

	CustomerID	FirstName	LastName	TotalOrderAmount
1	29,485	Catherine	Abel	43962.7901
2	29,531	Cory	Booth	7330.8972
3	29,546	Christopher	Beck	98138.2131
4	29,568	Donald	Blanton	2669.3183
5	29,584	Walter	Brian	272.6468
6	29,612	Richard	Byham	608.1766
7	29,638	Rosmarie	Carroll	2361.6403
8	29,644	Brigid	Cavendish	1170.5376
9	29,653	Pei	Chow	15275.1977
10	29,660	Anthony	Chen	63686.2708

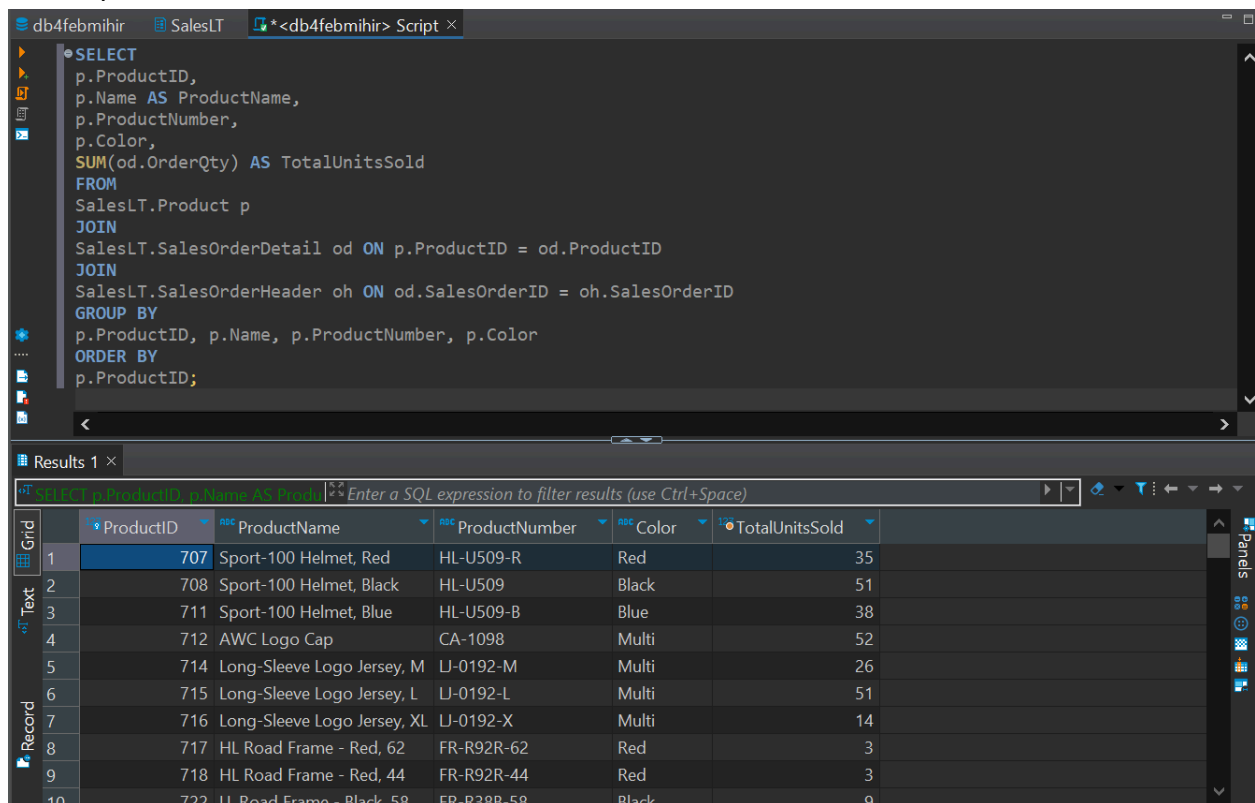
2. Display product information along with the number of units sold for each product.

```
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;

```



db4febmihir SalesLT <db4febmihir> Script x

```

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;

```

Results 1 x

Enter a SQL expression to filter results (use Ctrl+Space)

	ProductID	ProductName	ProductNumber	Color	TotalUnitsSold
1	707	Sport-100 Helmet, Red	HL-U509-R	Red	35
2	708	Sport-100 Helmet, Black	HL-U509	Black	51
3	711	Sport-100 Helmet, Blue	HL-U509-B	Blue	38
4	712	AWC Logo Cap	CA-1098	Multi	52
5	714	Long-Sleeve Logo Jersey, M	LJ-0192-M	Multi	26
6	715	Long-Sleeve Logo Jersey, L	LJ-0192-L	Multi	51
7	716	Long-Sleeve Logo Jersey, XL	LJ-0192-X	Multi	14
8	717	HL Road Frame - Red, 62	FR-R92R-62	Red	3
9	718	HL Road Frame - Red, 44	FR-R92R-44	Red	3
10	722	HL Road Frame - Black, 58	FR-R92R-58	Black	0

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

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

```

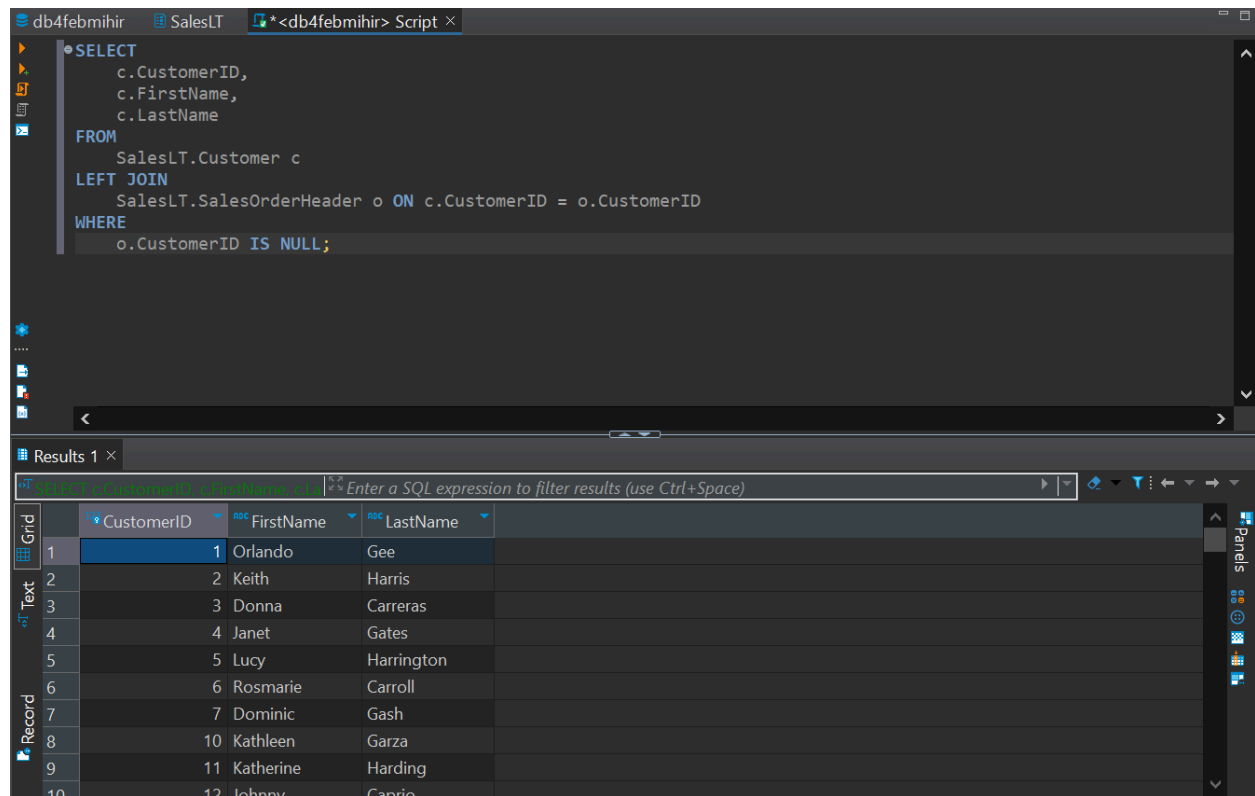
SELECT
    c.CustomerID,
    c.FirstName,

```

```

        c.LastName
FROM
    SalesLT.Customer c
LEFT JOIN
    SalesLT.SalesOrderHeader o ON c.CustomerID = o.CustomerID
WHERE
    o.CustomerID IS NULL;

```



The screenshot shows a SQL client window with a query editor and a results pane. The query editor contains the following SQL code:

```

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;

```

The results pane displays 10 rows of data in a grid format. The columns are CustomerID, FirstName, and LastName. The data is as follows:

	CustomerID	FirstName	LastName
1	1	Orlando	Gee
2	2	Keith	Harris
3	3	Donna	Carreras
4	4	Janet	Gates
5	5	Lucy	Harrington
6	6	Rosmarie	Carroll
7	7	Dominic	Gash
8	10	Kathleen	Garza
9	11	Katherine	Harding
10	12	Johnny	Carrie

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

```

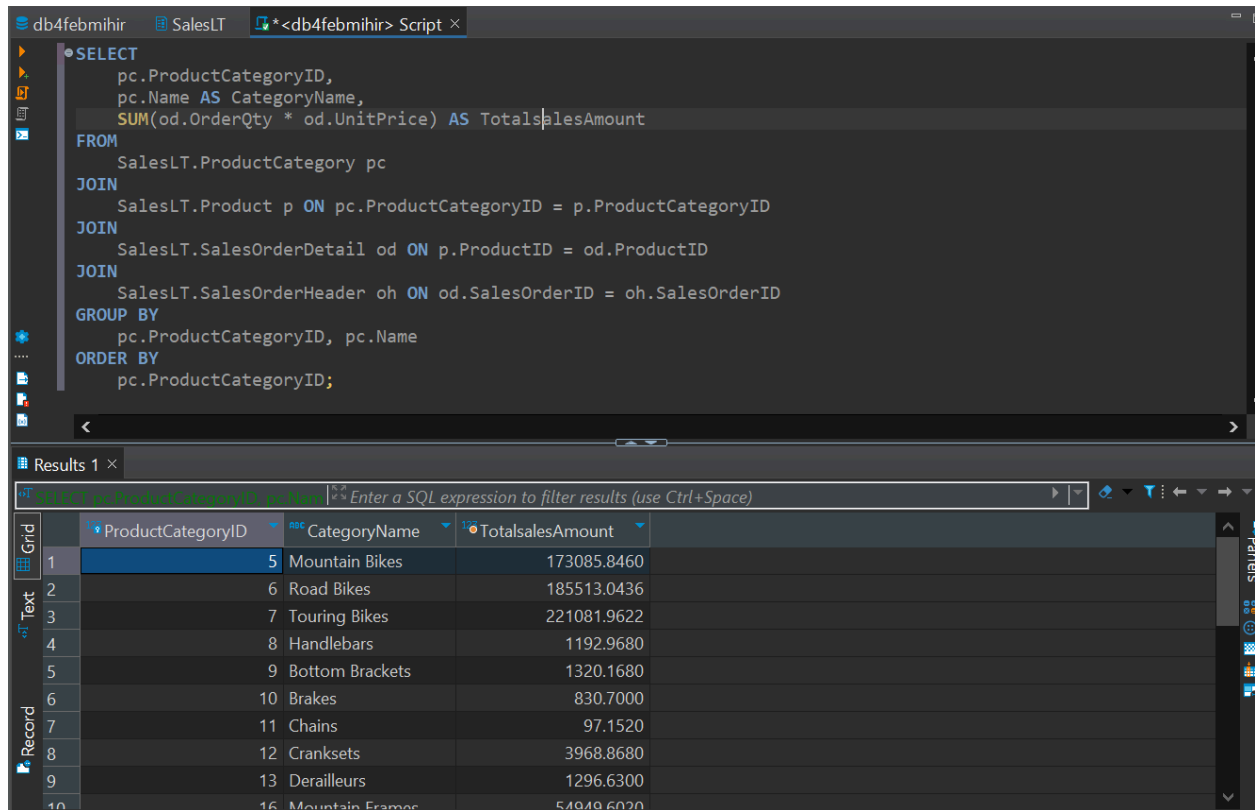
SELECT
    pc.ProductCategoryID,
    pc.Name AS CategoryName,
    SUM(od.OrderQty * od.UnitPrice) AS TotalsalesAmount
FROM
    SalesLT.ProductCategory pc
JOIN
    SalesLT.Product p ON pc.ProductCategoryID = p.ProductCategoryID
JOIN
    SalesLT.SalesOrderDetail od ON p.ProductID = od.ProductID
JOIN
    SalesLT.SalesOrderHeader oh ON od.SalesOrderID = oh.SalesOrderID
GROUP BY

```

```

        pc.ProductCategoryID, pc.Name
ORDER BY
        pc.ProductCategoryID;

```



The screenshot shows a SQL Server Enterprise Manager interface. The top pane displays a query window with the following SQL code:

```

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;

```

The bottom pane shows the results of the query in a table with 10 rows. The columns are ProductCategoryID, CategoryName, and TotalsalesAmount.

ProductCategoryID	CategoryName	TotalsalesAmount
5	Mountain Bikes	173085.8460
6	Road Bikes	185513.0436
7	Touring Bikes	221081.9622
8	Handlebars	1192.9680
9	Bottom Brackets	1320.1680
10	Brakes	830.7000
11	Chains	97.1520
12	Cranksets	3968.8680
13	Derailleurs	1296.6300
14	Mountain Frames	54040.6020

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

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

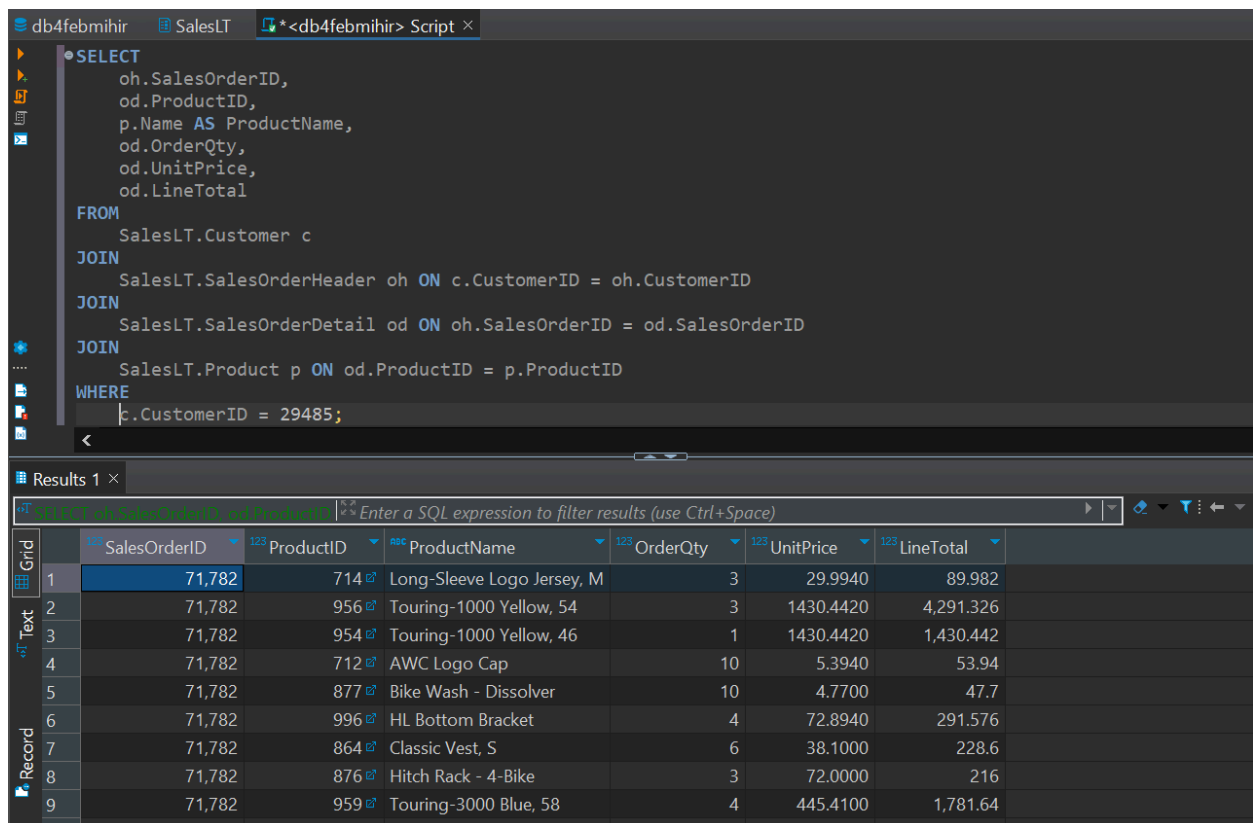
```

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;



The screenshot shows a SQL query editor with a script window and a results window. The script window contains a SELECT query that joins SalesLT.Customer, SalesLT.SalesOrderHeader, SalesLT.SalesOrderDetail, and SalesLT.Product tables, filtering for CustomerID = 29485. The results window displays a table with 9 rows of data.

```
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;
```

	122 SalesOrderID	123 ProductID	896 ProductName	123 OrderQty	123 UnitPrice	123 LineTotal	
1	71,782	714	Long-Sleeve Logo Jersey, M	3	29.9940	89.982	
2	71,782	956	Touring-1000 Yellow, 54	3	1430.4420	4,291.326	
3	71,782	954	Touring-1000 Yellow, 46	1	1430.4420	1,430.442	
4	71,782	712	AWC Logo Cap	10	5.3940	53.94	
5	71,782	877	Bike Wash - Dissolver	10	4.7700	47.7	
6	71,782	996	HL Bottom Bracket	4	72.8940	291.576	
7	71,782	864	Classic Vest, S	6	38.1000	228.6	
8	71,782	876	Hitch Rack - 4-Bike	3	72.0000	216	
9	71,782	959	Touring-3000 Blue, 58	4	445.4100	1,781.64	

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

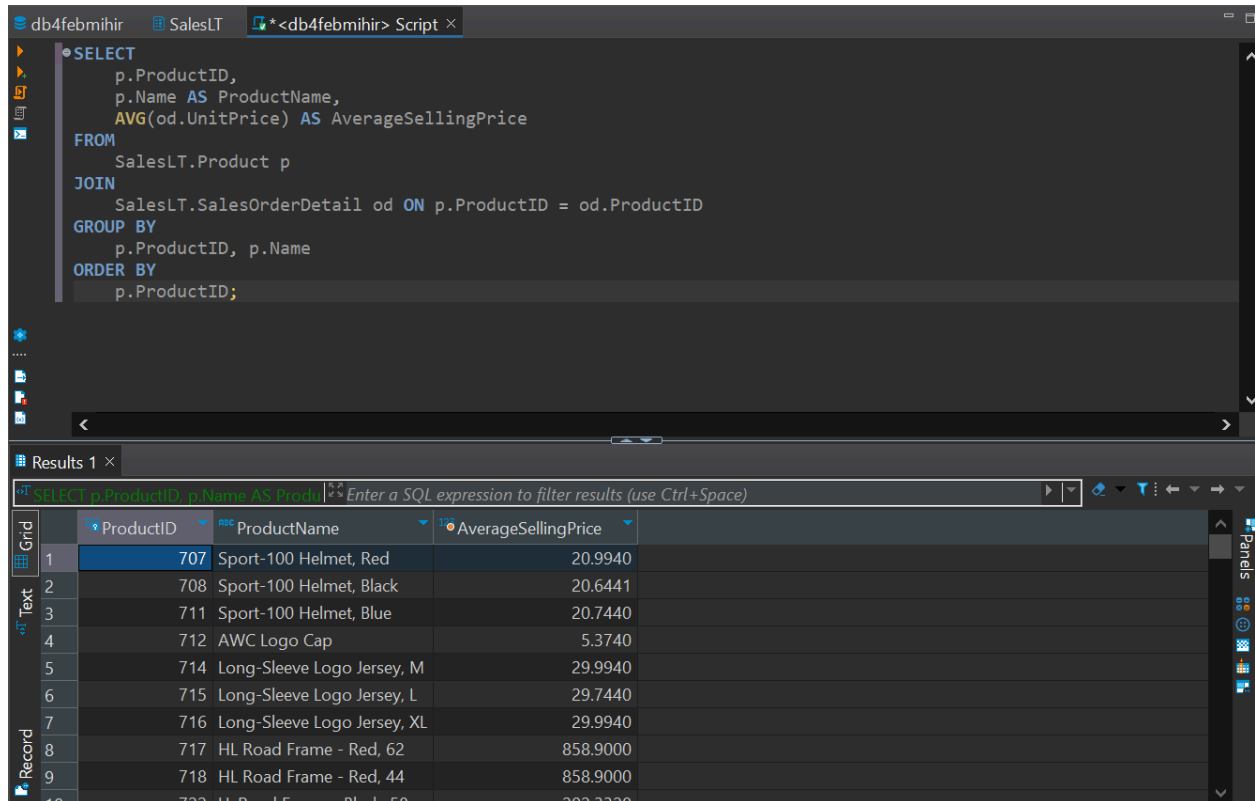
SELECT

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;

```



The screenshot shows a SQL IDE with a query window and a results window. The query window contains the following SQL code:

```

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;

```

The results window shows the following data:

ProductID	ProductName	AverageSellingPrice
707	Sport-100 Helmet, Red	20.9940
708	Sport-100 Helmet, Black	20.6441
711	Sport-100 Helmet, Blue	20.7440
712	AWC Logo Cap	5.3740
714	Long-Sleeve Logo Jersey, M	29.9940
715	Long-Sleeve Logo Jersey, L	29.7440
716	Long-Sleeve Logo Jersey, XL	29.9940
717	HL Road Frame - Red, 62	858.9000
718	HL Road Frame - Red, 44	858.9000
722	HL Road Frame - Black, 58	202.3320

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;

```


The screenshot shows a SQL Server Enterprise Manager window with a query executed in the 'Script' tab. The query is a SELECT statement that retrieves the top 1 sales order based on total amount. The results are displayed in the 'Results' tab in a grid view.

```

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;

```

	SalesOrderID	OrderDate	TotalAmount
1	71,784	2008-06-01 00:00:00.000	89,869.276314

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);

```

The screenshot shows a SQL query editor with a script window and a results grid. The script window contains a query that uses a CTE named 'CustomerOrderTotals' to calculate the total amount for each customer. The results grid displays 10 rows of data, including CustomerID, FirstName, LastName, and TotalAmount.

```

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

```

Grid	CustomerID	FirstName	LastName	TotalAmount
1	29,485	Catherine	Abel	33,319.986
2	29,546	Christopher	Beck	74,160.228
3	29,660	Anthony	Chor	47,848.026
4	29,736	Terry	Eminhizer	89,869.276314
5	29,796	Jon	Grande	65,123.463418
6	29,922	Pamala	Kotc	28,950.678108
7	29,929	Jeffrey	Kurtz	59,894.2092
8	29,932	Rebecca	Laszlo	53,248.692
9	29,938	Frank	Campbell	34,118.5356
10	29,957	Kevin	...	65,693.367096

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);

The screenshot shows a SQL query editor with the following query:

```
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);
```

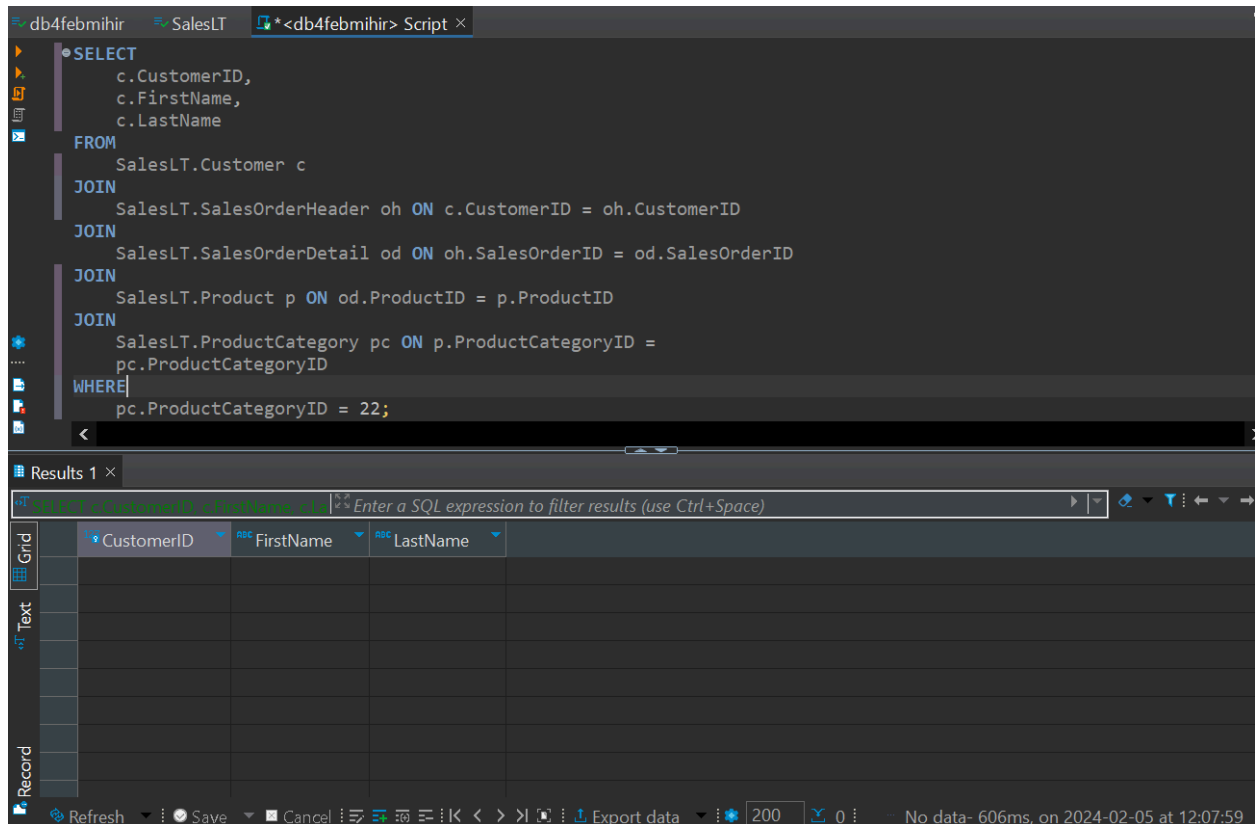
The results are displayed in a table with the following columns: ProductID, ProductName, and ListPrice.

ProductID	ProductName	ListPrice
680	HL Road Frame - Black, 58	1431.5000
706	HL Road Frame - Red, 58	1431.5000
717	HL Road Frame - Red, 62	1431.5000
718	HL Road Frame - Red, 44	1431.5000
719	HL Road Frame - Red, 48	1431.5000
720	HL Road Frame - Red, 52	1431.5000
721	HL Road Frame - Red, 56	1431.5000
739	HL Mountain Frame - Silver, 42	1364.5000
740	HL Mountain Frame - Silver, 44	1364.5000

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;

```

The screenshot displays a SQL query in the 'Script' tab of SQL Server Enterprise Manager. The query is as follows:

```

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;

```

The 'Results' tab shows the output of the query in a grid format. The grid has four columns: SalesOrderID, ProductID, ProductName, and TotalQuantity. The first row contains the following data:

Grid	SalesOrderID	ProductID	ProductName	TotalQuantity
1	123	714	Long-Sleeve Logo Jersey, M	9

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

```

```

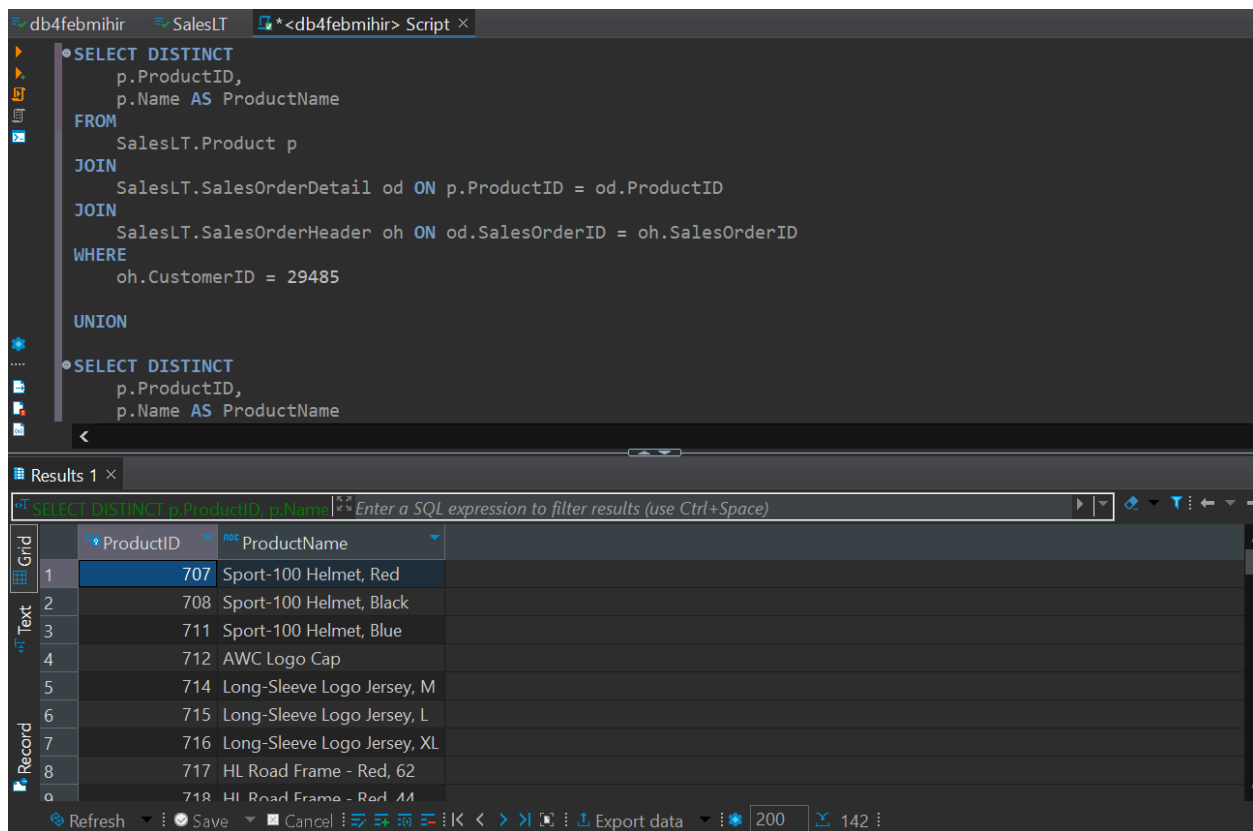
UNION

```

```

SELECT DISTINCT
    p.ProductID,
    p.Name AS ProductName
FROM
    SalesLT.Product p
JOIN
    SalesLT.SalesOrderDetail od ON p.ProductID = od.ProductID
ORDER BY
    ProductID;

```



The screenshot shows a SQL query editor with a query that uses a UNION to combine two SELECT statements. The first SELECT statement joins SalesLT.Product with SalesLT.SalesOrderDetail and SalesLT.SalesOrderHeader, filtering for CustomerID = 29485. The second SELECT statement is a simple DISTINCT query on SalesLT.Product. The results pane shows 142 rows of data, with the first 8 rows visible in a grid view.

	ProductID	ProductName
1	707	Sport-100 Helmet, Red
2	708	Sport-100 Helmet, Black
3	711	Sport-100 Helmet, Blue
4	712	AWC Logo Cap
5	714	Long-Sleeve Logo Jersey, M
6	715	Long-Sleeve Logo Jersey, L
7	716	Long-Sleeve Logo Jersey, XL
8	717	HL Road Frame - Red, 62
9	718	HL Road Frame - Red, 44

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;
```


The screenshot shows a SQL IDE with a query window and a results window. The query window contains the following SQL code:

```

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;

```

The results window shows the following data:

	CustomerID	FirstName	LastName	ProductName
1	29,485	Catherine	Abel	Sport-100 Helmet, Red
2	29,485	Catherine	Abel	Sport-100 Helmet, Black
3	29,485	Catherine	Abel	Sport-100 Helmet, Blue
4	29,485	Catherine	Abel	AWC Logo Cap
5	29,485	Catherine	Abel	Long-Sleeve Logo Jersey, M
6	29,485	Catherine	Abel	Long-Sleeve Logo Jersey, L
7	29,485	Catherine	Abel	Half-Finger Gloves, M
8	29,485	Catherine	Abel	Classic Vest, S
9	29,485	Catherine	Abel	Classic Vest, M

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;

```

The screenshot shows a SQL IDE with a query window and a results window. The query window contains the following SQL code:

```

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;

```

The results window shows the following data:

Grid	SalesOrderID	NumberOfProducts	ProductNames
1	71,774	2	ML Road Frame-W - Yellow, 48, ML Road Frame-W - Yellow, 38
2	71,780	29	LL Mountain Frame - Silver, 44, LL Mountain Frame - Black, 44, LL Mountain Frame - Black, 48, ML M
3	71,782	43	HL Bottom Bracket, LL Bottom Bracket, Touring-3000 Blue, 44, Front Derailleur, Front Brakes, HL Crar
4	71,783	43	Short-Sleeve Classic Jersey, XL, Classic Vest, S, Classic Vest, M, Water Bottle - 30 oz., Patch Kit/8 Patc
5	71,784	43	Touring-1000 Yellow, 60, Touring-1000 Yellow, 50, Touring-1000 Yellow, 46, Touring-2000 Blue, 60,
6	71,796	20	HL Touring Seat/Saddle, HL Touring Frame - Blue, 50, LL Touring Frame - Yellow, 50, LL Touring Fran
7	71,797	46	Road-750 Black, 52, Road-750 Black, 48, LL Road Pedal, HL Road Pedal, ML Road Pedal, Road-350-1
8	71,815	3	LL Road Frame - Black, 52, ML Road Frame-W - Yellow, 44, Racing Socks, M
9	71,816	7	Short-Sleeve Classic Jersey, L, AWC Logo Cap, Long-Sleeve Logo Jersey, L, Touring-3000 Yellow, 44,

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;

```

The screenshot shows a SQL query in the 'Script' tab of SQL Server Enterprise Manager. The query is a SELECT statement that joins the SalesLT.Customer, SalesLT.SalesOrderHeader, SalesLT.SalesOrderDetail, and SalesLT.Product tables. The results are displayed in the 'Results' tab, showing a grid of data with columns: CustomerID, FirstName, LastName, SalesOrderID, ProductID, ProductName, OrderQty, and UnitPrice. The data shows multiple orders for the same customer (Catherine Abel) and various products.

	CustomerID	FirstName	LastName	SalesOrderID	ProductID	ProductName	OrderQty	UnitPrice
1	29,485	Catherine	Abel	71,782	707	Sport-100 Helmet, Red	3	20
2	29,485	Catherine	Abel	71,782	708	Sport-100 Helmet, Black	7	20
3	29,485	Catherine	Abel	71,782	711	Sport-100 Helmet, Blue	6	20
4	29,485	Catherine	Abel	71,782	712	AWC Logo Cap	10	5
5	29,485	Catherine	Abel	71,782	714	Long-Sleeve Logo Jersey, M	3	29
6	29,485	Catherine	Abel	71,782	715	Long-Sleeve Logo Jersey, L	8	29
7	29,485	Catherine	Abel	71,782	859	Half-Finger Gloves, M	1	14
8	29,485	Catherine	Abel	71,782	864	Classic Vest, S	6	38

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.

```

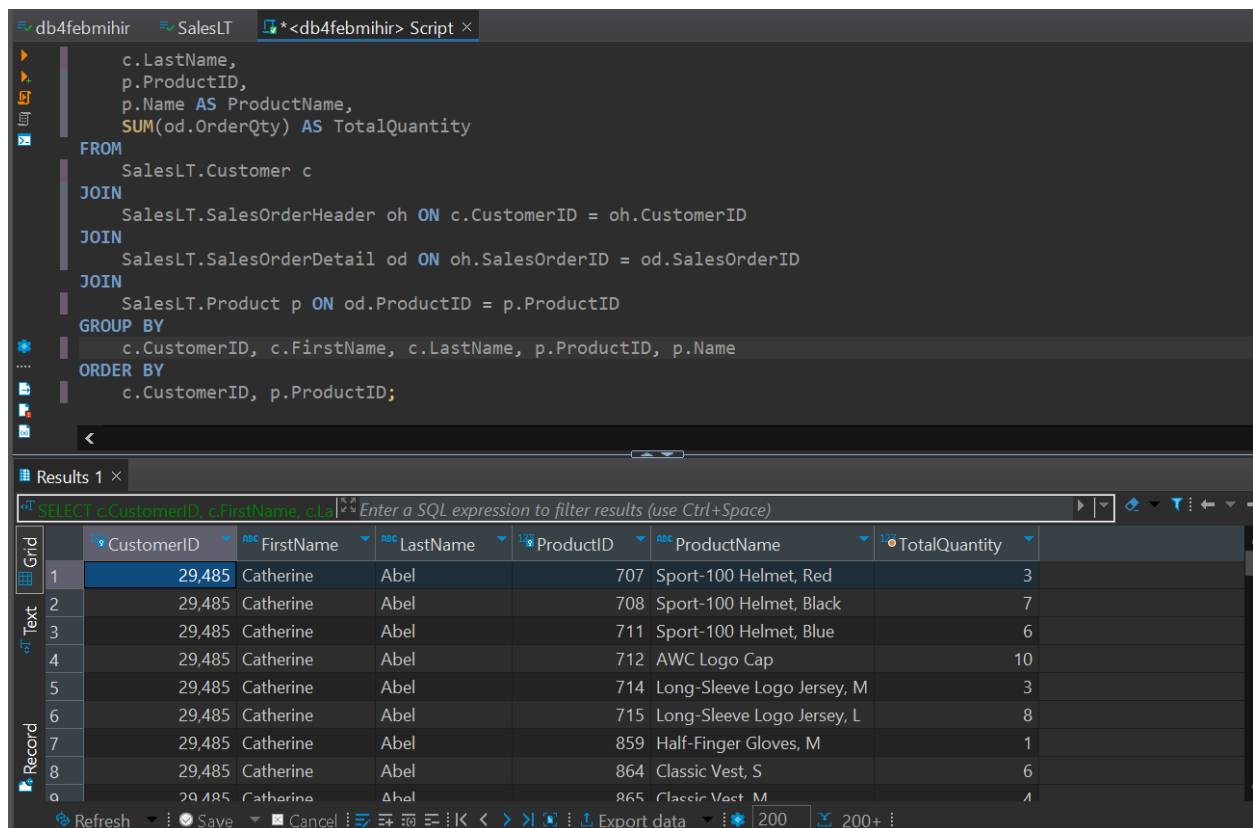
SELECT
c.CustomerID,
c.FirstName,
c.LastName,
p.ProductID,

```

```

        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;

```



The screenshot shows a SQL query editor with a query window and a results window. The query window contains the same SQL query as above. The results window shows a table with 8 rows of data.

	CustomerID	FirstName	LastName	ProductID	ProductName	TotalQuantity
1	29,485	Catherine	Abel	707	Sport-100 Helmet, Red	3
2	29,485	Catherine	Abel	708	Sport-100 Helmet, Black	7
3	29,485	Catherine	Abel	711	Sport-100 Helmet, Blue	6
4	29,485	Catherine	Abel	712	AWC Logo Cap	10
5	29,485	Catherine	Abel	714	Long-Sleeve Logo Jersey, M	3
6	29,485	Catherine	Abel	715	Long-Sleeve Logo Jersey, L	8
7	29,485	Catherine	Abel	859	Half-Finger Gloves, M	1
8	29,485	Catherine	Abel	864	Classic Vest, S	6

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;

```

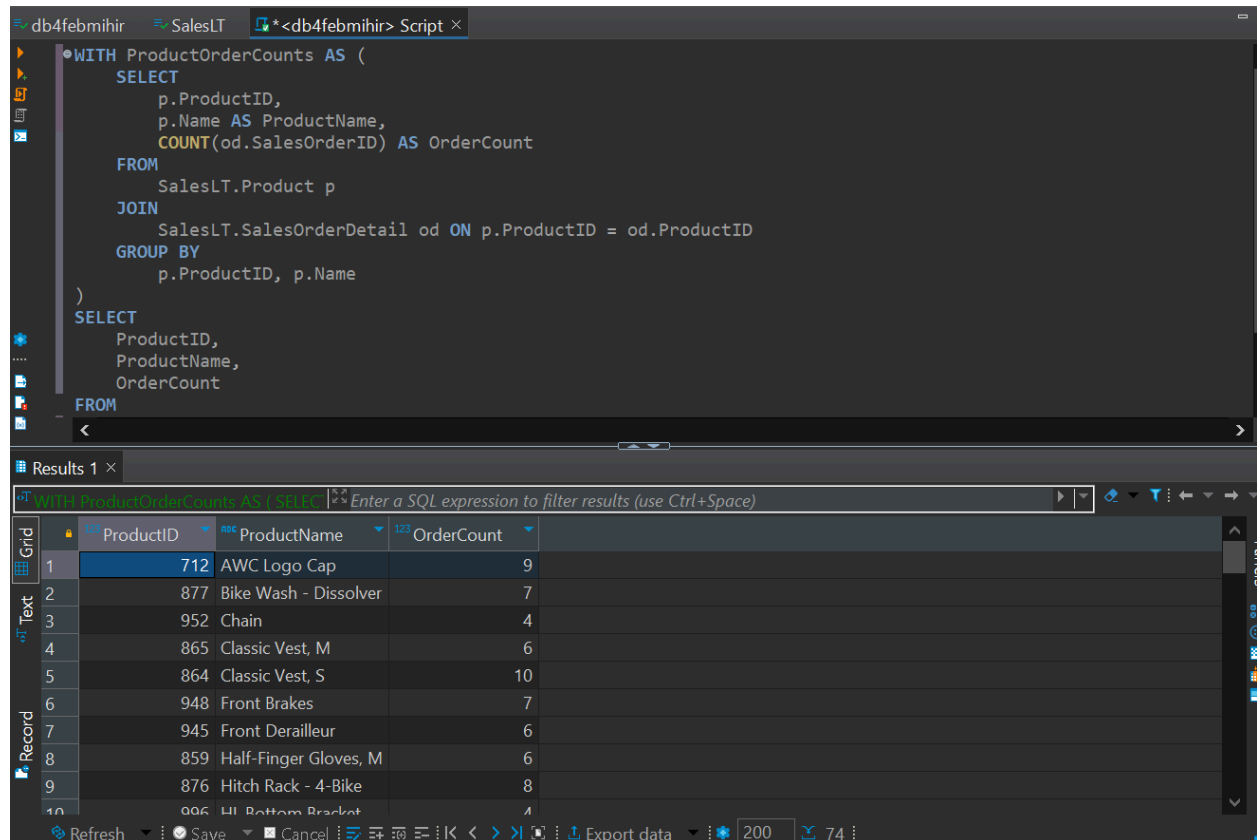
42. Display products that have been ordered more than the average number of times.

```

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);



The screenshot shows a SQL IDE with a query editor and a results pane. The query in the editor is as follows:

```
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
```

The results pane displays the following data:

ProductID	ProductName	OrderCount
712	AWC Logo Cap	9
877	Bike Wash - Dissolver	7
952	Chain	4
865	Classic Vest, M	6
864	Classic Vest, S	10
948	Front Brakes	7
945	Front Derailleur	6
859	Half-Finger Gloves, M	6
876	Hitch Rack - 4-Bike	8
996	Hi Bottom Bracket	4

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;
```


Home > db4febmihir (dbs4febmihir/db4febmihir)

db4febmihir (dbs4febmihir/db4febmihir) | Query editor (preview) ☆ ...

SQL database

» Login + New Query ↑ Open query Feedback Getting started

Query 1 ✕

Run ☐ Cancel query Save query Export data as Show only Editor

```

20 JOIN
21 (SELECT ProductID, AvgQuantity FROM AvgQuantityPerProduct) avgQ ON od.ProductID = avgQ.ProductID
22 WHERE
23 od.OrderQty > avgQ.AvgQuantity
24 ORDER BY
25 od.SalesOrderID, od.ProductID;
26

```

Results Messages

Search to filter items...

SalesOrderID	ProductID	ProductName	OrderQty
71780	780	Mountain-200 Silver, 42	4
71780	809	ML Mountain Handlebars	3

Query succeeded | 3s

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;

Query 1 ✕

Run ☐ Cancel query Save query Export data as Show only Editor

```
1 WITH ProductsOrderedByCustomers AS (  
2   SELECT DISTINCT  
3     od.ProductID  
4   FROM  
5     SalesLT.SalesOrderDetail od  
6 )
```

Results Messages

Query succeeded: Affected rows: 0

Query succeeded | 3s