

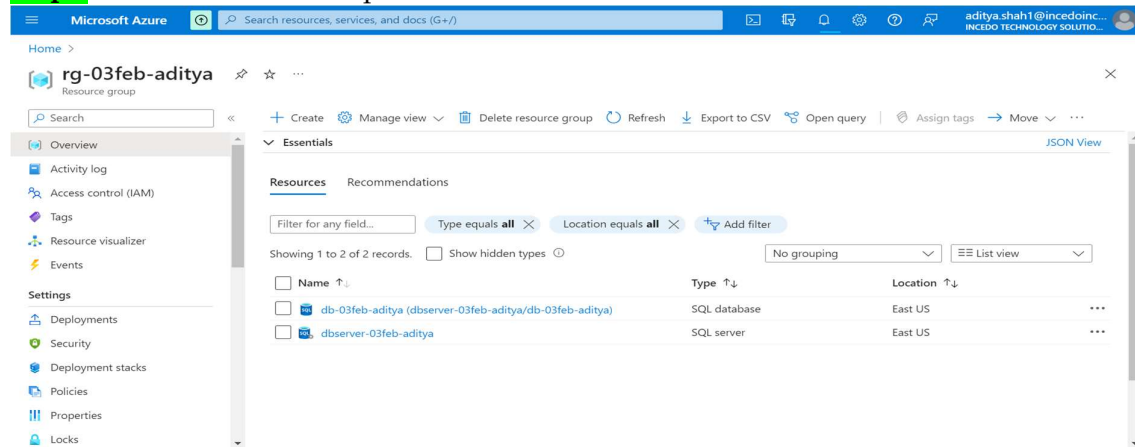
MODULE 1:

AdventureWorks sample database in Azure SQL Database

Process Flow –

Step 1 Login to Azure Portal

Step 2 Create Resource Group –



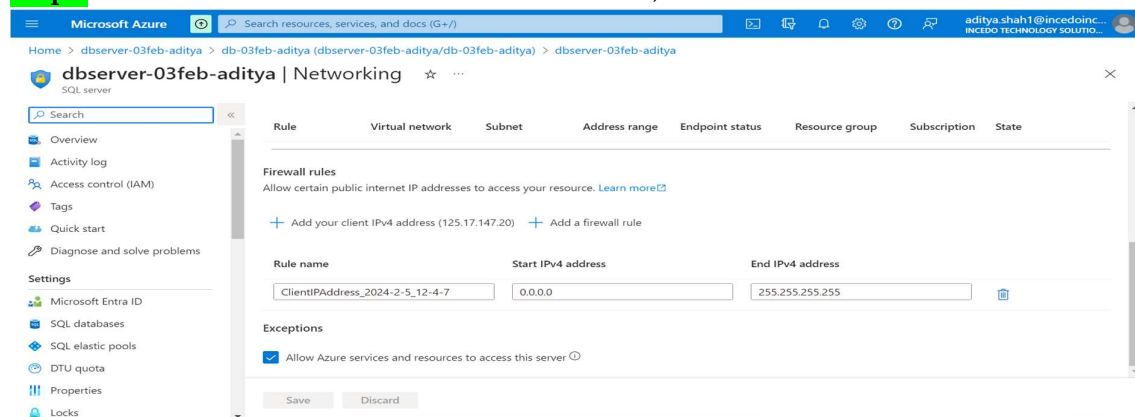
Step 3 Create Azure SQL Database

Step 4 In the "Restore" Option, configure Restore Options as "Sample -> AdventureWorks"

Step 5 Review and Create the Database

Step 6 Validation and Restore

Step 7 Set Firewall to connect to all the Networks i.e., 0.0.0.0 to 255.255.255.255 –



Step 8 Monitor the Restore Progress

Step 9 Verify the Restored Database

Step 10 Sample Dataset Loaded on SQL Database -

Microsoft Azure | Upgrade | Search resources, services, and docs (G+)

Home > db-03feb-aditya (dbserver-03feb-aditya/db-03feb-aditya) | Query editor (preview) ☆ ...

SQL database

» Login + New Query Open query Feedback Getting started

db-03feb-aditya (dbadmin)

Showing limited object explorer here. For full capability please click here to open Azure Data Studio.

Tables

- > dbo.BuildVersion ...
- > dbo.ErrorLog ...
- > SalesLT.Address ...
- > SalesLT.Customer ...
- > SalesLT.CustomerAddress ...
- > SalesLT.Product ...
- > SalesLT.ProductCategory ...
- > SalesLT.ProductDescription ...
- > SalesLT.ProductModel ...
- > SalesLT.ProductModelProductDesc ...
- > SalesLT.SalesOrderDetail ...
- > SalesLT.SalesOrderHeader ...

Views

- > Stored Procedures

Query 1

Run Cancel query Save query Export data as Show only Editor

1

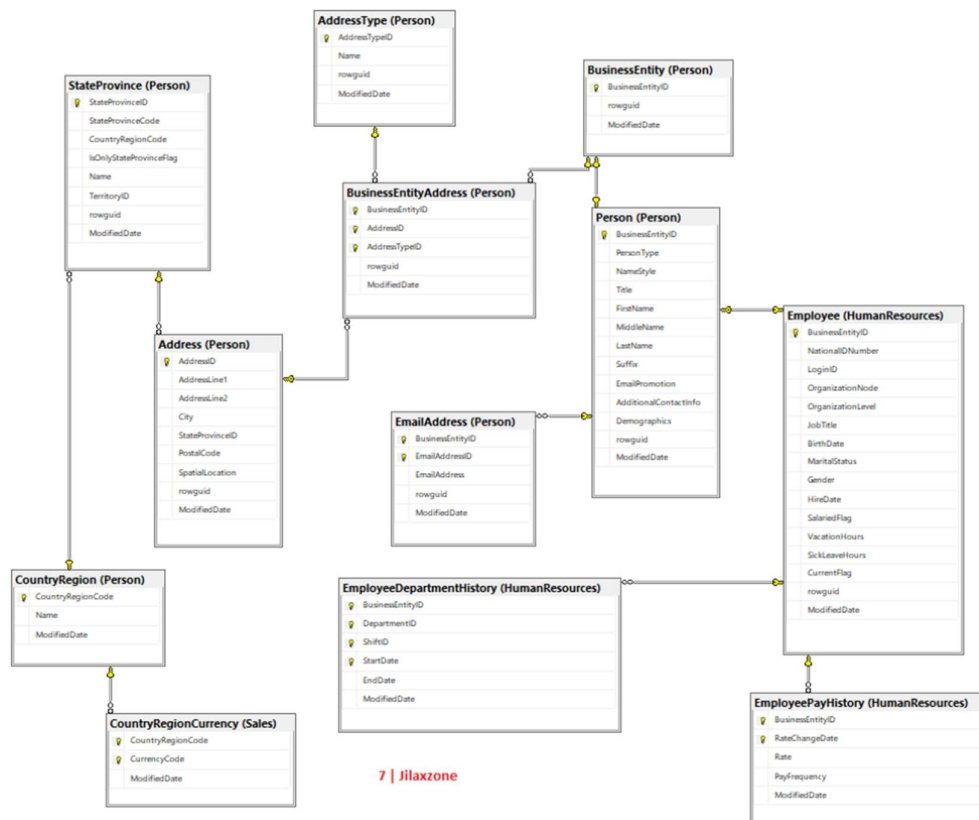
Results Messages

Search to filter items...

Ready

Step 11 Connect the Restored Database to Dbeaver and Execute the Queries.

MODULE 2: T-SQL Operations/Queries

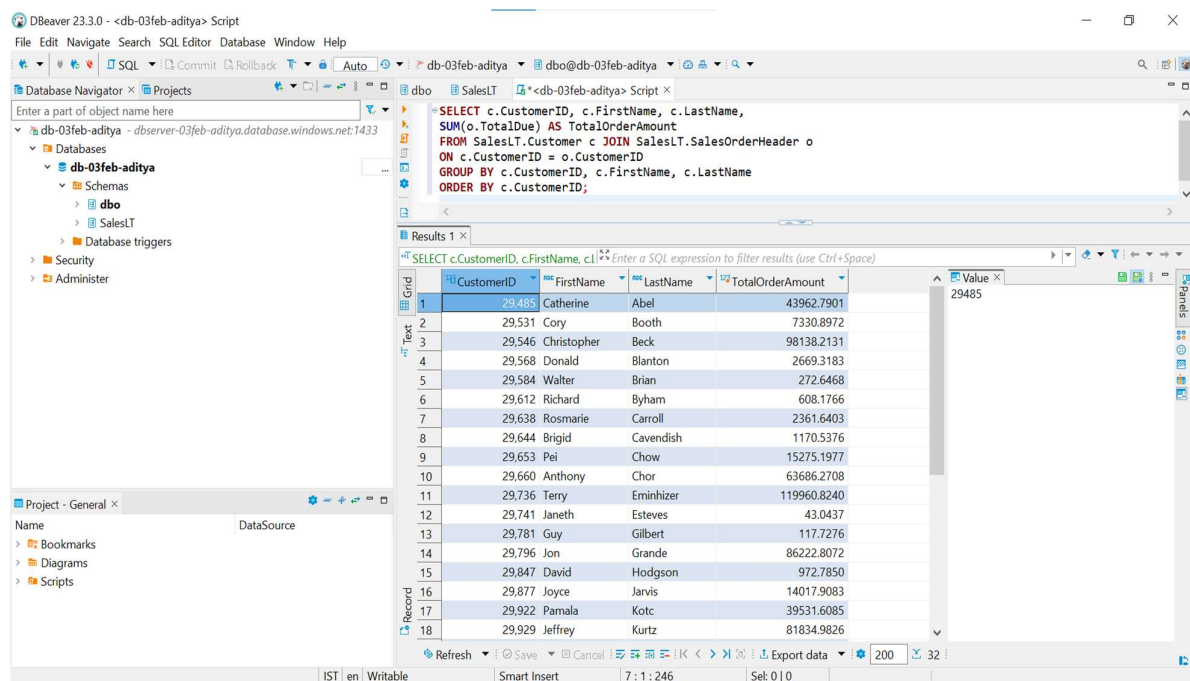


BASIC

Joins:

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

```
SELECT c.CustomerID, c.FirstName, c.LastName,  
SUM(o.TotalDue) AS TotalOrderAmount  
FROM SalesLT.Customer c JOIN SalesLT.SalesOrderHeader o  
ON c.CustomerID = o.CustomerID  
GROUP BY c.CustomerID, c.FirstName, c.LastName  
ORDER BY c.CustomerID;
```



The screenshot shows the DBeaver 23.3.0 interface. The SQL Editor displays the following query:

```
SELECT c.CustomerID, c.FirstName, c.LastName,  
SUM(o.TotalDue) AS TotalOrderAmount  
FROM SalesLT.Customer c JOIN SalesLT.SalesOrderHeader o  
ON c.CustomerID = o.CustomerID  
GROUP BY c.CustomerID, c.FirstName, c.LastName  
ORDER BY c.CustomerID;
```

The Results tab shows the following data:

CustomerID	FirstName	LastName	TotalOrderAmount
29.485	Catherine	Abel	43962.7901
29.531	Cory	Booth	7330.8972
29.546	Christopher	Beck	98138.2131
29.568	Donald	Blanton	2669.3183
29.584	Walter	Brian	272.6468
29.612	Richard	Byham	608.1766
29.638	Rosmarie	Carroll	2361.6403
29.644	Brigid	Cavendish	1170.5376
29.653	Pei	Chow	15275.1977
29.660	Anthony	Chor	63686.2708
29.736	Terry	Eminhizer	119960.8240
29.741	Janeth	Esteves	43.0437
29.781	Guy	Gilbert	117.7276
29.796	Jon	Grande	86222.8072
29.847	David	Hodgson	972.7850
29.877	Joyce	Jarvis	14017.9083
29.922	Pamala	Kotc	39531.6085
29.929	Jeffrey	Kurtz	81834.9826

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;
```

dbo SalesLT *<db-03feb-aditya> 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

SELECT p.ProductID, p.Name AS ProductName, p.ProductNumber, p.Color, TotalUnitsSold

Grid	ProductID	ProductName	ProductNumber	Color	TotalUnitsSold
1	707	Sport-100 Helmet, Red	HL-U509-R	Red	707
2	708	Sport-100 Helmet, Black	HL-U509	Black	
3	711	Sport-100 Helmet, Blue	HL-U509-B	Blue	
4	712	AWC Logo Cap	CA-1098	Multi	
5	714	Long-Sleeve Logo Jersey, M	LJ-0192-M	Multi	
6	715	Long-Sleeve Logo Jersey, L	LJ-0192-L	Multi	
7	716	Long-Sleeve Logo Jersey, XL	LJ-0192-X	Multi	
8	717	HL Road Frame - Red, 62	FR-R92R-62	Red	
9	718	HL Road Frame - Red, 44	FR-R92R-44	Red	
10	722	LL Road Frame - Black, 58	FR-R38B-58	Black	
11	738	LL Road Frame - Black, 52	FR-R38B-52	Black	
12	739	HL Mountain Frame - Silver, 42	FR-M94S-42	Silver	
13	742	HL Mountain Frame - Silver, 46	FR-M94S-46	Silver	
14	743	HL Mountain Frame - Black, 42	FR-M94B-42	Black	
15	747	HL Mountain Frame - Black, 38	FR-M94B-38	Black	

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;

```

dbo SalesLT *<db-03feb-aditya> Script x

```

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;

```

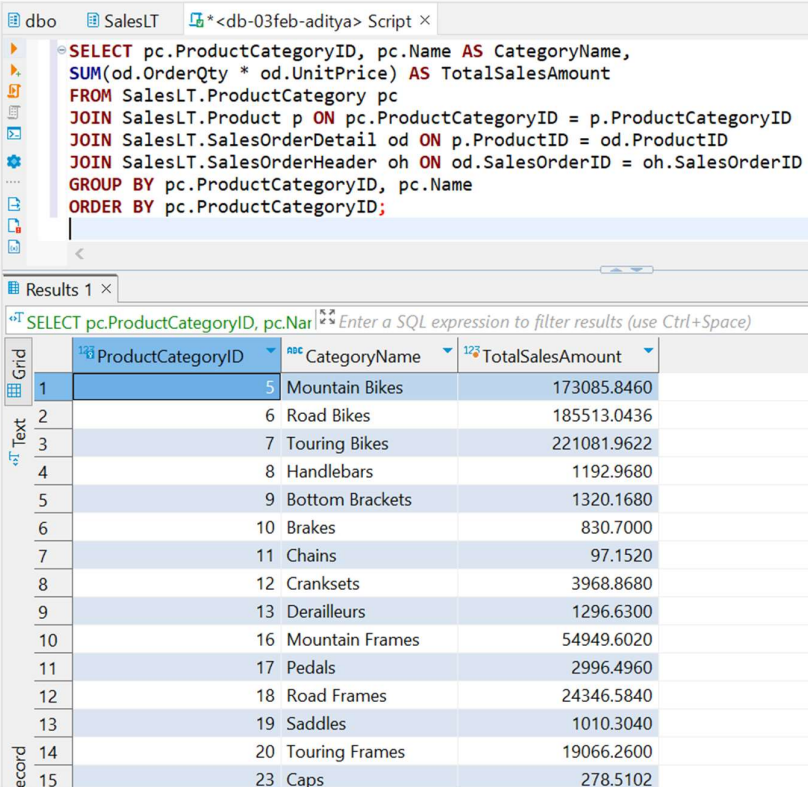
Results 1 x

SELECT c.CustomerID, c.FirstName, c.LastName

Grid	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	Caprio
11	16	Christopher	Beck
12	18	David	Liu
13	19	John	Beaver
14	20	Jean	Handley
15	21	Jinghao	Liu
16	22	Linda	Burnett
17	23	Kerim	Hanif
18	24	Kevin	Liu

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 T-SQL query that calculates the total sales amount for each product category. The bottom pane shows the results of the query in a grid format. The results table has three columns: ProductCategoryID, CategoryName, and TotalSalesAmount. The data is sorted by ProductCategoryID in ascending order.

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
16	Mountain Frames	54949.6020
17	Pedals	2996.4960
18	Road Frames	24346.5840
19	Saddles	1010.3040
20	Touring Frames	19066.2600
23	Caps	278.5102

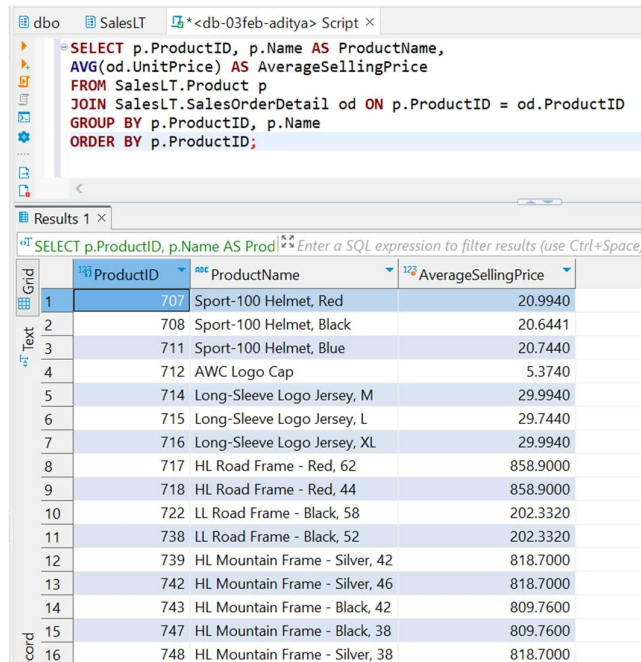
6. Display the names of employees and their direct managers.

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;
```


GROUP BY p.ProductID, p.Name
ORDER BY p.ProductID;



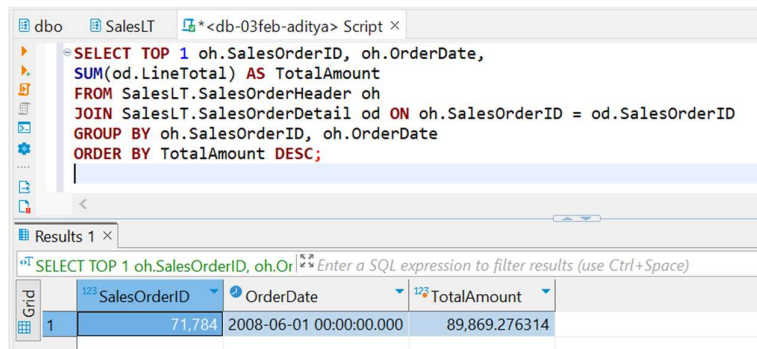
The screenshot shows a SQL query in the 'Script' tab of SQL Server Enterprise Manager. The query is a SELECT statement that joins SalesLT.Product (p) and SalesLT.SalesOrderDetail (od) on p.ProductID = od.ProductID. It calculates the average unit price for each product. The results are shown in the 'Results' tab as a grid with 16 rows and 4 columns: ProductID, ProductName, AverageSellingPrice, and an empty column.

	ProductID	ProductName	AverageSellingPrice
1	707	Sport-100 Helmet, Red	20.9940
2	708	Sport-100 Helmet, Black	20.6441
3	711	Sport-100 Helmet, Blue	20.7440
4	712	AWC Logo Cap	5.3740
5	714	Long-Sleeve Logo Jersey, M	29.9940
6	715	Long-Sleeve Logo Jersey, L	29.7440
7	716	Long-Sleeve Logo Jersey, XL	29.9940
8	717	HL Road Frame - Red, 62	858.9000
9	718	HL Road Frame - Red, 44	858.9000
10	722	LL Road Frame - Black, 58	202.3320
11	738	LL Road Frame - Black, 52	202.3320
12	739	HL Mountain Frame - Silver, 42	818.7000
13	742	HL Mountain Frame - Silver, 46	818.7000
14	743	HL Mountain Frame - Black, 42	809.7600
15	747	HL Mountain Frame - Black, 38	809.7600
16	748	HL Mountain Frame - Silver, 38	818.7000

Subquery –

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 query in the 'Script' tab of SQL Server Enterprise Manager. The query is a SELECT statement that joins SalesLT.SalesOrderHeader (oh) and SalesLT.SalesOrderDetail (od) on oh.SalesOrderID = od.SalesOrderID. It calculates the total amount for each order and returns the top 1 order with the highest total amount. The results are shown in the 'Results' tab as a grid with 1 row and 4 columns: SalesOrderID, OrderDate, TotalAmount, and an empty column.

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

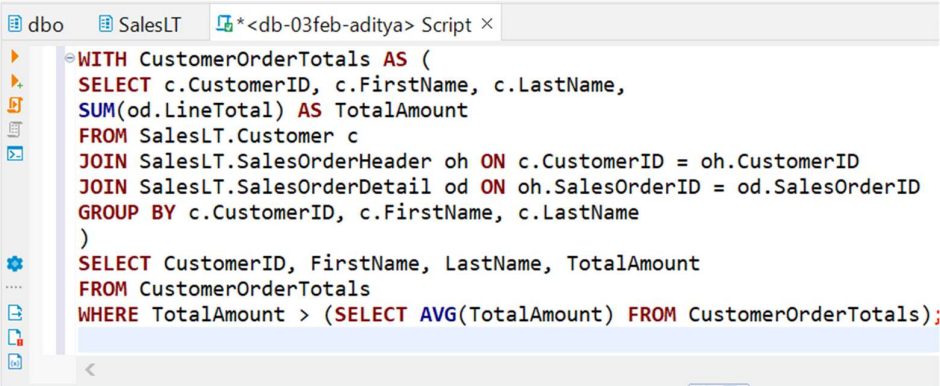
12. Display customers who have placed orders with a total amount greater than the average.

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 script in the 'Script' tab of SQL Server Enterprise Manager. The query defines a Common Table Expression (CTE) named 'CustomerOrderTotals' which calculates the total amount for each customer by summing the line totals from the SalesOrderDetail table, joined with the Customer and SalesOrderHeader tables. The main query then selects from this CTE, filtering for customers whose total order amount is greater than the average total order amount across all customers.

The 'Results' tab shows the output of the query as a table with 12 rows. The columns are CustomerID, FirstName, LastName, and TotalAmount. The data is as follows:

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	Liu	65,683.367986
11	30,050	Krishna	Sunkammurali	79,589.616024
12	30,113	Raja	Venugopal	29,923.008

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);

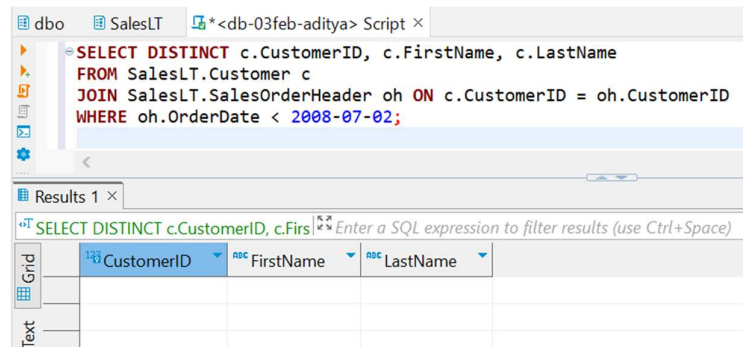
```


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;
```



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results grid. The query is:

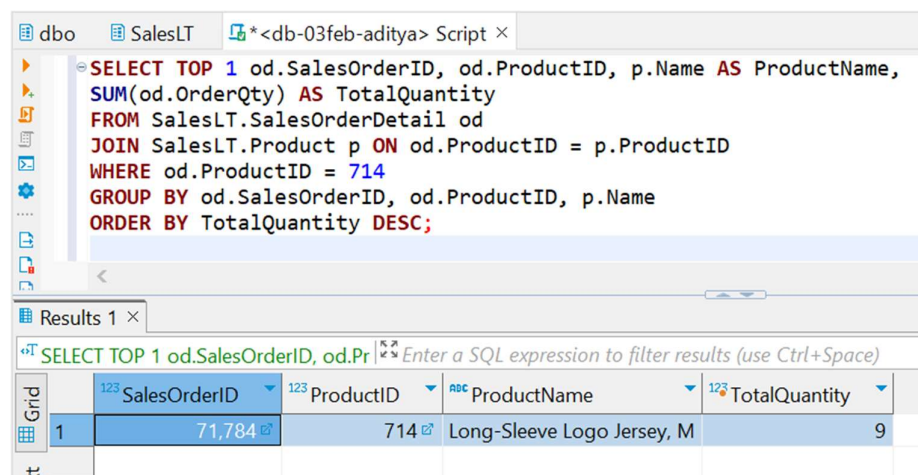
```
SELECT DISTINCT c.CustomerID, c.FirstName, c.LastName
FROM SalesLT.Customer c
JOIN SalesLT.SalesOrderHeader oh ON c.CustomerID = oh.CustomerID
WHERE oh.OrderDate < 2008-07-02;
```

The results grid shows the following data:

CustomerID	FirstName	LastName
123	CustomerID	FirstName

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 shows a SQL Server Enterprise Manager window with a query editor and a results grid. The query is:

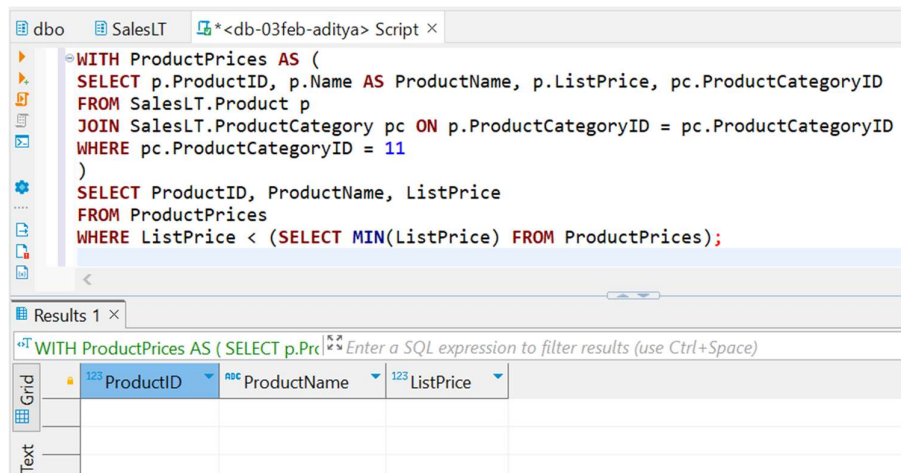
```
SELECT TOP 1 od.SalesOrderID, od.ProductID, p.Name AS ProductName,
SUM(od.OrderQty) AS TotalQuantity
FROM SalesLT.SalesOrderDetail od
JOIN SalesLT.Product p ON od.ProductID = p.ProductID
WHERE od.ProductID = 714
GROUP BY od.SalesOrderID, od.ProductID, p.Name
ORDER BY TotalQuantity DESC;
```

The results grid shows the following data:

SalesOrderID	ProductID	ProductName	TotalQuantity
1	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.

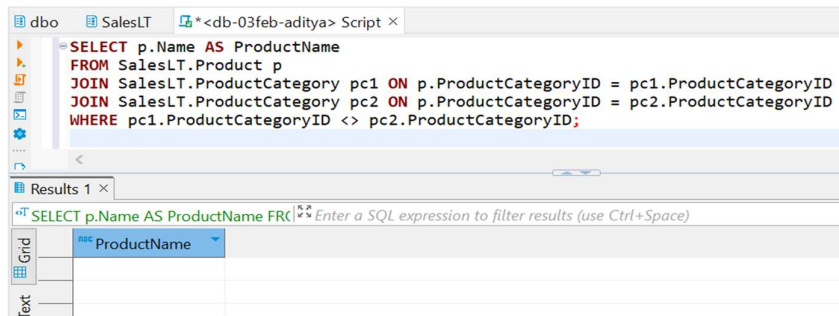
Data Insufficient

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.

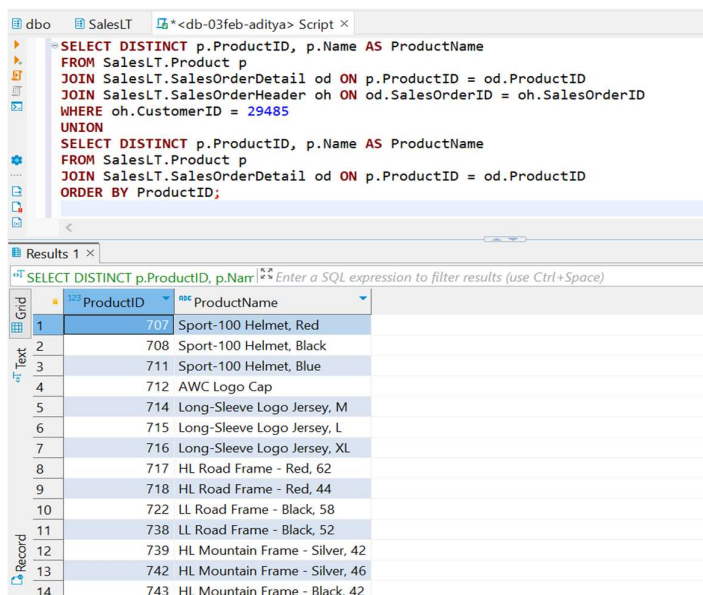
Data Insufficient

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;
```



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

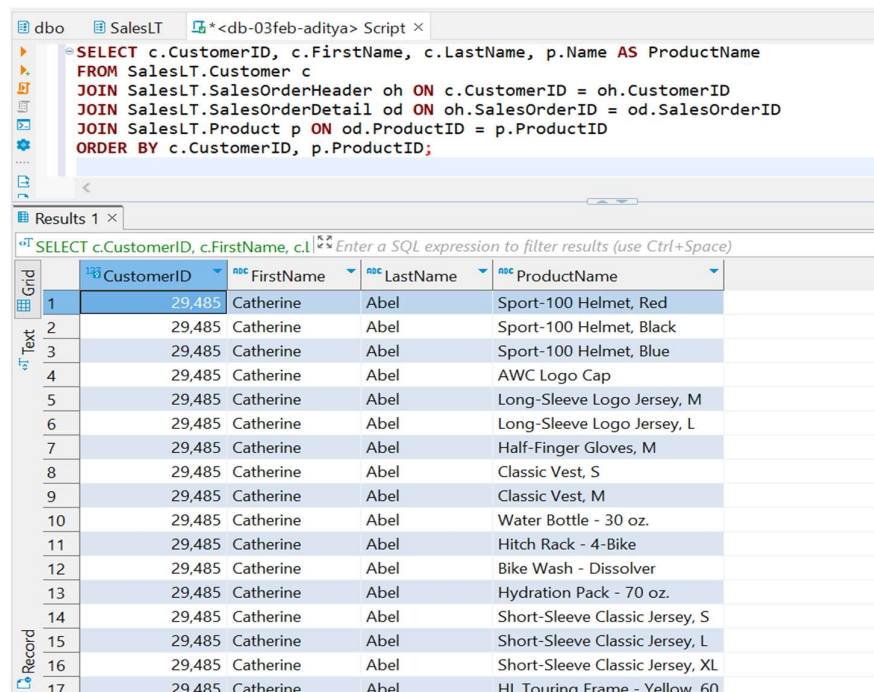
30. Combine information about employees who are managers and employees who have managers

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 Server Enterprise Manager interface. At the top, a script window displays the following SQL query:

```
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;
```

Below the script window, the 'Results' tab shows the output of the query. The results are displayed in a grid with the following columns: CustomerID, FirstName, LastName, and ProductName. The data is sorted by CustomerID and then by ProductID.

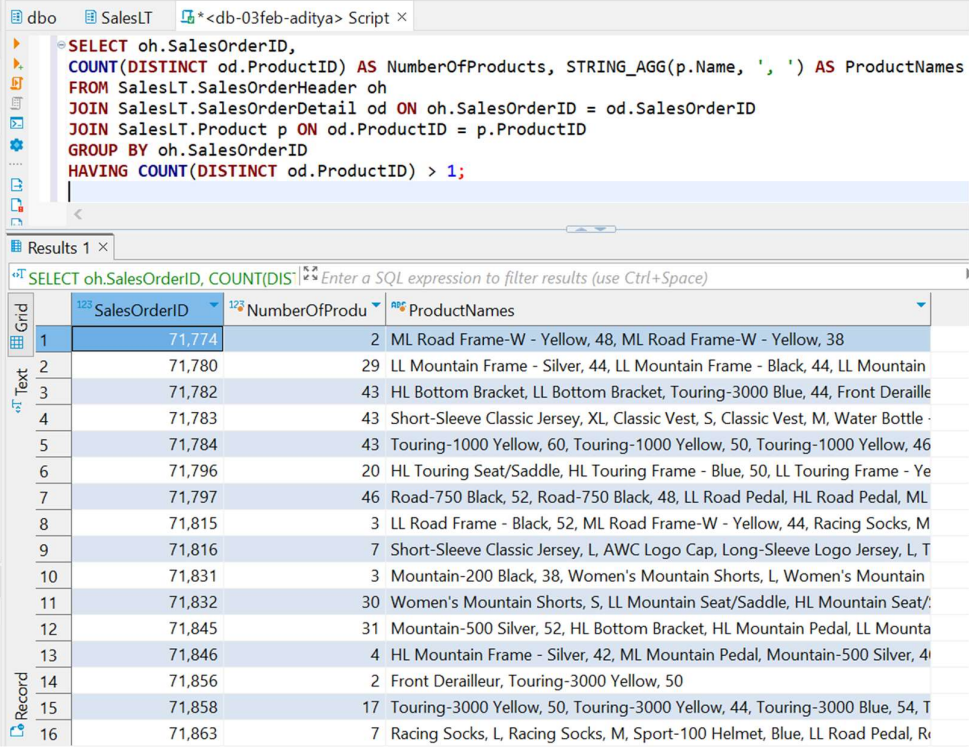
	CustomerID	FirstName	LastName	ProductName
1	29,485	Catherine	Abel	Sport-100 Helmet, Red
2	29,485	Catherine	Abel	Sport-100 Helmet, Black
3	29,485	Catherine	Abel	Sport-100 Helmet, Blue
4	29,485	Catherine	Abel	AWC Logo Cap
5	29,485	Catherine	Abel	Long-Sleeve Logo Jersey, M
6	29,485	Catherine	Abel	Long-Sleeve 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
10	29,485	Catherine	Abel	Water Bottle - 30 oz.
11	29,485	Catherine	Abel	Hitch Rack - 4-Bike
12	29,485	Catherine	Abel	Bike Wash - Dissolver
13	29,485	Catherine	Abel	Hydration Pack - 70 oz.
14	29,485	Catherine	Abel	Short-Sleeve Classic Jersey, S
15	29,485	Catherine	Abel	Short-Sleeve Classic Jersey, L
16	29,485	Catherine	Abel	Short-Sleeve Classic Jersey, XL
17	29,485	Catherine	Abel	HL Touring Frame - Yellow, 60

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 Server Enterprise Manager window with a query executed against the 'SalesLT' database. The query is displayed in the 'Script' tab, and the results are shown in the 'Results' tab. The results are presented in a grid with columns: SalesOrderID, NumberOfProducts, and ProductNames. The data shows 16 rows of orders, each with a unique SalesOrderID and a list of product names.

SalesOrderID	NumberOfProducts	ProductNames
71,774	2	ML Road Frame-W - Yellow, 48, ML Road Frame-W - Yellow, 38
71,780	29	LL Mountain Frame - Silver, 44, LL Mountain Frame - Black, 44, LL Mountain
71,782	43	HL Bottom Bracket, LL Bottom Bracket, Touring-3000 Blue, 44, Front Deraille
71,783	43	Short-Sleeve Classic Jersey, XL, Classic Vest, S, Classic Vest, M, Water Bottle
71,784	43	Touring-1000 Yellow, 60, Touring-1000 Yellow, 50, Touring-1000 Yellow, 46
71,796	20	HL Touring Seat/Saddle, HL Touring Frame - Blue, 50, LL Touring Frame - Ye
71,797	46	Road-750 Black, 52, Road-750 Black, 48, LL Road Pedal, HL Road Pedal, ML
71,815	3	LL Road Frame - Black, 52, ML Road Frame-W - Yellow, 44, Racing Socks, M
71,816	7	Short-Sleeve Classic Jersey, L, AWC Logo Cap, Long-Sleeve Logo Jersey, L, T
71,831	3	Mountain-200 Black, 38, Women's Mountain Shorts, L, Women's Mountain
71,832	30	Women's Mountain Shorts, S, LL Mountain Seat/Saddle, HL Mountain Seat/
71,845	31	Mountain-500 Silver, 52, HL Bottom Bracket, HL Mountain Pedal, LL Mouna
71,846	4	HL Mountain Frame - Silver, 42, ML Mountain Pedal, Mountain-500 Silver, 4
71,856	2	Front Derailleur, Touring-3000 Yellow, 50
71,858	17	Touring-3000 Yellow, 50, Touring-3000 Yellow, 44, Touring-3000 Blue, 54, T
71,863	7	Racing Socks, L, Racing Socks, M, Sport-100 Helmet, Blue, LL Road Pedal, R

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.

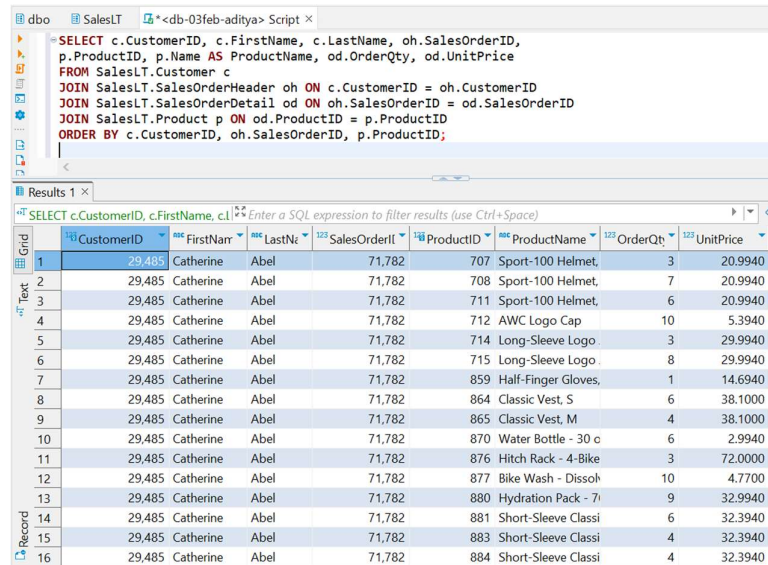
Data Insufficient

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;

```



Results 1 x

SELECT c.CustomerID, c.FirstName, c.LastName, oh.SalesOrderID, p.ProductID, p.Name AS ProductName, od.OrderQty, od.UnitPrice

	CustomerID	FirstName	LastName	SalesOrderID	ProductID	ProductName	OrderQty	UnitPrice
1	29,485	Catherine	Abel	71,782	707	Sport-100 Helmet,	3	20.9940
2	29,485	Catherine	Abel	71,782	708	Sport-100 Helmet,	7	20.9940
3	29,485	Catherine	Abel	71,782	711	Sport-100 Helmet,	6	20.9940
4	29,485	Catherine	Abel	71,782	712	AWC Logo Cap	10	5.3940
5	29,485	Catherine	Abel	71,782	714	Long-Sleeve Logo	3	29.9940
6	29,485	Catherine	Abel	71,782	715	Long-Sleeve Logo	8	29.9940
7	29,485	Catherine	Abel	71,782	859	Half-Finger Gloves,	1	14.6940
8	29,485	Catherine	Abel	71,782	864	Classic Vest, S	6	38.1000
9	29,485	Catherine	Abel	71,782	865	Classic Vest, M	4	38.1000
10	29,485	Catherine	Abel	71,782	870	Water Bottle - 30 c	6	2.9940
11	29,485	Catherine	Abel	71,782	876	Hitch Rack - 4-Bike	3	72.0000
12	29,485	Catherine	Abel	71,782	877	Bike Wash - Dissoh	10	4.7700
13	29,485	Catherine	Abel	71,782	880	Hydration Pack - 7l	9	32.9940
14	29,485	Catherine	Abel	71,782	881	Short-Sleeve Classi	6	32.3940
15	29,485	Catherine	Abel	71,782	883	Short-Sleeve Classi	4	32.3940
16	29,485	Catherine	Abel	71,782	884	Short-Sleeve Classi	4	32.3940

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;

```

dbo SalesLT *<db-03feb-aditya> Script x

```

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;

```

Results 1 x

SELECT c.CustomerID, c.FirstName, c.LastName, p.ProductID, p.Name AS ProductName, SUM(od.OrderQty) AS TotalQuantity

Grid	CustomerID	FirstName	LastName	ProductID	ProductName	TotalQuantity
1	29485	Catherine	Abel	707	Sport-100 Helmet, Red	3
2	29485	Catherine	Abel	708	Sport-100 Helmet, Black	7
3	29485	Catherine	Abel	711	Sport-100 Helmet, Blue	6
4	29485	Catherine	Abel	712	AWC Logo Cap	10
5	29485	Catherine	Abel	714	Long-Sleeve Logo Jersey, M	3
6	29485	Catherine	Abel	715	Long-Sleeve Logo Jersey, L	8
7	29485	Catherine	Abel	859	Half-Finger Gloves, M	1
8	29485	Catherine	Abel	864	Classic Vest, S	6
9	29485	Catherine	Abel	865	Classic Vest, M	4
10	29485	Catherine	Abel	870	Water Bottle - 30 oz.	6
11	29485	Catherine	Abel	876	Hitch Rack - 4-Bike	3
12	29485	Catherine	Abel	877	Bike Wash - Dissolver	10
13	29485	Catherine	Abel	880	Hydration Pack - 70 oz.	9
14	29485	Catherine	Abel	881	Short-Sleeve Classic Jersey, S	6
15	29485	Catherine	Abel	883	Short-Sleeve Classic Jersey, L	4

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 CustomerPurchaseCounts pc ON c.CustomerID = pc.CustomerID
WHERE pc.PurchaseCount > (SELECT AVG(PurchaseCount)
FROM CustomerPurchaseCounts);

```

dbo SalesLT *<db-03feb-aditya> Script x

```

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 CustomerPurchaseCounts pc ON c.CustomerID = pc.CustomerID
WHERE pc.PurchaseCount > (SELECT AVG(PurchaseCount)
FROM CustomerPurchaseCounts);

```

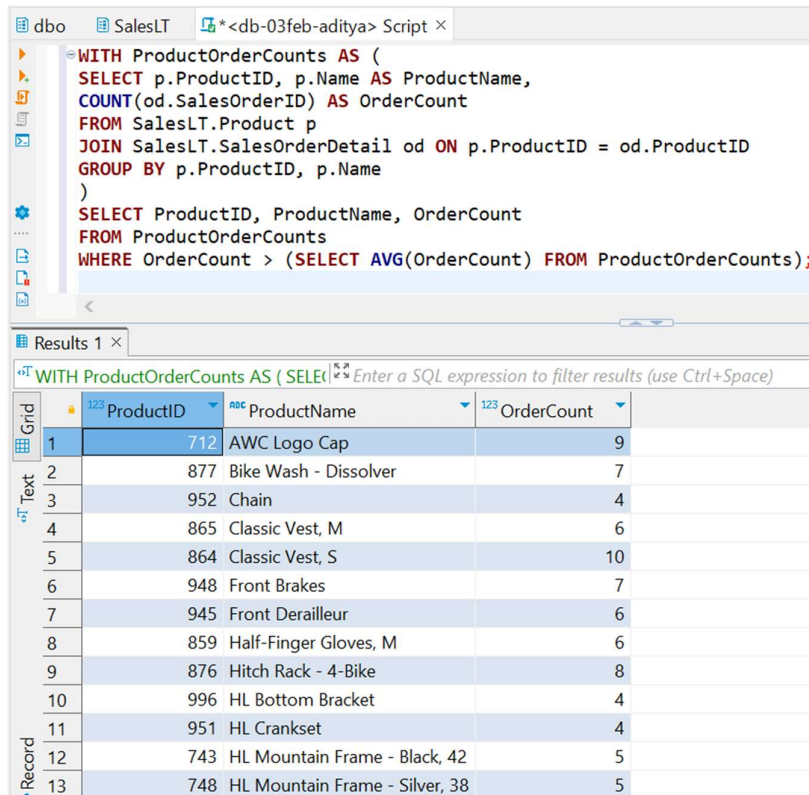
Results 1 x

WITH CustomerPurchaseCounts AS (

Grid	CustomerID	FirstName	LastName	EmailAddress	Phone	CompanyName
Text						

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 Server Enterprise Manager window with a query editor and a results grid. The query editor contains the following SQL code:

```
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 results grid 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	HL Bottom Bracket	4
951	HL Crankset	4
743	HL Mountain Frame - Black, 42	5
748	HL Mountain Frame - Silver, 38	5

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  
);
```

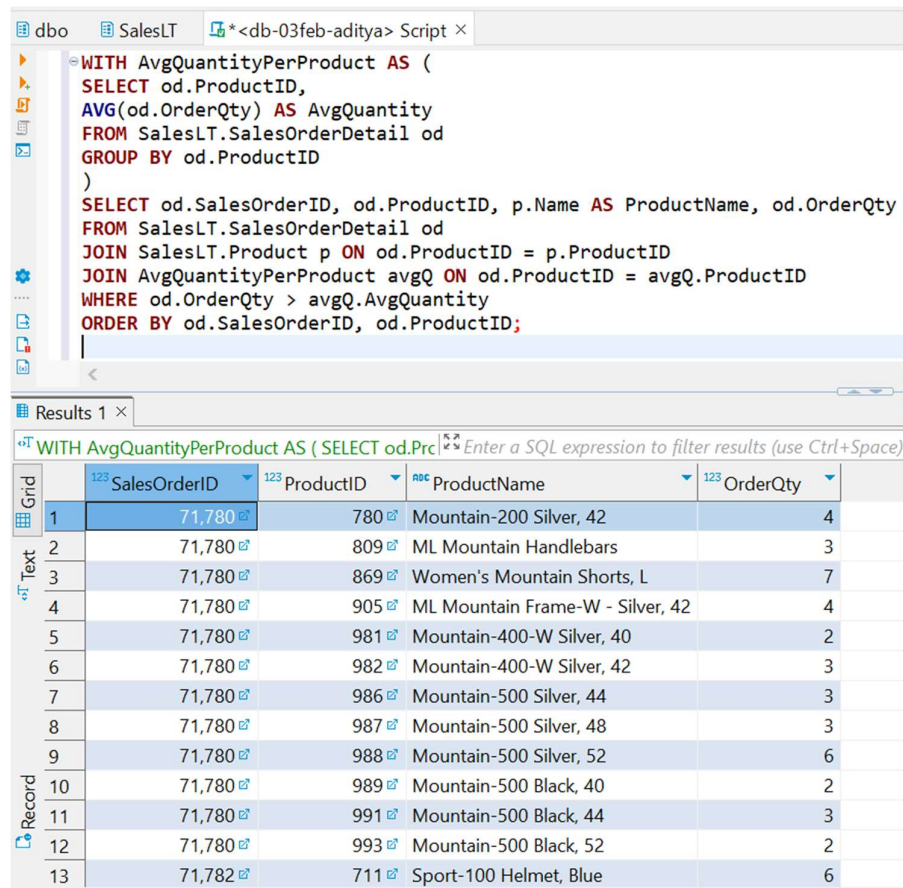
The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a script window with the following SQL query:

```
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  
);
```

The bottom pane shows the results of the query in a grid format. The columns are: ProductID, ProductName, SalesOrderID, CustomerID, AddressID, AddressLine1, AddressLine2, City, StateProvince, CountryRegion, and PostalCode. The results are currently empty.

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 AvgQuantityPerProduct avgQ ON od.ProductID = avgQ.ProductID  
WHERE od.OrderQty > avgQ.AvgQuantity  
ORDER BY od.SalesOrderID, od.ProductID;
```



The screenshot shows a SQL Server Enterprise Manager interface. The top pane displays a T-SQL query that uses a Common Table Expression (CTE) to calculate the average order quantity per product and then filters for orders where the quantity is greater than the average. The bottom pane shows the results of the query, which are displayed in a grid. The grid has four columns: SalesOrderID, ProductID, ProductName, and OrderQty. The results show 13 rows of data, including various mountain bikes and a helmet.

	SalesOrderID	ProductID	ProductName	OrderQty
1	71,780	780	Mountain-200 Silver, 42	4
2	71,780	809	ML Mountain Handlebars	3
3	71,780	869	Women's Mountain Shorts, L	7
4	71,780	905	ML Mountain Frame-W - Silver, 42	4
5	71,780	981	Mountain-400-W Silver, 40	2
6	71,780	982	Mountain-400-W Silver, 42	3
7	71,780	986	Mountain-500 Silver, 44	3
8	71,780	987	Mountain-500 Silver, 48	3
9	71,780	988	Mountain-500 Silver, 52	6
10	71,780	989	Mountain-500 Black, 40	2
11	71,780	991	Mountain-500 Black, 44	3
12	71,780	993	Mountain-500 Black, 52	2
13	71,782	711	Sport-100 Helmet, Blue	6

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;

```

The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a script window titled '<db-03feb-aditya> Script' containing the following SQL query:

```

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;

```

The bottom pane shows the 'Results 1' window. It contains a table with the following columns: CustomerID, FirstName, LastName, SalesOrderID, ProductID, and ProductName. The table is currently empty, and the 'Grid' view is selected.

CustomerID	FirstName	LastName	SalesOrderID	ProductID	ProductName