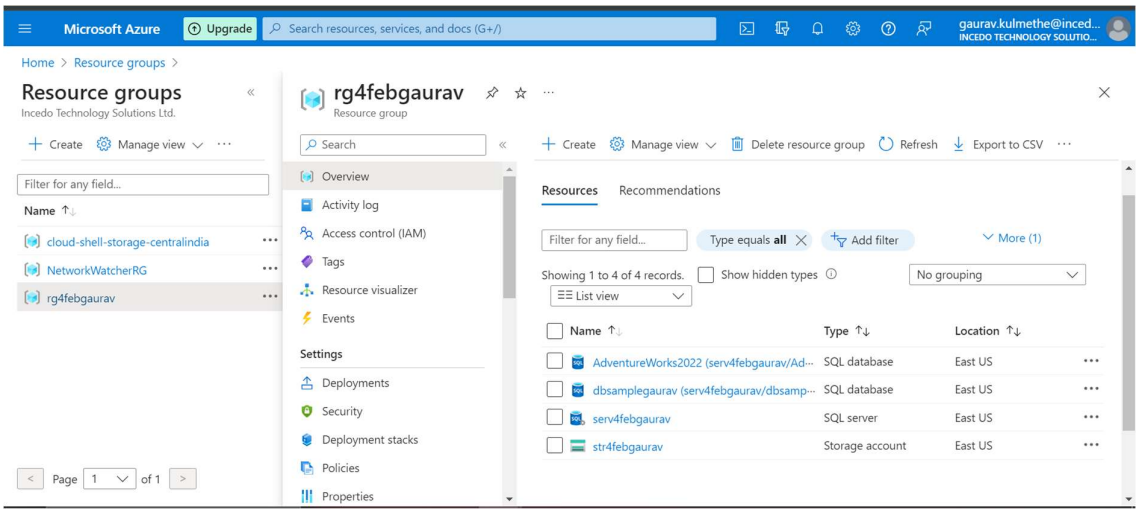
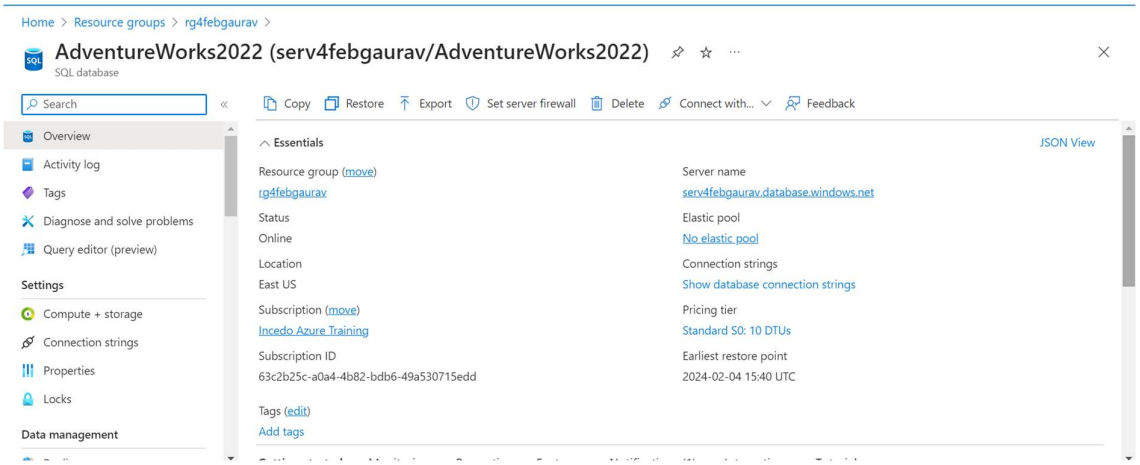
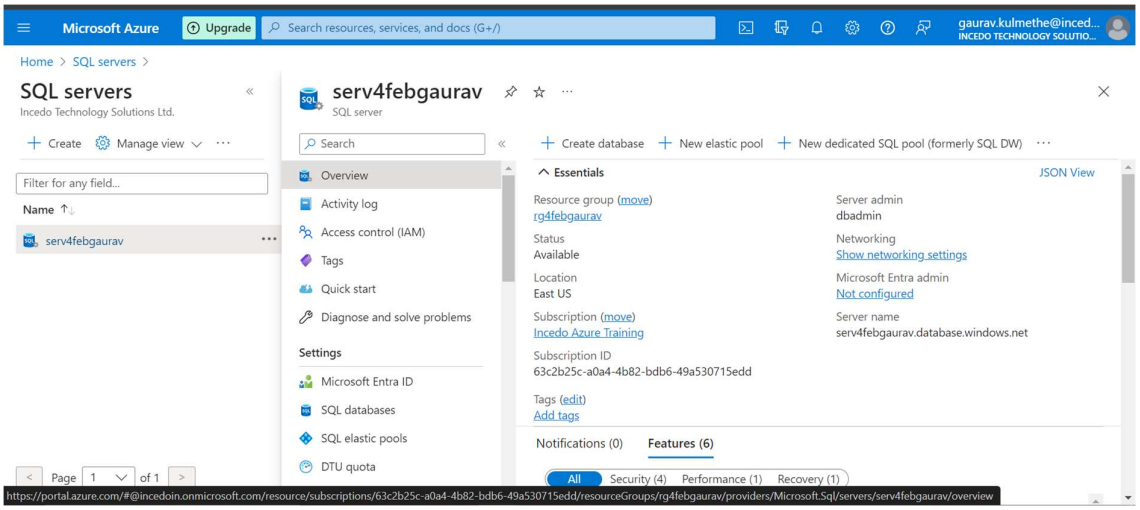
SQL Project

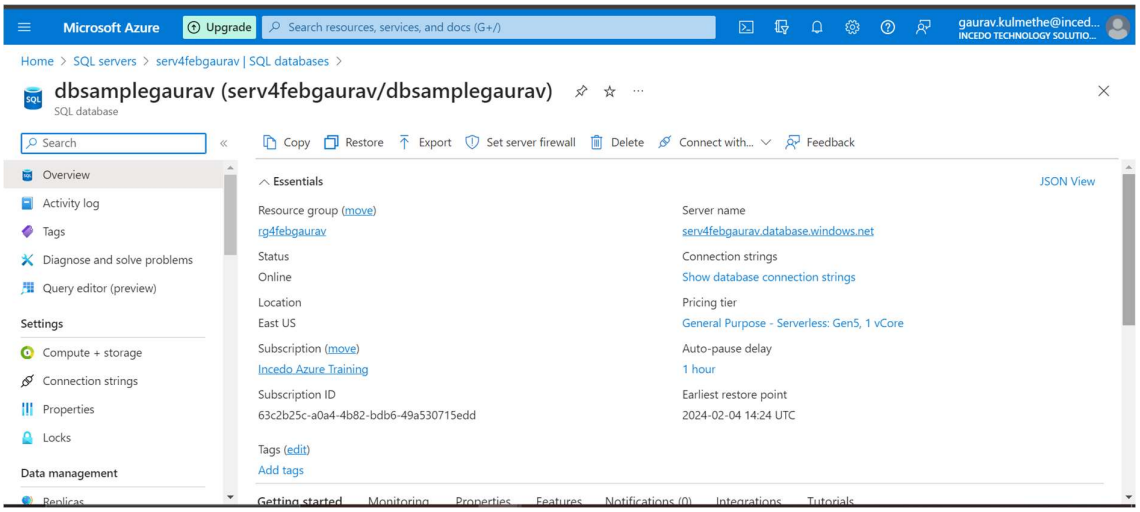
Submitted by – Gaurav Kulmethe

Module 1 :









Module 2: Perform T-SQL Operations on Restored Database (50 Marks)

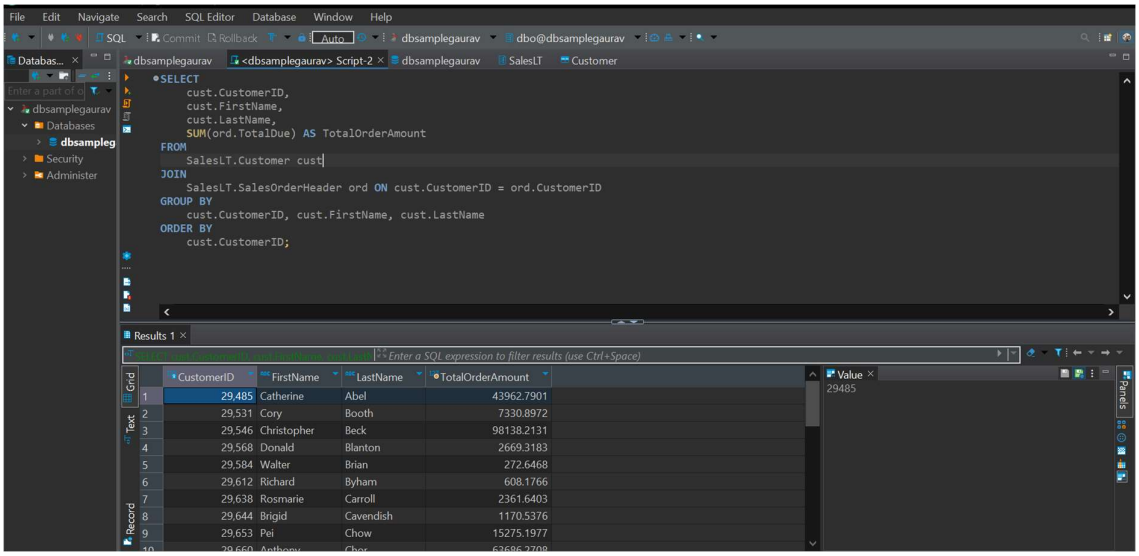
Basic = 50 Marks Below are 30 scenario questions we have you need to select any 25 and solve them, that cover various advanced SQL concepts like joins, subqueries, unions, and more. make sure you have the database setup before attempting these queries.

Joins:

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

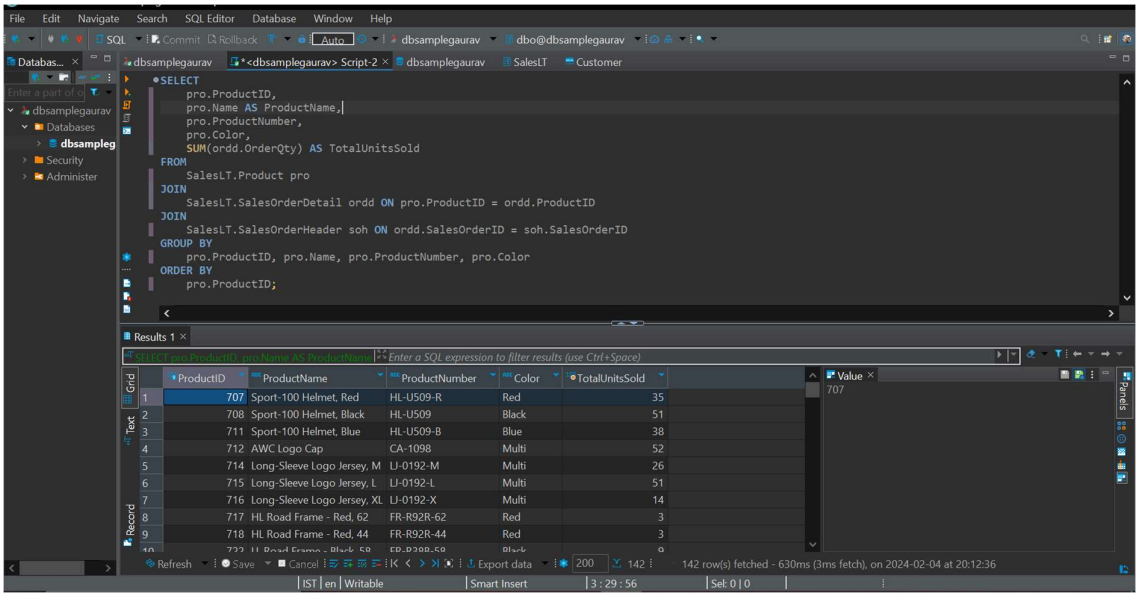
```
SELECT
    cust.CustomerID,
    cust.FirstName,
    cust.LastName,
    SUM(ord.TotalDue) AS TotalOrderAmount
```

```
FROM
   SalesLT.Customer cust
JOIN
   SalesLT.SalesOrderHeader ord ON cust.CustomerID = ord.CustomerID
GROUP BY
   cust.CustomerID, cust.FirstName, cust.LastName
ORDER BY
   cust.CustomerID;
```



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

```
SELECT
   pro.ProductID,
   pro.Name AS ProductName,
   pro.ProductNumber,
   pro.Color,
   SUM(ordd.OrderQty) AS TotalUnitsSold
FROM
   SalesLT.Product pro
JOIN
   SalesLT.SalesOrderDetail ordd ON pro.ProductID = ordd.ProductID
JOIN
   SalesLT.SalesOrderHeader soh ON ordd.SalesOrderID = soh.SalesOrderID
GROUP BY
   pro.ProductID, pro.Name, pro.ProductNumber, pro.Color
ORDER BY
   pro.ProductID;
```

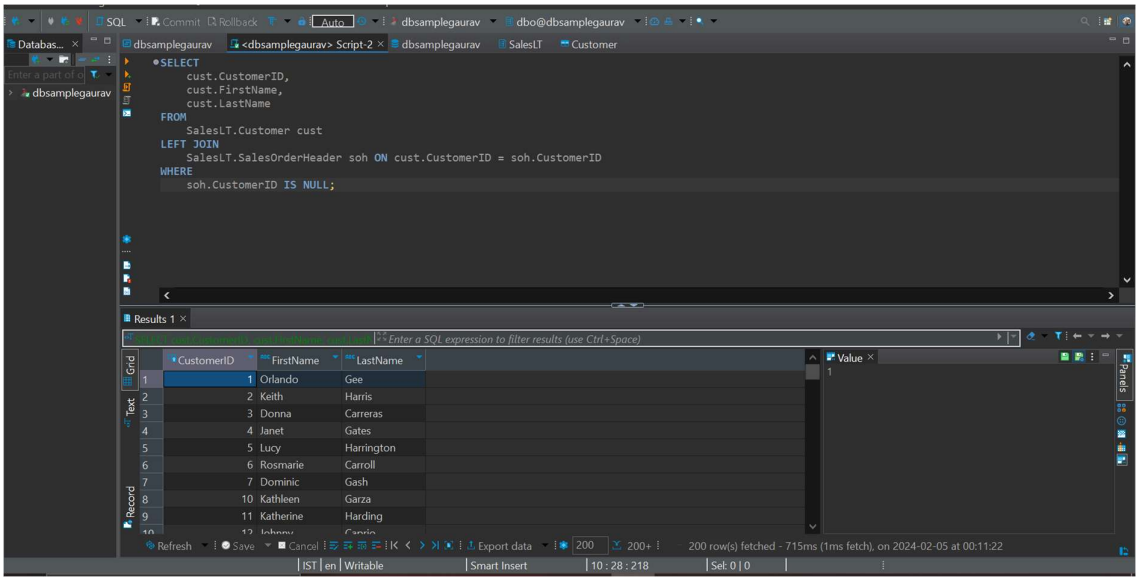


3. Find employees who have the same manager

Data not available

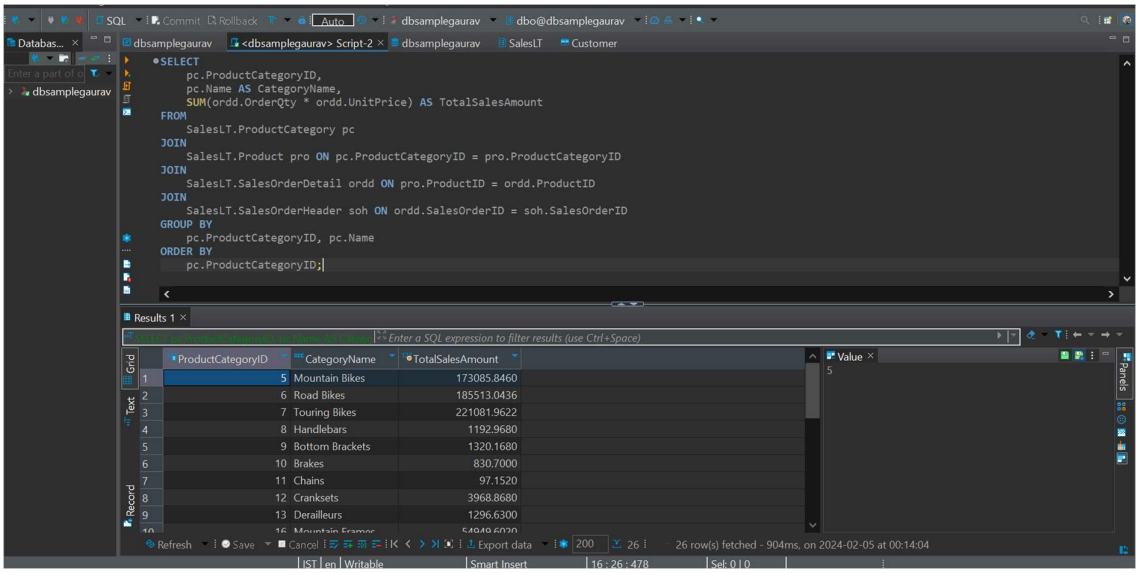
4. List all customers who have never placed an order.

```
SELECT
   cust.CustomerID,
   cust.FirstName,
   cust.LastName
FROM
   SalesLT.Customer cust
LEFT JOIN
   SalesLT.SalesOrderHeader soh ON cust.CustomerID = soh.CustomerID
WHERE
   soh.CustomerID IS NULL;
```



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

```
SELECT
   pc.ProductCategoryID,
   pc.Name AS CategoryName,
   SUM(ordd.OrderQty * ordd.UnitPrice) AS TotalSalesAmount
FROM
   SalesLT.ProductCategory pc
JOIN
   SalesLT.Product pro ON pc.ProductCategoryID = pro.ProductCategoryID
JOIN
   SalesLT.SalesOrderDetail ordd ON pro.ProductID = ordd.ProductID
JOIN
   SalesLT.SalesOrderHeader soh ON ordd.SalesOrderID = soh.SalesOrderID
GROUP BY
   pc.ProductCategoryID, pc.Name
ORDER BY
   pc.ProductCategoryID;
```

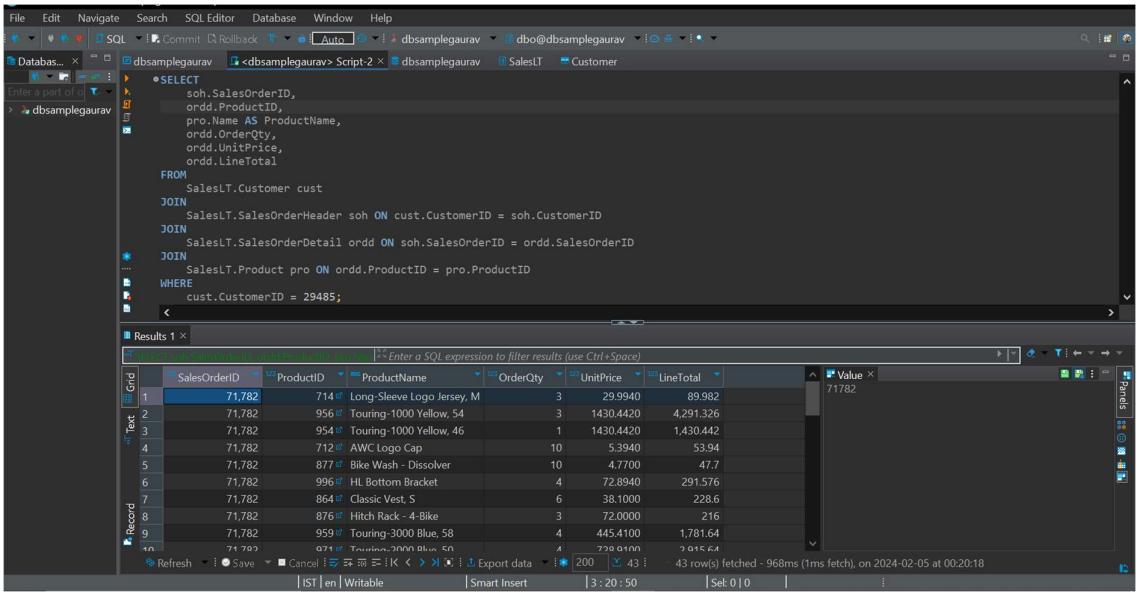


6. Display the names of employees and their direct managers

Data not available

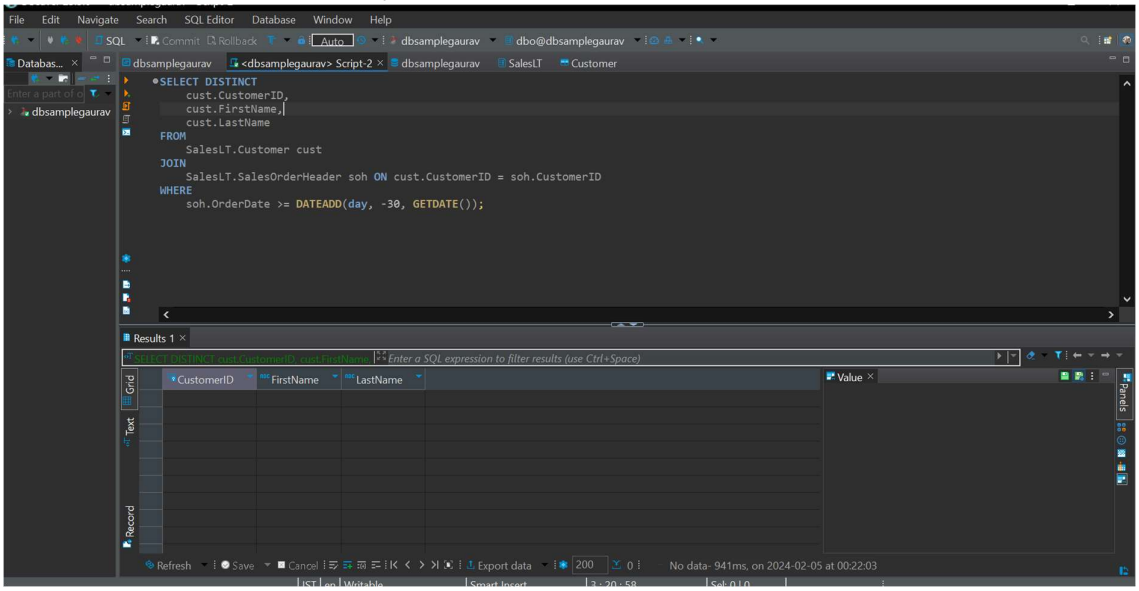
7. Show the order details with product names for a specific customer

```
SELECT
   soh.SalesOrderID,
   ordd.ProductID,
   pro.Name AS ProductName,
   ordd.OrderQty,
   ordd.UnitPrice,
   ordd.LineTotal
FROM
   SalesLT.Customer cust
JOIN
   SalesLT.SalesOrderHeader soh ON cust.CustomerID = soh.CustomerID
JOIN
   SalesLT.SalesOrderDetail ordd ON soh.SalesOrderID = ordd.SalesOrderID
JOIN
   SalesLT.Product pro ON ordd.ProductID = pro.ProductID
WHERE
   cust.CustomerID = 29485;
```



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

```
SELECT DISTINCT
   cust.CustomerID,
   cust.FirstName,
   cust.LastName
FROM
   SalesLT.Customer cust
JOIN
   SalesLT.SalesOrderHeader soh ON cust.CustomerID = soh.CustomerID
WHERE
   soh.OrderDate >= DATEADD(day, -30, GETDATE());
```

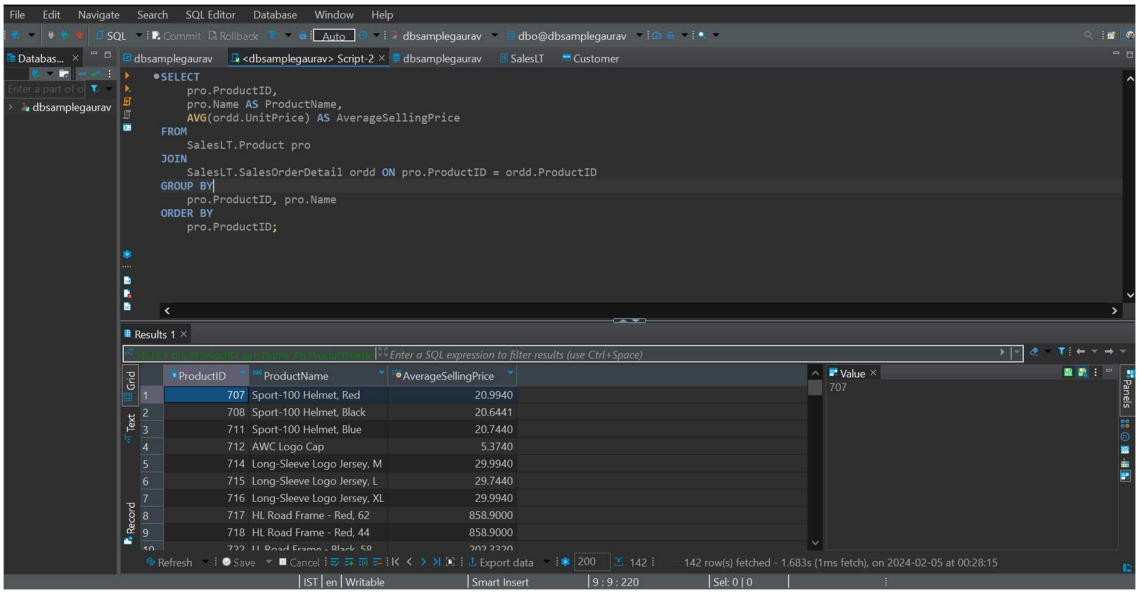


9. **Find employees who do not have any direct reports.**

Data Not available

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

```
SELECT
   pro.ProductID,
   pro.Name AS ProductName,
   AVG(ordd.UnitPrice) AS AverageSellingPrice
FROM
   SalesLT.Product pro
JOIN
   SalesLT.SalesOrderDetail ordd ON pro.ProductID = ordd.ProductID
GROUP BY
   pro.ProductID, pro.Name
ORDER BY
   pro.ProductID;
```

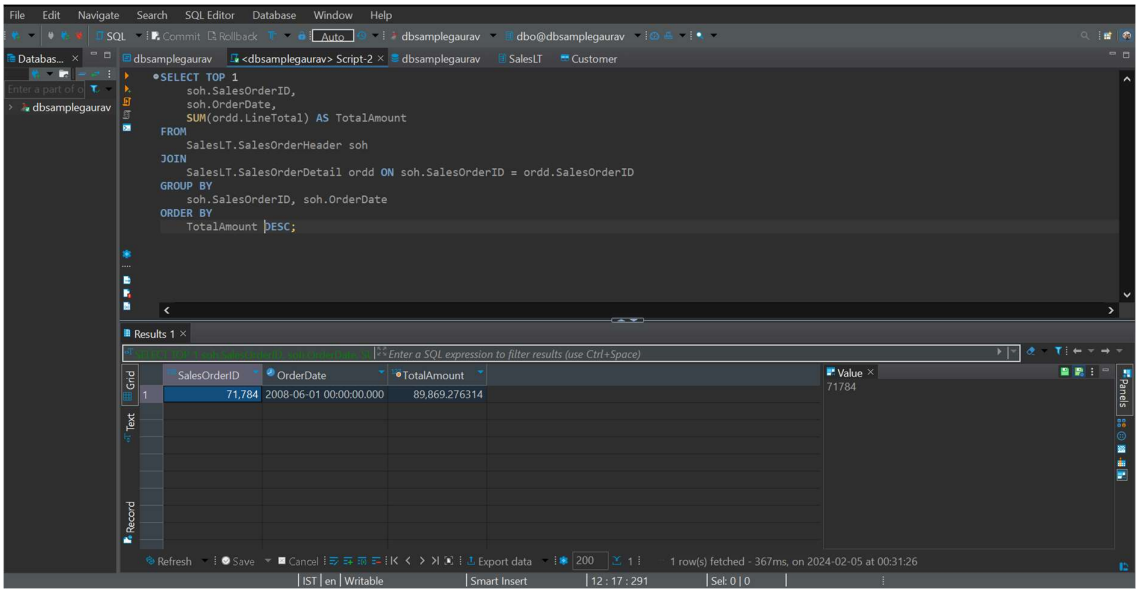


Subqueries

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

```
SELECT TOP 1
   soh.SalesOrderID,
   soh.OrderDate,
   SUM(ordd.LineTotal) AS TotalAmount
FROM
   SalesLT.SalesOrderHeader soh
JOIN
```

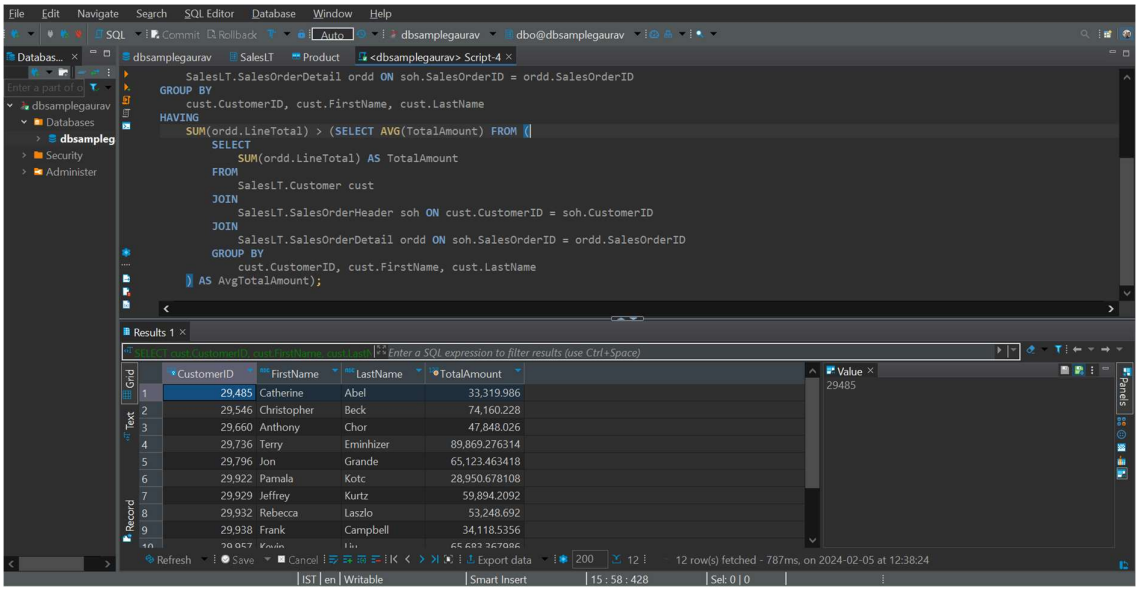
```
  SalesLT.SalesOrderDetail ordd ON soh.SalesOrderID = ordd.SalesOrderID
GROUP BY
  soh.SalesOrderID, soh.OrderDate
ORDER BY
  TotalAmount DESC;
```



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

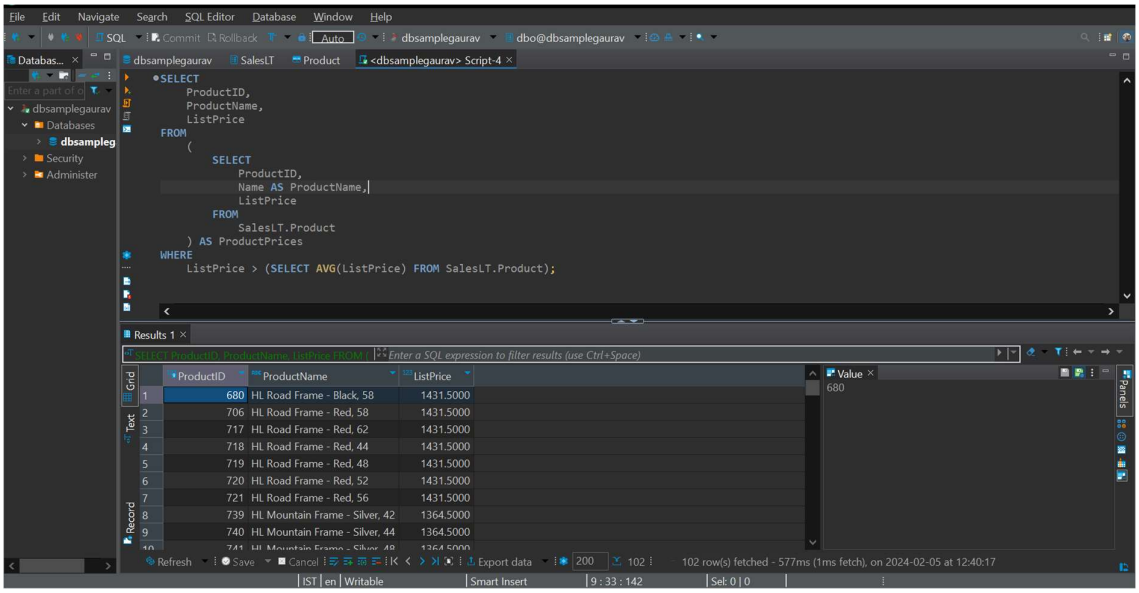
```
SELECT
  cust.CustomerID,
  cust.FirstName,
  cust.LastName,
  SUM(ordd.LineTotal) AS TotalAmount
FROM
  SalesLT.Customer cust
JOIN
  SalesLT.SalesOrderHeader soh ON cust.CustomerID = soh.CustomerID
JOIN
  SalesLT.SalesOrderDetail ordd ON soh.SalesOrderID = ordd.SalesOrderID
GROUP BY
  cust.CustomerID, cust.FirstName, cust.LastName
HAVING
  SUM(ordd.LineTotal) > (SELECT AVG(TotalAmount) FROM (
    SELECT
      SUM(ordd.LineTotal) AS TotalAmount
    FROM
      SalesLT.Customer cust
    JOIN
      SalesLT.SalesOrderHeader soh ON cust.CustomerID = soh.CustomerID
    JOIN
      SalesLT.SalesOrderDetail ordd ON soh.SalesOrderID = ordd.SalesOrderID
```

```
    GROUP BY
        cust.CustomerID, cust.FirstName, cust.LastName
  ) AS AvgTotalAmount);
```



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

```
SELECT
  ProductID,
  ProductName,
  ListPrice
FROM
  (
    SELECT
      ProductID,
      Name AS ProductName,
      ListPrice
    FROM
      SalesLT.Product
  ) AS ProductPrices
WHERE
  ListPrice > (SELECT AVG(ListPrice) FROM SalesLT.Product);
```

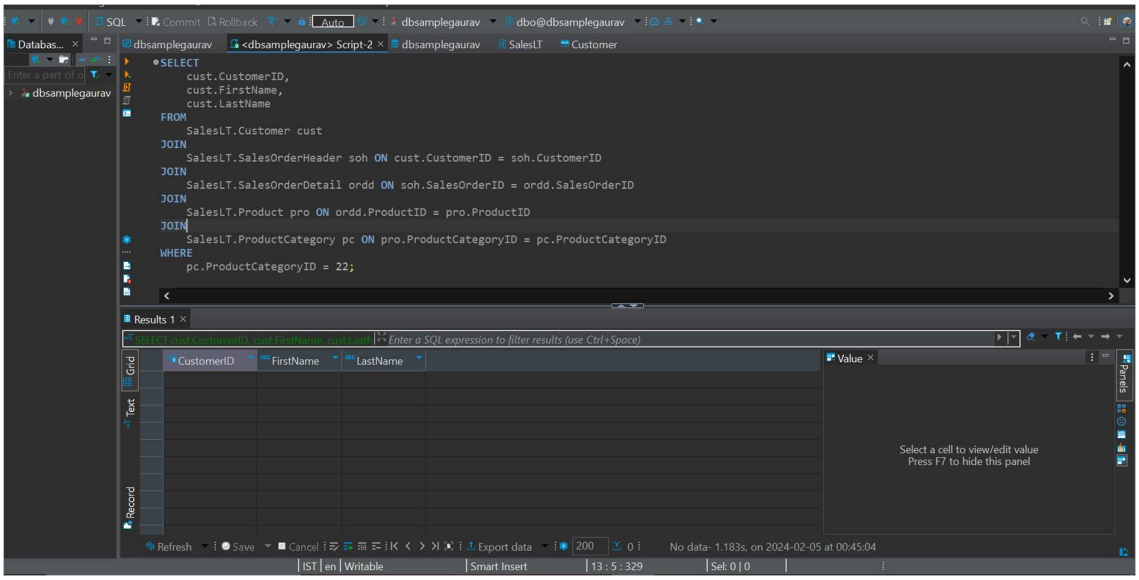


14. Retrieve orders placed by employees who have a specific job title

No data available

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

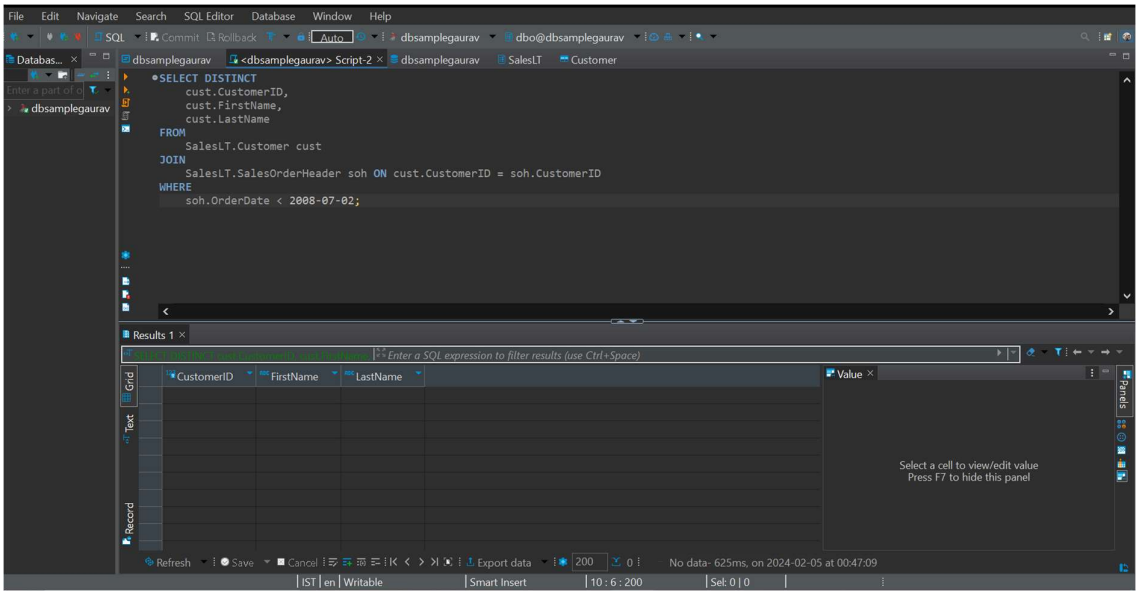
```
SELECT
   cust.CustomerID,
   cust.FirstName,
   cust.LastName
FROM
   SalesLT.Customer cust
JOIN
   SalesLT.SalesOrderHeader soh ON cust.CustomerID = soh.CustomerID
JOIN
   SalesLT.SalesOrderDetail ordd ON soh.SalesOrderID = ordd.SalesOrderID
JOIN
   SalesLT.Product pro ON ordd.ProductID = pro.ProductID
JOIN
   SalesLT.ProductCategory pc ON pro.ProductCategoryID = pc.ProductCategoryID
WHERE
   pc.ProductCategoryID = 22;
```



16. **Find employees with salaries greater than the average salary in their department.**
    Data Not available

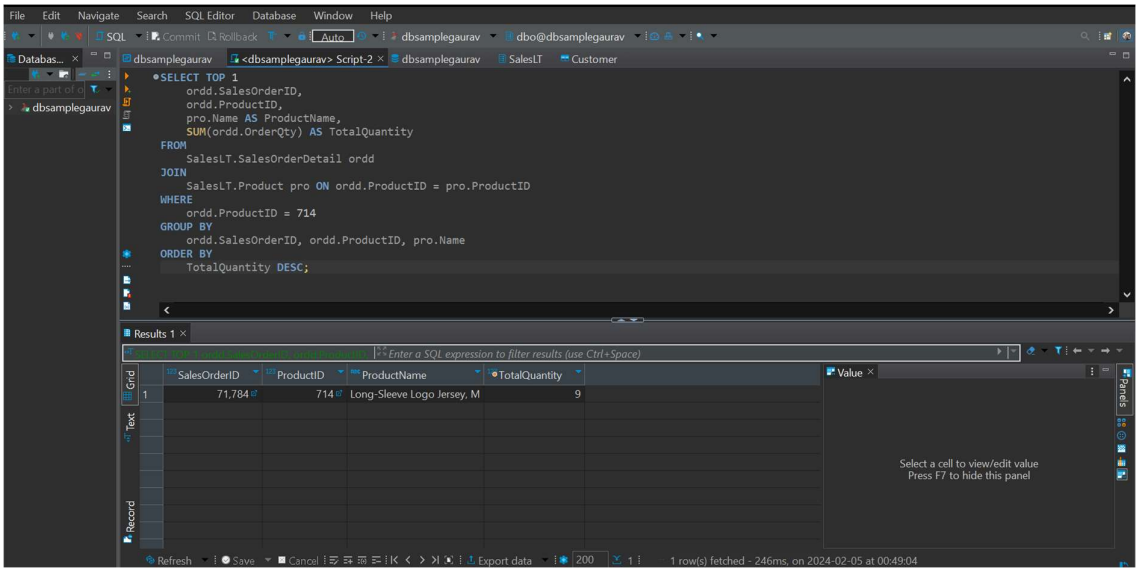
17. **List customers who have placed orders before a specific date.**
    Not enough data

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



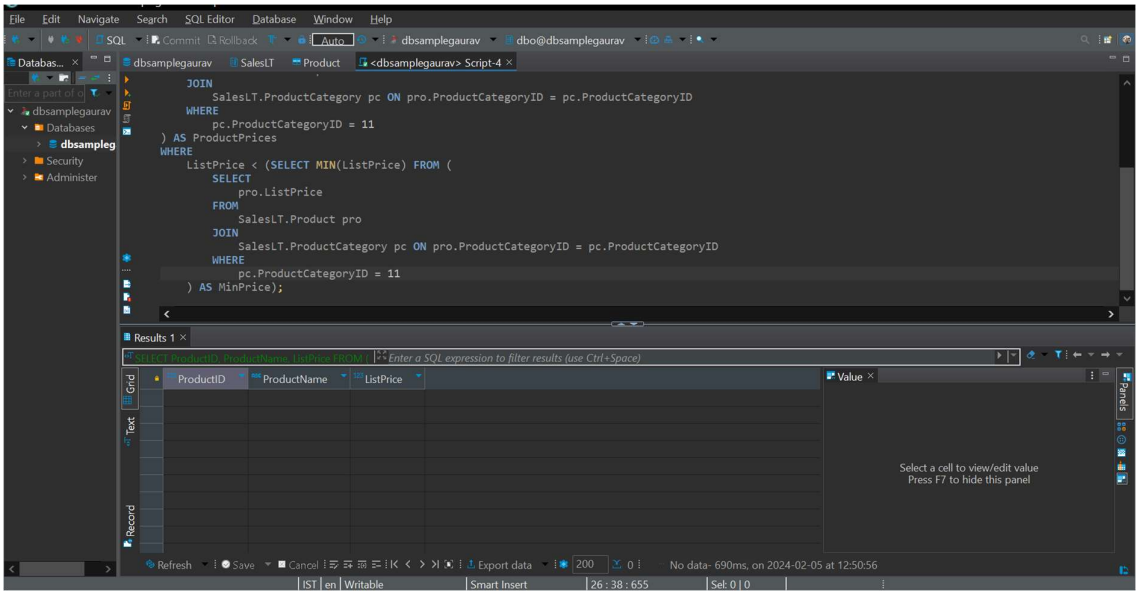
18. Retrieve the order with the highest quantity of a specific product.

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



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

```
SELECT
  ProductID,
  ProductName,
  ListPrice
FROM (
  SELECT
    pro.ProductID,
    pro.Name AS ProductName,
    pro.ListPrice
  FROM
    SalesLT.Product pro
  JOIN
    SalesLT.ProductCategory pc ON pro.ProductCategoryID = pc.ProductCategoryID
  WHERE
    pc.ProductCategoryID = 11
) AS ProductPrices
WHERE
  ListPrice < (SELECT MIN(ListPrice) FROM (
    SELECT
      pro.ListPrice
    FROM
      SalesLT.Product pro
    JOIN
      SalesLT.ProductCategory pc ON pro.ProductCategoryID = pc.ProductCategoryID
    WHERE
      pc.ProductCategoryID = 11
  ) AS MinPrice);
```



20. Find employees who have the same job title as their manager.

Data not available

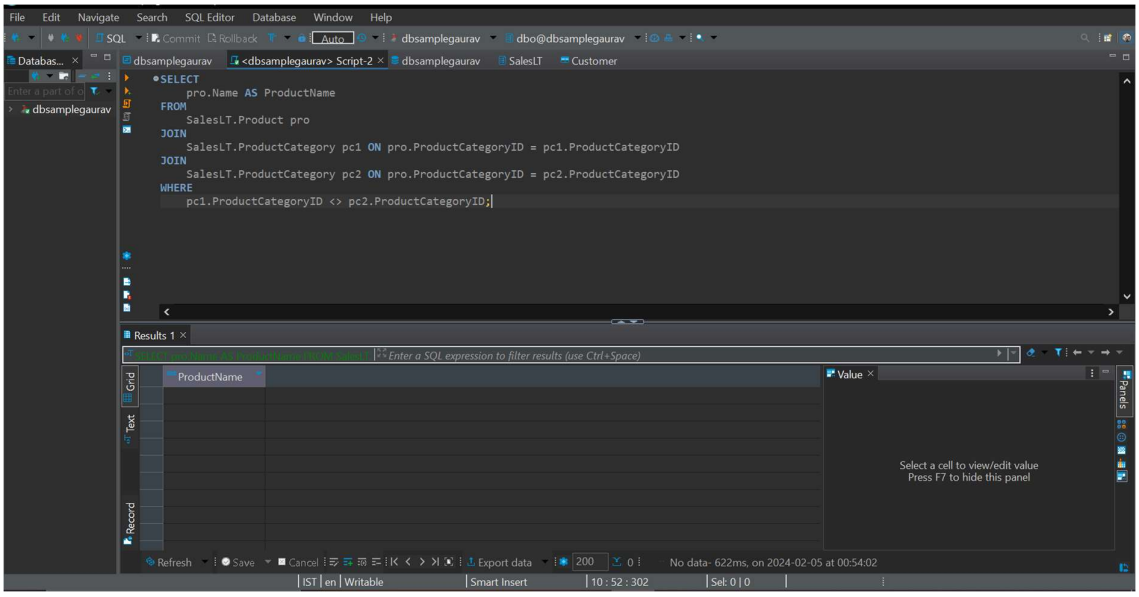
21. Combine results from two queries to get a list of unique customer and employee names.

Data Not Available

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

Not enough data

```
SELECT
   pro.Name AS ProductName
FROM
   SalesLT.Product pro
JOIN
   SalesLT.ProductCategory pc1 ON pro.ProductCategoryID = pc1.ProductCategoryID
JOIN
   SalesLT.ProductCategory pc2 ON pro.ProductCategoryID = pc2.ProductCategoryID
WHERE
   pc1.ProductCategoryID <> pc2.ProductCategoryID;
```



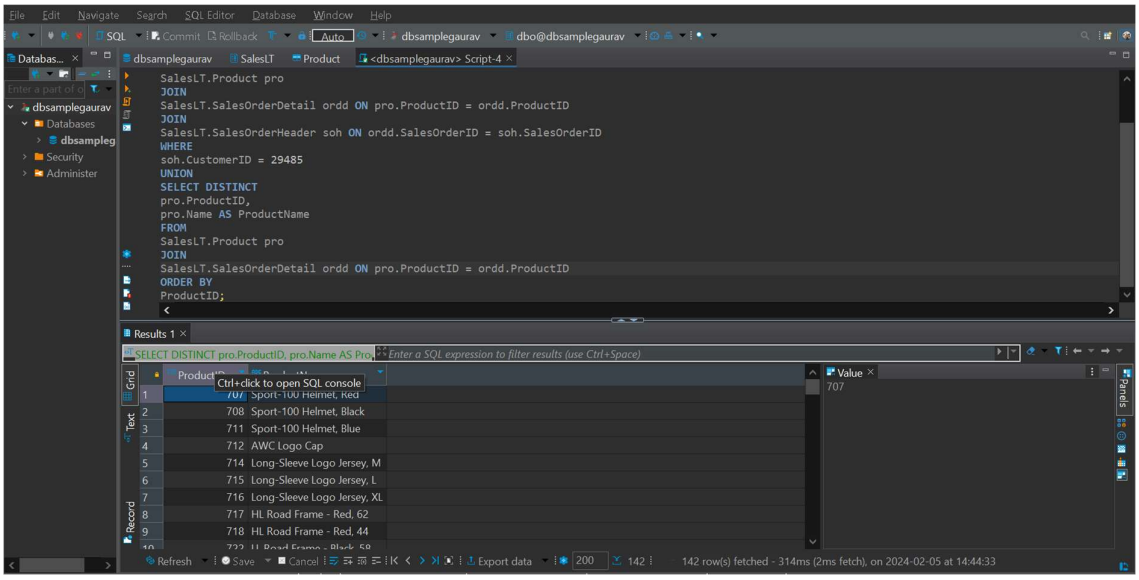
23. **Display the names of employees and customers in a single result set**
    Data insufficient

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
pro.ProductID,
pro.Name AS ProductName
FROM
SalesLT.Product pro
JOIN
SalesLT.SalesOrderDetail ordd ON pro.ProductID = ordd.ProductID
JOIN
SalesLT.SalesOrderHeader soh ON ordd.SalesOrderID = soh.SalesOrderID
WHERE
soh.CustomerID = 29485
UNION
SELECT DISTINCT
pro.ProductID,
pro.Name AS ProductName
FROM
SalesLT.Product pro
JOIN
SalesLT.SalesOrderDetail ordd ON pro.ProductID = ordd.ProductID
```

ORDER BY
ProductID;



26. **Retrieve orders placed by customers and employees in a single result set**
    Data not available

27. **Display products that are either in a specific category or have a specific safety stock level.**
    Data Not Available

28. **List customers who have placed orders and employees who have direct reports in a single result set.**
    Data Not Available

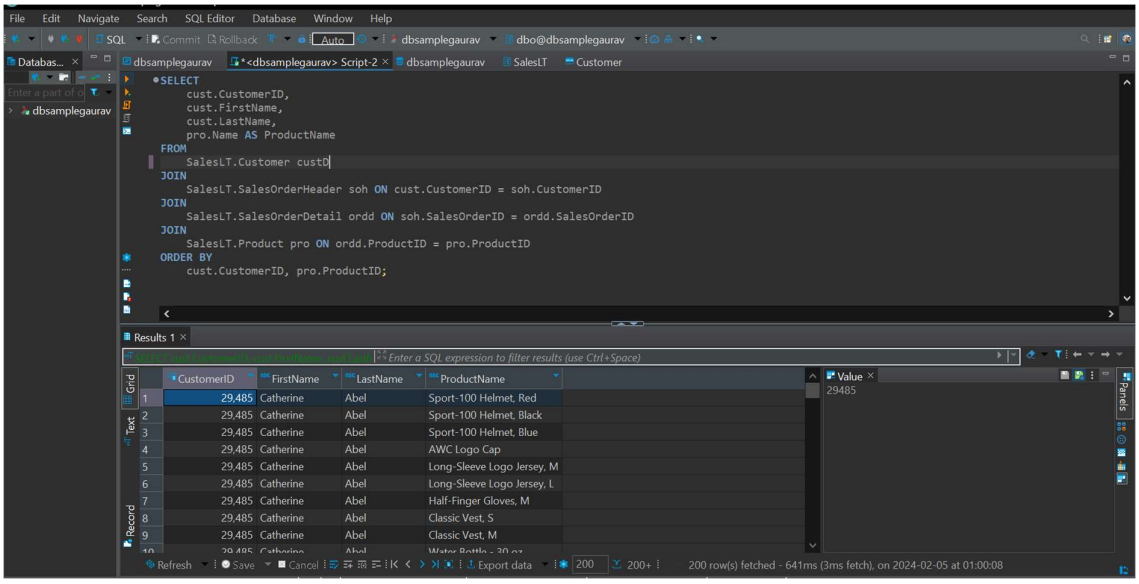
29. **.Retrieve products that are in stock in one location and out of stock in another.**
    Data Not Available

30. **Combine information about employees who are managers and employees who have managers**
    Data Not available

Joins:

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

```
SELECT
   cust.CustomerID,
   cust.FirstName,
   cust.LastName,
   pro.Name AS ProductName
FROM
   SalesLT.Customer cust
JOIN
   SalesLT.SalesOrderHeader soh ON cust.CustomerID = soh.CustomerID
JOIN
   SalesLT.SalesOrderDetail ordd ON sroh.SalesOrderID = ordd.SalesOrderID
JOIN
   SalesLT.Product pro ON ordd.ProductID = pro.ProductID
ORDER BY
   cust.CustomerID, pro.ProductID;
```



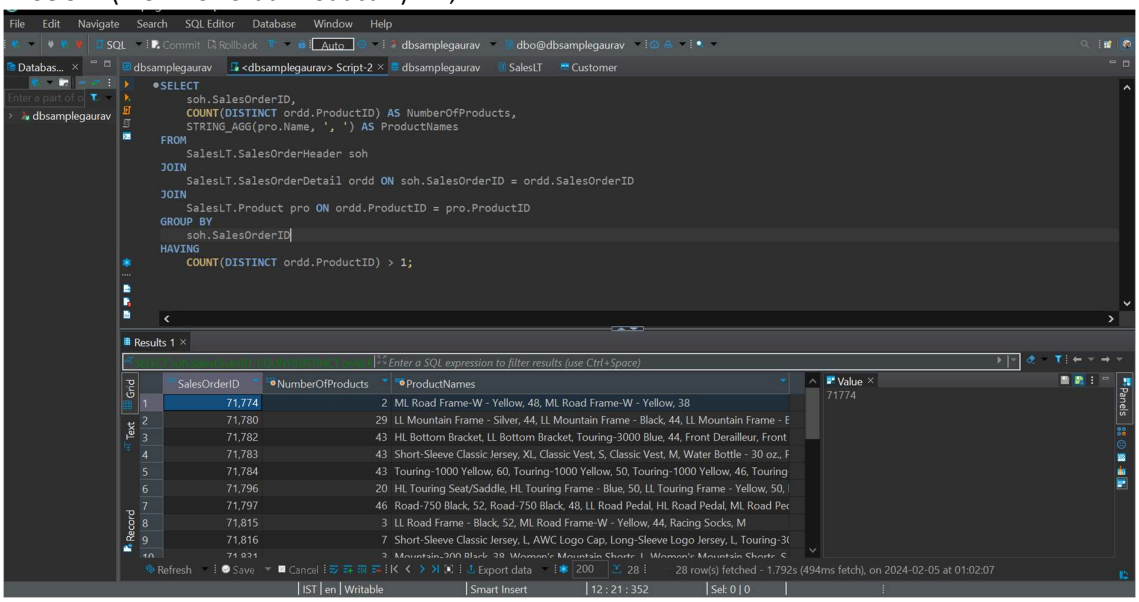
32. Display employees who have the same manager, including indirect reports.

Data not available

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

```
SELECT
   soh.SalesOrderID,
   COUNT(DISTINCT ordd.ProductID) AS NumberOfProducts,
   STRING_AGG(pro.Name, ', ') AS ProductNames
FROM
```

```
  SalesLT.SalesOrderHeader soh
JOIN
  SalesLT.SalesOrderDetail ordd ON soh.SalesOrderID = ordd.SalesOrderID
JOIN
  SalesLT.Product pro ON ordd.ProductID = pro.ProductID
GROUP BY
  soh.SalesOrderID
HAVING
  COUNT(DISTINCT ordd.ProductID) > 1;
```



34. .List customers along with the names of the salespeople who handled their orders.

Data not available

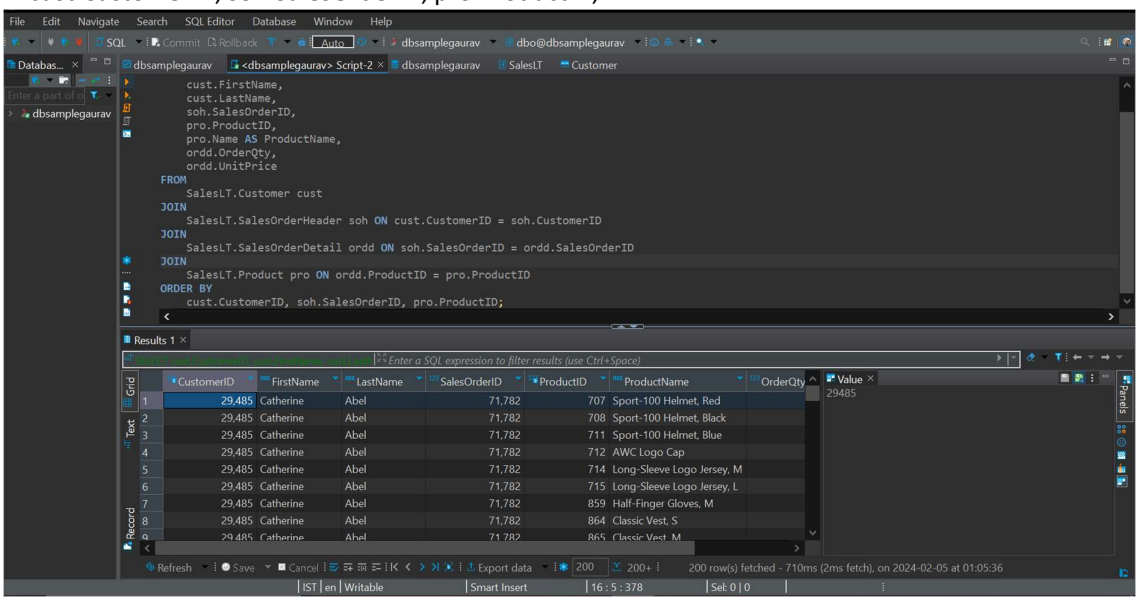
35. Retrieve a list of products along with the names of suppliers.

Data Not available

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

```
SELECT
  cust.CustomerID,
  cust.FirstName,
  cust.LastName,
  soh.SalesOrderID,
  pro.ProductID,
  pro.Name AS ProductName,
  ordd.OrderQty,
  ordd.UnitPrice
FROM
```

```
    SalesLT.Customer cust
JOIN
    SalesLT.SalesOrderHeader soh ON cust.CustomerID = soh.CustomerID
JOIN
    SalesLT.SalesOrderDetail ordd ON soh.SalesOrderID = ordd.SalesOrderID
JOIN
    SalesLT.Product pro ON ordd.ProductID = pro.ProductID
ORDER BY
    cust.CustomerID, soh.SalesOrderID, pro.ProductID;
```



37. Find orders where multiple employees were involved, showing the employee names

Data Not Available

38. List products that have similar names but belong to different categories.

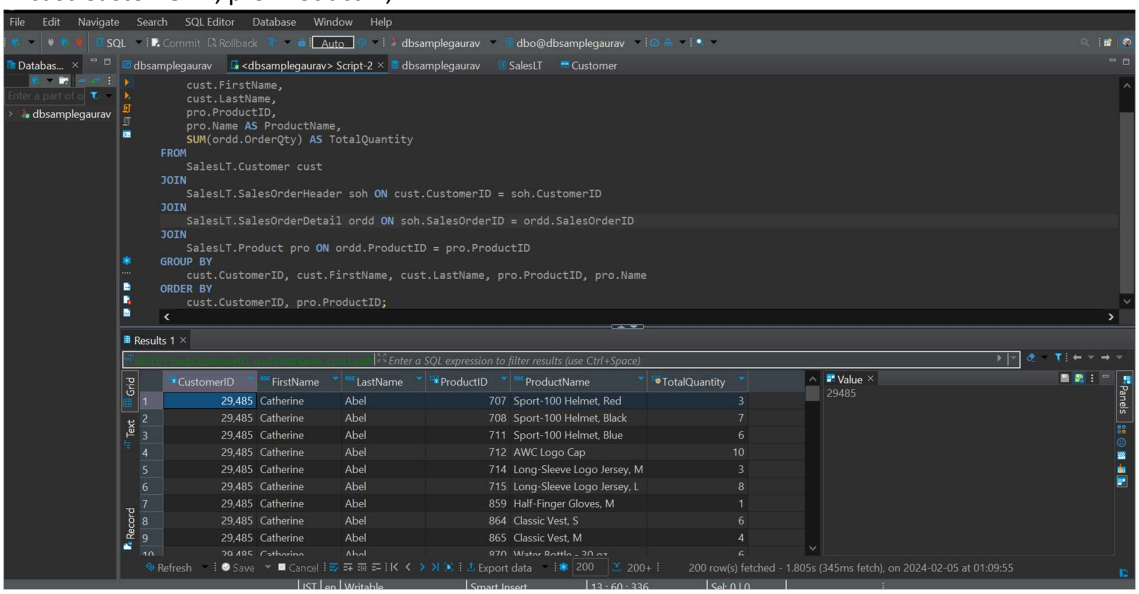
Not Enough data

39. .Retrieve a list of employees along with their training courses and training dates

Data not available

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

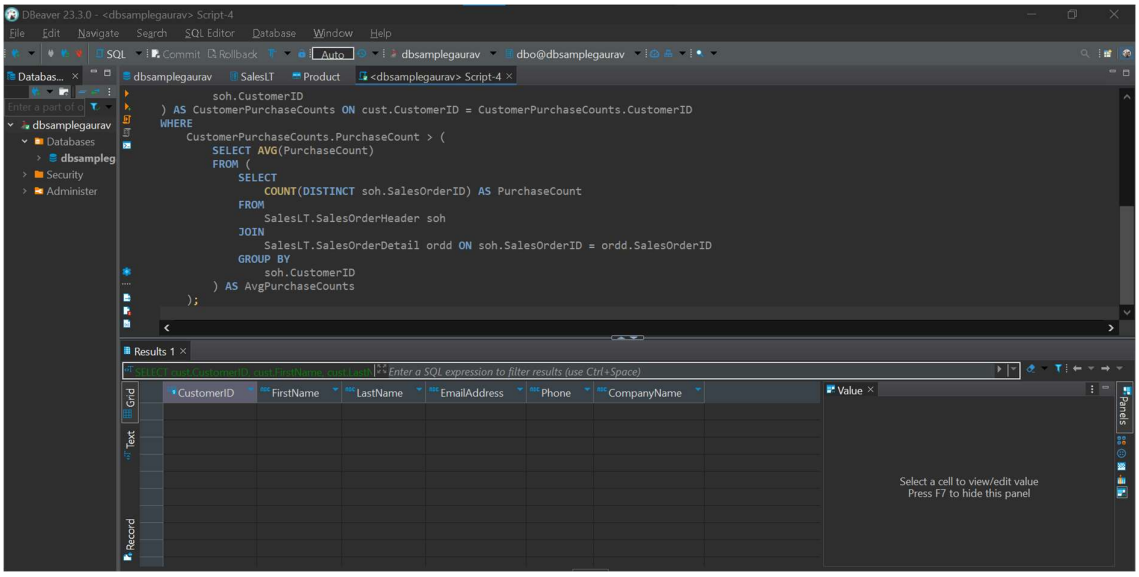
```
SELECT
   cust.CustomerID,
   cust.FirstName,
   cust.LastName,
   pro.ProductID,
   pro.Name AS ProductName,
   SUM(ordd.OrderQty) AS TotalQuantity
FROM
   SalesLT.Customer cust
JOIN
   SalesLT.SalesOrderHeader soh ON cust.CustomerID = soh.CustomerID
JOIN
   SalesLT.SalesOrderDetail ordd ON soh.SalesOrderID = ordd.SalesOrderID
JOIN
   SalesLT.Product pro ON ordd.ProductID = pro.ProductID
GROUP BY
   cust.CustomerID, cust.FirstName, cust.LastName, pro.ProductID, pro.Name
ORDER BY
   cust.CustomerID, pro.ProductID;
```



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

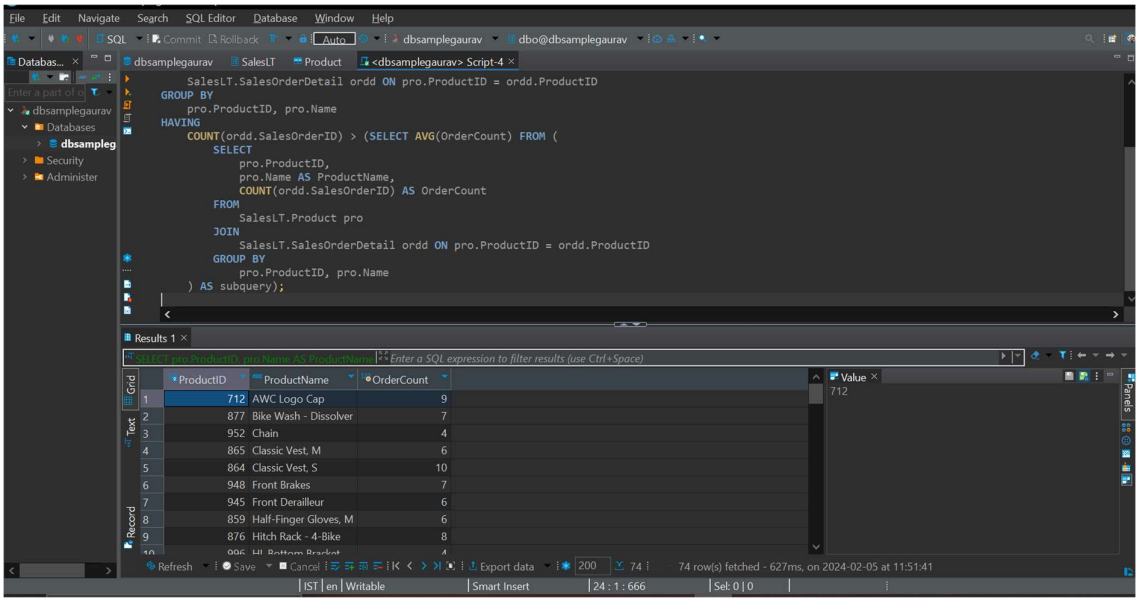
```
SELECT
   cust.CustomerID,
   cust.FirstName,
   cust.LastName,
   cust.EmailAddress,
   cust.Phone,
   cust.CompanyName
```

```
FROM
  SalesLT.Customer cust
JOIN (
  SELECT
    soh.CustomerID,
    COUNT(DISTINCT soh.SalesOrderID) AS PurchaseCount
  FROM
    SalesLT.SalesOrderHeader soh
  JOIN
    SalesLT.SalesOrderDetail ordd ON soh.SalesOrderID = ordd.SalesOrderID
  GROUP BY
    soh.CustomerID
) AS CustomerPurchaseCounts ON cust.CustomerID = CustomerPurchaseCounts.CustomerID
WHERE
  CustomerPurchaseCounts.PurchaseCount > (
    SELECT AVG(PurchaseCount)
    FROM (
      SELECT
        COUNT(DISTINCT soh.SalesOrderID) AS PurchaseCount
      FROM
        SalesLT.SalesOrderHeader soh
      JOIN
        SalesLT.SalesOrderDetail ordd ON soh.SalesOrderID = ordd.SalesOrderID
      GROUP BY
        soh.CustomerID
    ) AS AvgPurchaseCounts
  );
```



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

```
SELECT
   pro.ProductID,
   pro.Name AS ProductName,
   COUNT(ordd.SalesOrderID) AS OrderCount
FROM
   SalesLT.Product pro
JOIN
   SalesLT.SalesOrderDetail ordd ON pro.ProductID = ordd.ProductID
GROUP BY
   pro.ProductID, pro.Name
HAVING
   COUNT(ordd.SalesOrderID) > (SELECT AVG(OrderCount) FROM (
      SELECT
         pro.ProductID,
         pro.Name AS ProductName,
         COUNT(ordd.SalesOrderID) AS OrderCount
      FROM
         SalesLT.Product pro
      JOIN
         SalesLT.SalesOrderDetail ordd ON pro.ProductID = ordd.ProductID
      GROUP BY
         pro.ProductID, pro.Name
   ) AS subquery);
```



43. Retrieve orders placed by employees who have completed a specific training course.

Data Not Available

44. **List employees who have a higher salary than at least one employee in another department.**
    Data Not Available

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

46. **Find employees who have the same job title as the employee with the highest salary.**
    Data Not Available

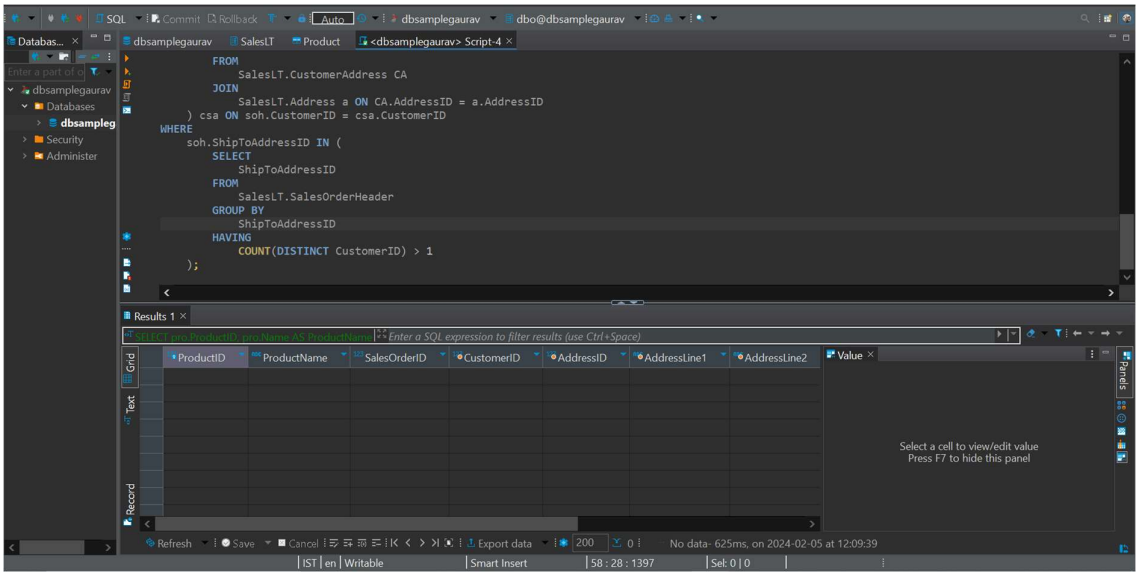
47. **List customers who have placed orders with a total amount greater than the total amount of a specific order.**

    Data insufficient

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

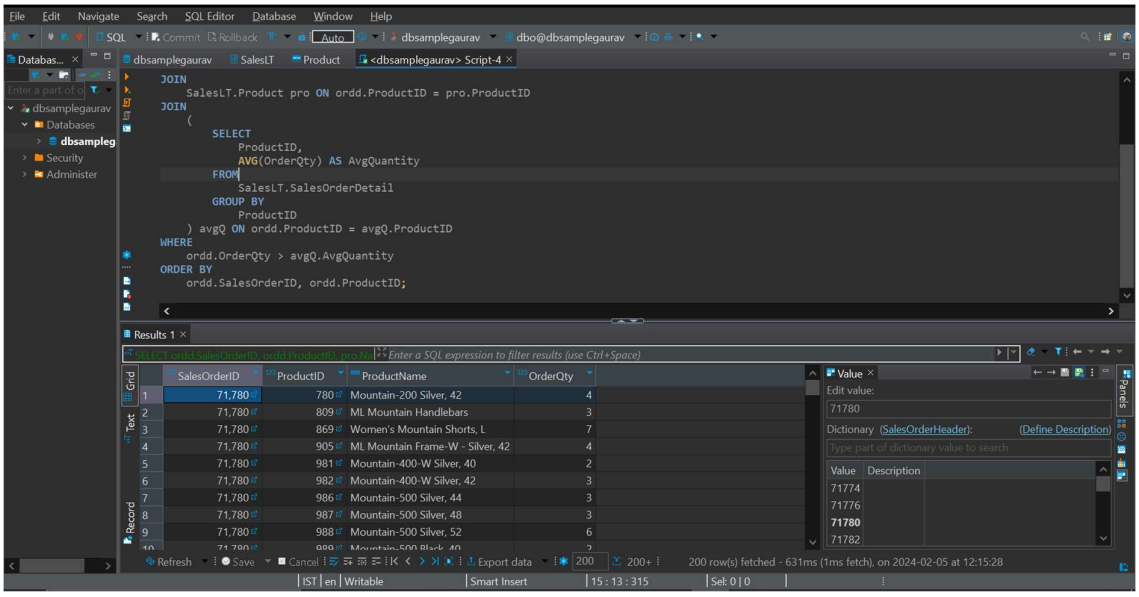
```
WITH CustomerShippingAddresses AS (
  SELECT
    CA.CustomerID,
    CA.AddressID,
    a.AddressLine1,
    a.AddressLine2,
    a.City,
    a.StateProvince,
    a.CountryRegion,
    a.PostalCode
  FROM
    SalesLT.CustomerAddress CA
  JOIN
    SalesLT.Address a ON CA.AddressID = a.AddressID
)

SELECT
  pro.ProductID,
  pro.Name AS ProductName,
  ordd.SalesOrderID,
  csa.CustomerID,
  csa.AddressID,
  csa.AddressLine1,
  csa.AddressLine2,
  csa.City,
  csa.StateProvince,
  csa.CountryRegion,
```

```
   csa.PostalCode
FROM
   SalesLT.Product pro
JOIN
   SalesLT.SalesOrderDetail ordd ON pro.ProductID = ordd.ProductID
JOIN
   SalesLT.SalesOrderHeader soh ON ordd.SalesOrderID = soh.SalesOrderID
JOIN
   (
      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
   ) csa ON soh.CustomerID = csa.CustomerID
WHERE
   soh.ShipToAddressID IN (
      SELECT
         ShipToAddressID
      FROM
         SalesLT.SalesOrderHeader
      GROUP BY
         ShipToAddressID
      HAVING
         COUNT(DISTINCT CustomerID) > 1
   );
```



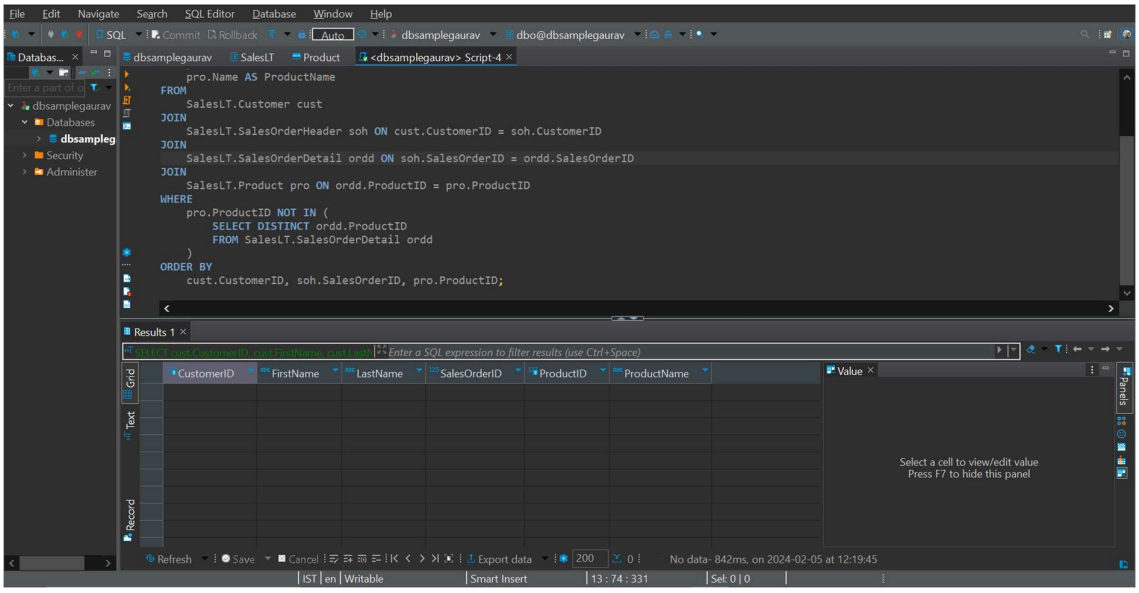
49. Display orders with quantities higher than the average quantity for a specific product.

```
SELECT
   ordd.SalesOrderID,
   ordd.ProductID,
   pro.Name AS ProductName,
   ordd.OrderQty
FROM
   SalesLT.SalesOrderDetail ordd
JOIN
   SalesLT.Product pro ON ordd.ProductID = pro.ProductID
JOIN
   (
      SELECT
         ProductID,
         AVG(OrderQty) AS AvgQuantity
      FROM
         SalesLT.SalesOrderDetail
      GROUP BY
         ProductID
   ) avgQ ON ordd.ProductID = avgQ.ProductID
WHERE
   ordd.OrderQty > avgQ.AvgQuantity
ORDER BY
   ordd.SalesOrderID, ordd.ProductID;
```



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

```
SELECT
   cust.CustomerID,
   cust.FirstName,
   cust.LastName,
   soh.SalesOrderID,
   pro.ProductID,
   pro.Name AS ProductName
FROM
   SalesLT.Customer cust
JOIN
   SalesLT.SalesOrderHeader soh ON cust.CustomerID = soh.CustomerID
JOIN
   SalesLT.SalesOrderDetail ordd ON soh.SalesOrderID = ordd.SalesOrderID
JOIN
   SalesLT.Product pro ON ordd.ProductID = pro.ProductID
WHERE
   pro.ProductID NOT IN (
      SELECT DISTINCT ordd.ProductID
      FROM SalesLT.SalesOrderDetail ordd
   )
ORDER BY
   cust.CustomerID, soh.SalesOrderID, pro.ProductID;
```


File Edit Navigate Search SQL Editor Database Window Help
SQL Commit Rollback Auto dbsamplegaurav dbo@dbsamplegaurav
Databas...
dbsamplegaurav
Databases
dbsampleg
Security
Administer
dbsamplegaurav SalesLT Product <dbsamplegaurav> Script-4
pro.Name AS ProductName
FROM
SalesLT.Customer cust
JOIN
SalesLT.SalesOrderHeader soh ON cust.CustomerID = soh.CustomerID
JOIN
SalesLT.SalesOrderDetail ordd ON soh.SalesOrderID = ordd.SalesOrderID
JOIN
SalesLT.Product pro ON ordd.ProductID = pro.ProductID
WHERE
pro.ProductID NOT IN (
SELECT DISTINCT ordd.ProductID
FROM SalesLT.SalesOrderDetail ordd
)
ORDER BY
cust.CustomerID, soh.SalesOrderID, pro.ProductID;
Results 1
Enter a SQL expression to filter results (use Ctrl+Space)
CustomerID FirstName LastName SalesOrderID ProductID ProductName
Value
Select a cell to view/edit value
Press F7 to hide this panel
Refresh Save Cancel Export data 200 0 No data- 842ms, on 2024-02-05 at 12:19:45
IST en Writable Smart Insert 13 : 74 : 331 Sel: 0 | 0