Analyze Data in a Model Car Database with MySQL Workbench

Project Description:

This project aims to conduct data analysis in the context of the fictional company Mint Classics. I will be playing the role of a novice data analyst assisting the company in addressing inventory and storage facility-related issues. The problem at hand involves the decision to close one of the existing storage facilities.

Demonstrated Skills:

1. Data Analysis: I will use SQL to extract and analyze data from the Mint Classics relational database.

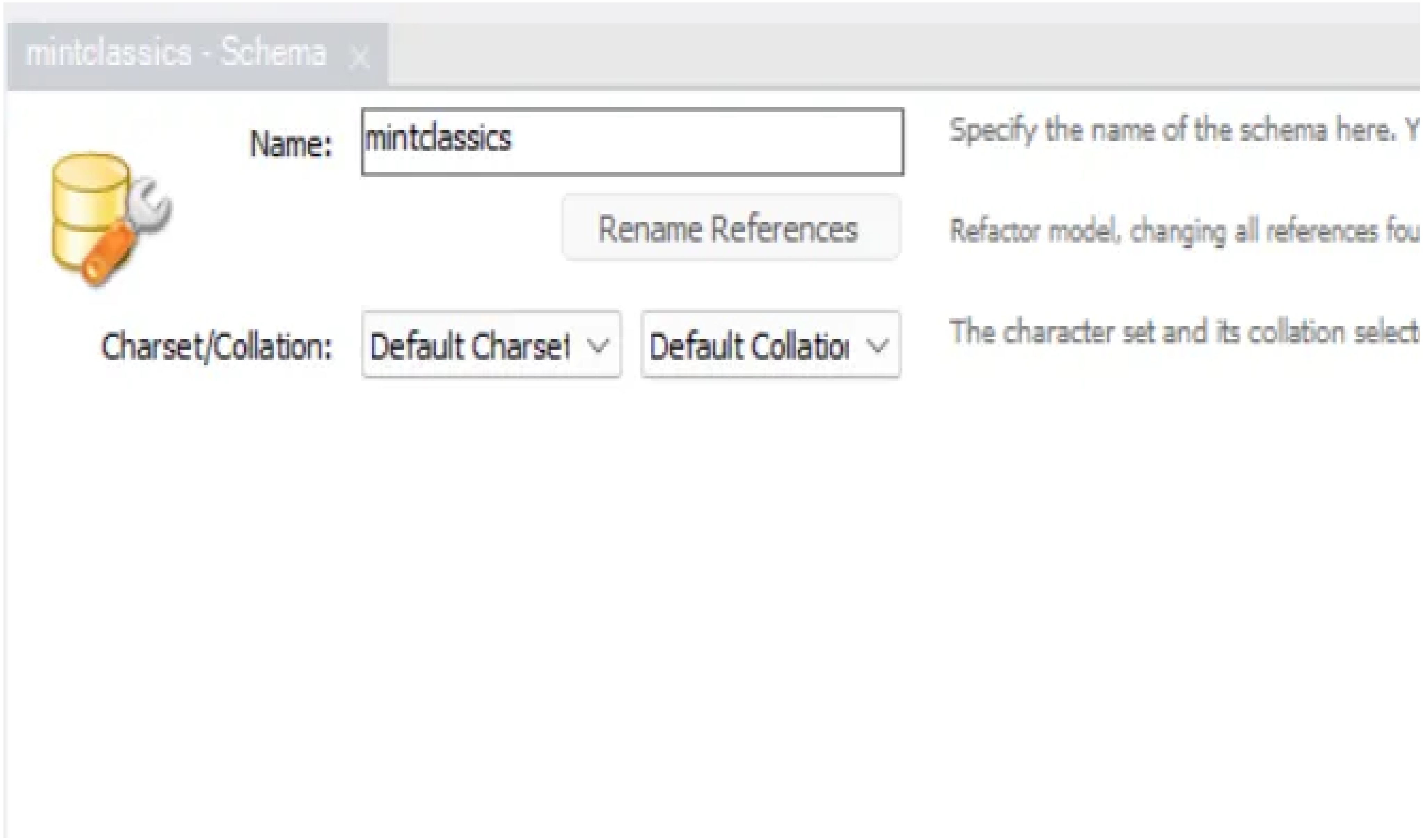
2. Business Understanding: I will grasp the database structure and Mint Classics' business processes to provide relevant solutions.

3. Decision-Making: I will formulate recommendations based on findings from data analysis.

Tools Used:

1. MySQL Workbench: This tool is utilized for importing the database, running SQL queries, and analyzing data.

2. GitHub: The project and reports will be uploaded to GitHub as part of the portfolio. [Analyze-Data-in-a-Model-Car-Database-with-MySQL-Workbench]



mintclassics - Schema

Name: mintclassics

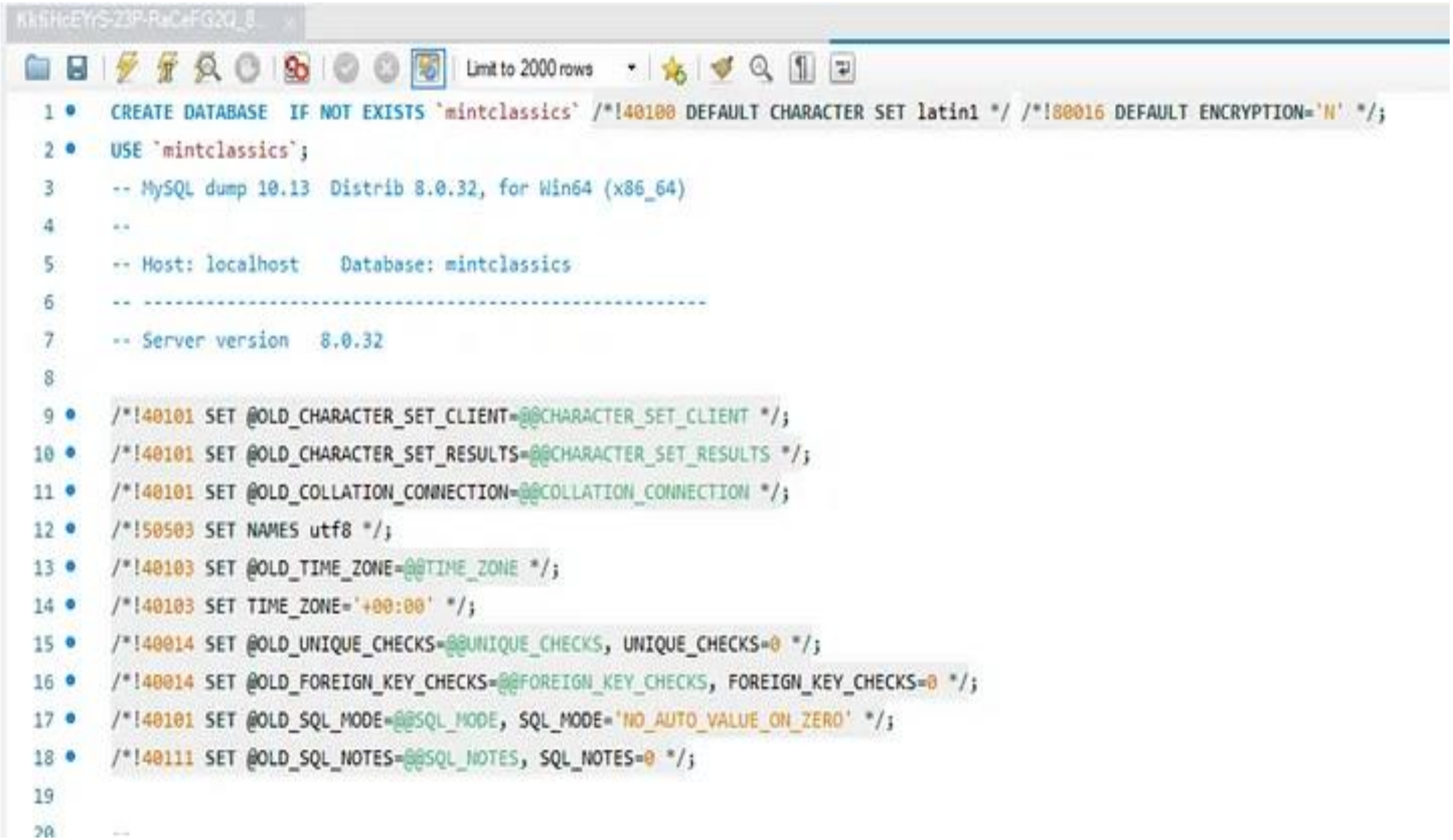
Specify the name of the schema here. Y

Rename References

Refactor model, changing all references fou

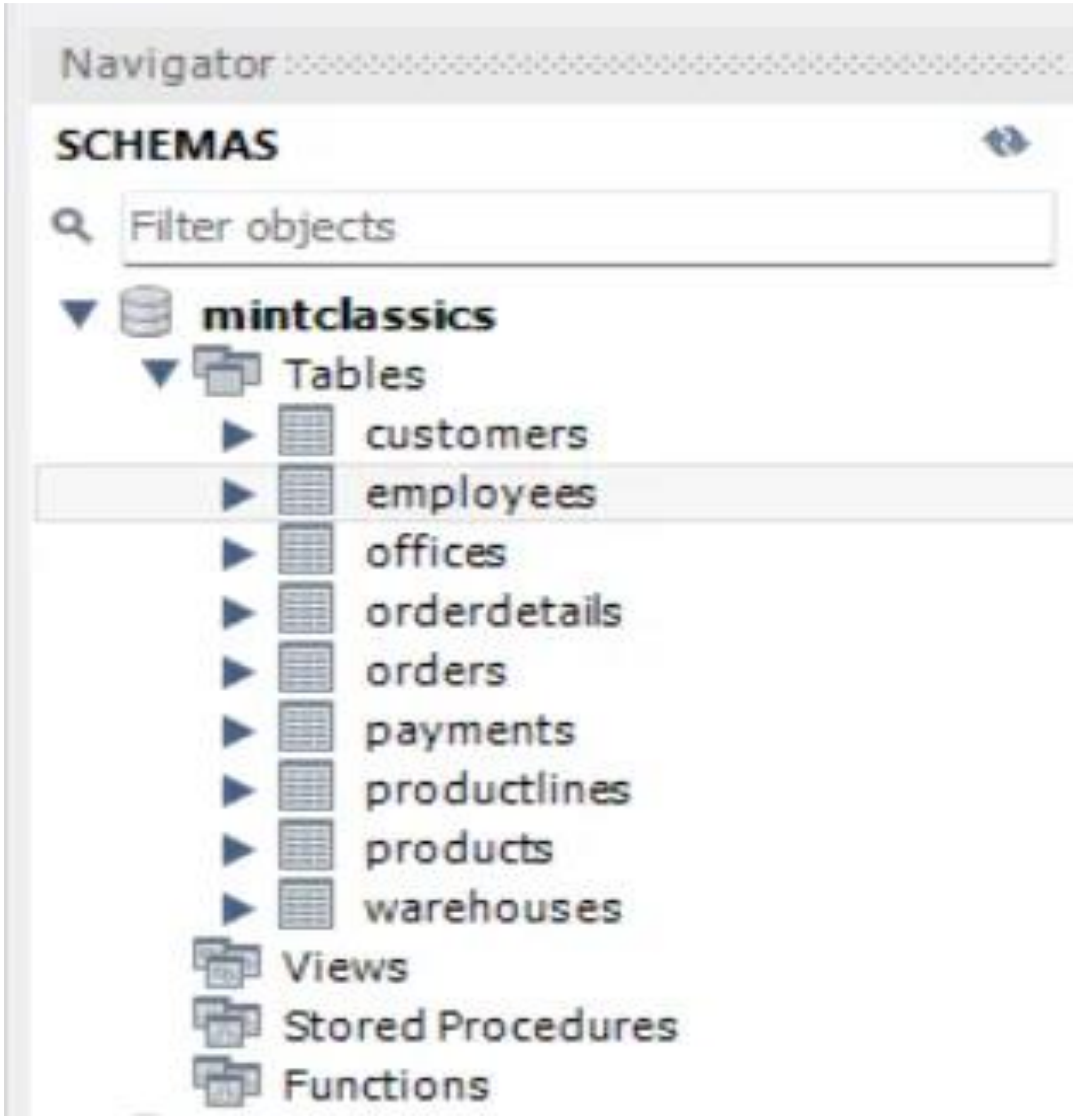
Charset/Collation: Default Charset Default Collatio

The character set and its collation select



Limit to 2000 rows

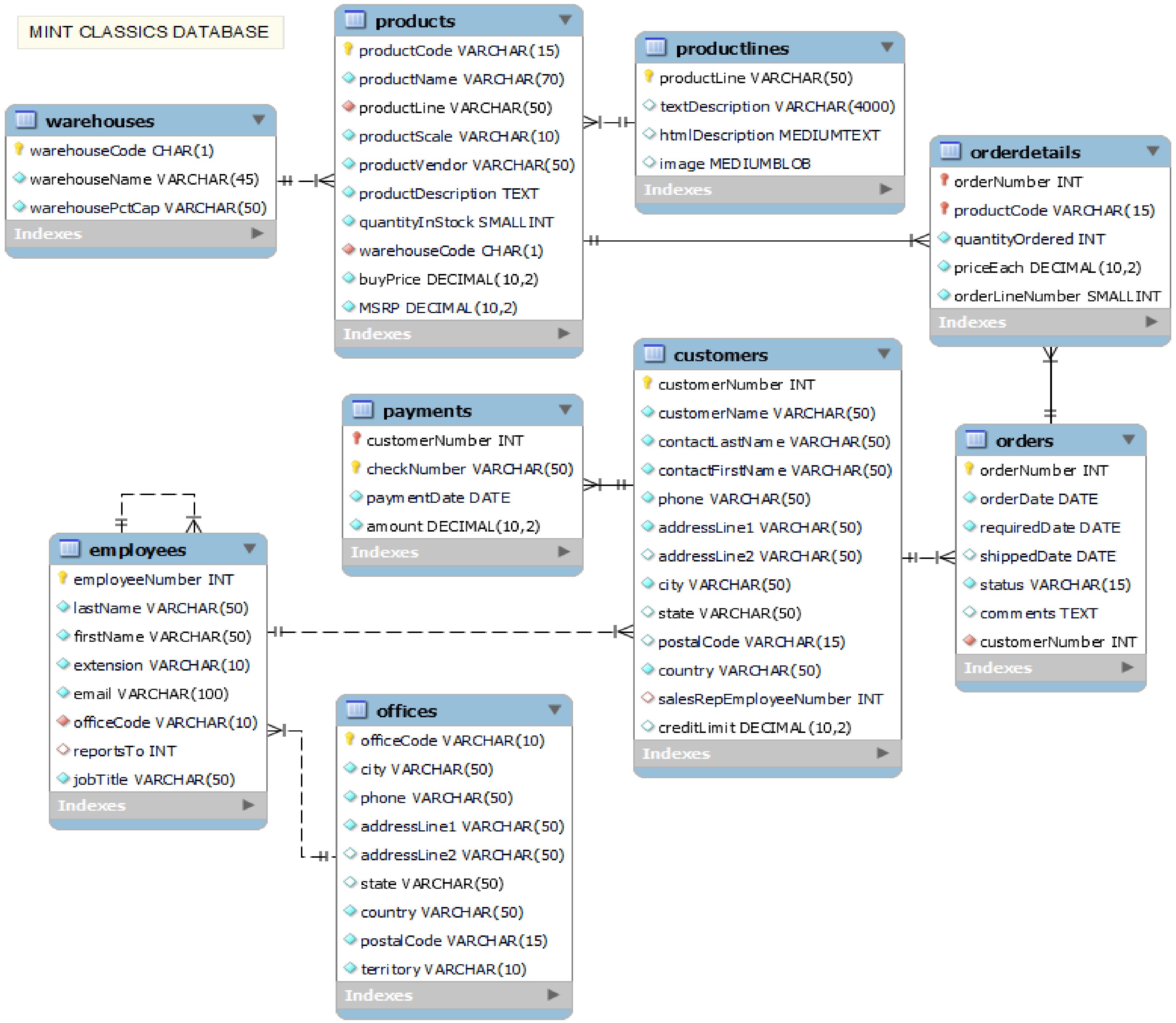
```
CREATE DATABASE  IF NOT EXISTS `mintclassics` /*!40100 DEFAULT CHARACTER SET latin1 */ /*!80016 DEFAULT ENCRYPTION='N' */;
USE `mintclassics`;
-- MySQL dump 10.13  Distrib 8.0.32, for Win64 (x86_64)
--
-- Host: localhost    Database: mintclassics
-- ------------------------------------------------------
-- Server version	8.0.32

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
```



```
mintclassics
  Tables
    customers
    employees
    offices
    orderdetails
    orders
    payments
    productlines
    products
    warehouses
  Views
  Stored Procedures
  Functions
```

Task 2 — Understanding the Mint Classics Database and Its Business Processes

I will delve into comprehending the Mint Classics database structure and how data is organized within each table. An Entity-Relationship Diagram (EER) will be employed to grasp the interrelationships among tables, and I will explore the table contents for a more profound understanding of the business data.

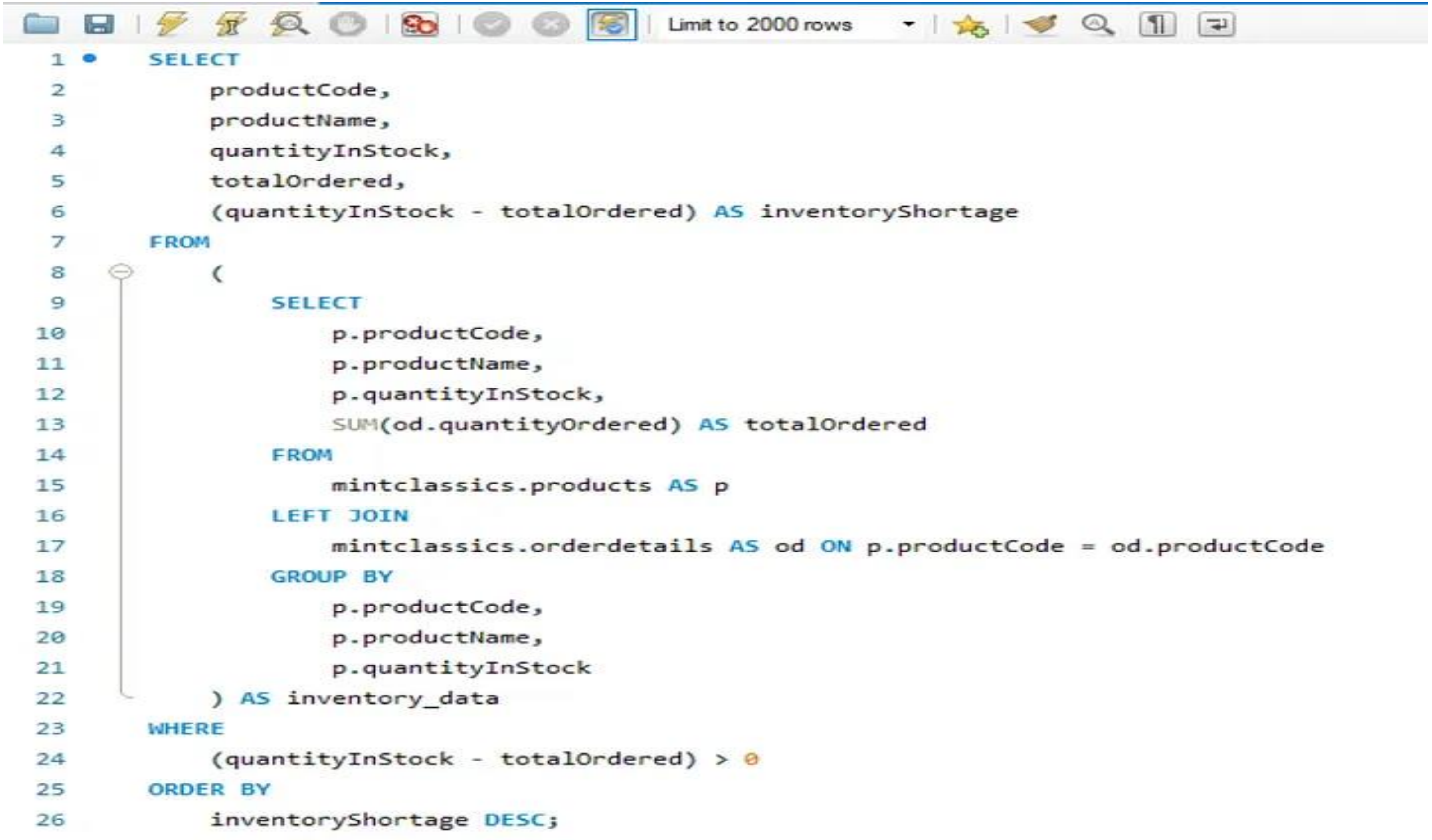

MINT CLASSICS DATABASE
products
productCode VARCHAR(15)
productName VARCHAR(70)
productLine VARCHAR(50)
productScale VARCHAR(10)
productVendor VARCHAR(50)
productDescription TEXT
quantityInStock SMALLINT
warehouseCode CHAR(1)
buyPrice DECIMAL(10,2)
MSRP DECIMAL(10,2)
Indexes
productlines
productLine VARCHAR(50)
textDescription VARCHAR(4000)
htmlDescription MEDIUMTEXT
image MEDIUMBLOB
Indexes
warehouses
warehouseCode CHAR(1)
warehouseName VARCHAR(45)
warehousePctCap VARCHAR(50)
Indexes
orderdetails
orderNumber INT
productCode VARCHAR(15)
quantityOrdered INT
priceEach DECIMAL(10,2)
orderLineNumber SMALLINT
Indexes
customers
customerNumber INT
customerName VARCHAR(50)
contactLastName VARCHAR(50)
contactFirstName VARCHAR(50)
phone VARCHAR(50)
addressLine1 VARCHAR(50)
addressLine2 VARCHAR(50)
city VARCHAR(50)
state VARCHAR(50)
postalCode VARCHAR(15)
country VARCHAR(50)
salesRepEmployeeNumber INT
creditLimit DECIMAL(10,2)
Indexes
payments
customerNumber INT
checkNumber VARCHAR(50)
paymentDate DATE
amount DECIMAL(10,2)
Indexes
orders
orderNumber INT
orderDate DATE
requiredDate DATE
shippedDate DATE
status VARCHAR(15)
comments TEXT
customerNumber INT
Indexes
employees
employeeNumber INT
lastName VARCHAR(50)
firstName VARCHAR(50)
extension VARCHAR(10)
email VARCHAR(100)
officeCode VARCHAR(10)
reportsTo INT
jobTitle VARCHAR(50)
Indexes
offices
officeCode VARCHAR(10)
city VARCHAR(50)
phone VARCHAR(50)
addressLine1 VARCHAR(50)
addressLine2 VARCHAR(50)
state VARCHAR(50)
country VARCHAR(50)
postalCode VARCHAR(15)
territory VARCHAR(10)
Indexes

Task 3 — Investigating Business Issues and Identifying Affected Tables

I will investigate the business issue faced by Mint Classics, which is the plan to close one of its storage facilities. I will identify the relevant tables for this problem and utilize SQL queries to retrieve the necessary data.

The questions to be addressed are:

- Are there products with high inventory but low sales? How can we optimize the inventory of such products?

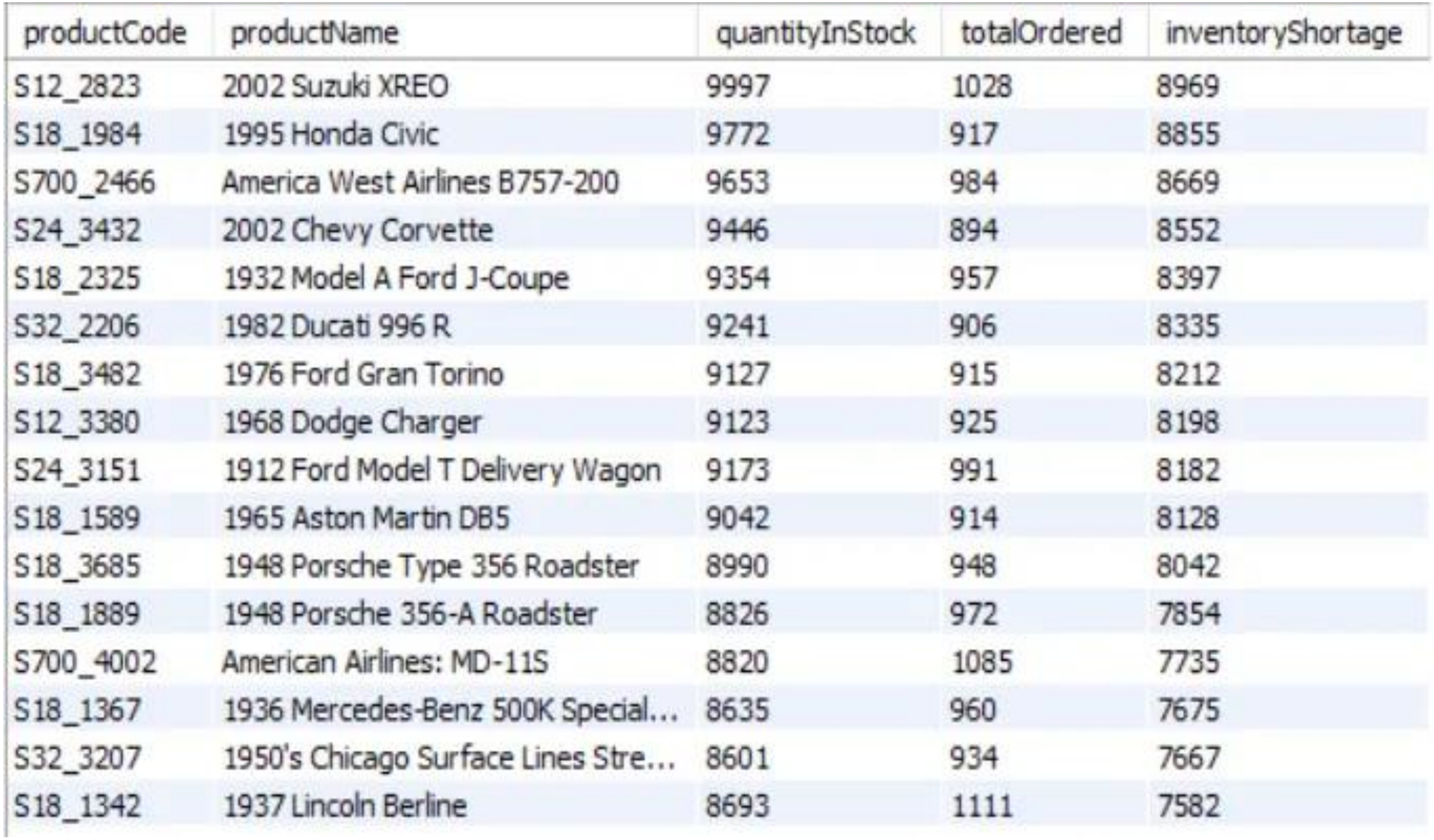


```sql
SELECT
    productCode,
    productName,
    quantityInStock,
    totalOrdered,
    (quantityInStock - totalOrdered) AS inventoryShortage
FROM
    (
        SELECT
            p.productCode,
            p.productName,
            p.quantityInStock,
            SUM(od.quantityOrdered) AS totalOrdered
        FROM
            mintclassics.products AS p
        LEFT JOIN
            mintclassics.orderdetails AS od ON p.productCode = od.productCode
        GROUP BY
            p.productCode,
            p.productName,
            p.quantityInStock
    ) AS inventory_data
WHERE
    (quantityInStock - totalOrdered) > 0
ORDER BY
    inventoryShortage DESC;
```

This query retrieves data from the “mintclassics.products” table and order details from the “mintclassics.orderdetails” table. It then groups the data by product code, product name, and the quantity of the product available in stock. Next, the query calculates the total quantity of the product ordered by combining data from both tables.

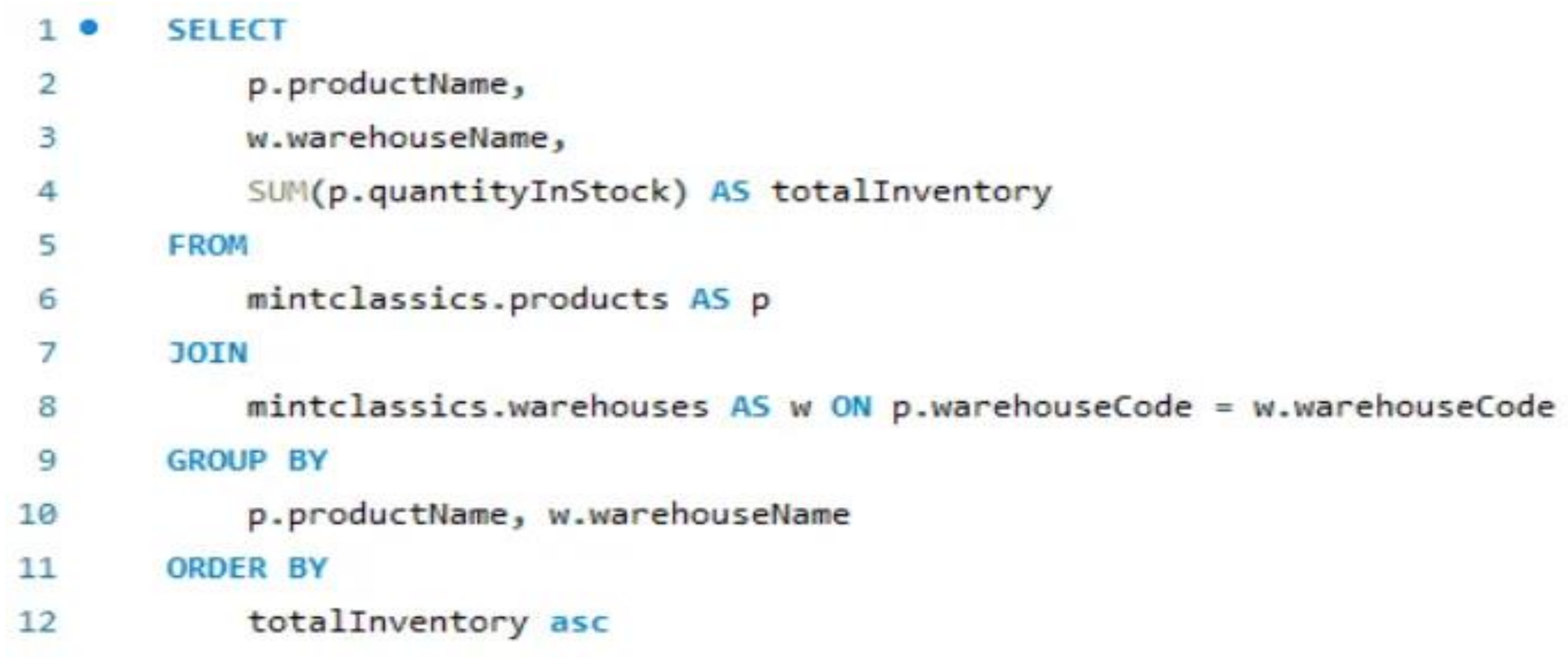
Subsequently, the query calculates the difference between the quantity of the product available in stock and the total quantity ordered, labeling this result as “inventoryShortage.” Finally, the query selects only those products that have a shortage in stock (inventory shortage) with the condition that the difference must be greater than 0, and it sorts the results from largest to smallest based on the inventory shortage.

In other words, this query is used to find products with quantities less than what was ordered (inventory shortage) and sorts them by the extent of their shortages.



productCode	productName	quantityInStock	totalOrdered	inventoryShortage
S12_2823	2002 Suzuki XREO	9997	1028	8969
S18_1984	1995 Honda Civic	9772	917	8855
S700_2466	America West Airlines B757-200	9653	984	8669
S24_3432	2002 Chevy Corvette	9446	894	8552
S18_2325	1932 Model A Ford J-Coupe	9354	957	8397
S32_2206	1982 Ducati 996 R	9241	906	8335
S18_3482	1976 Ford Gran Torino	9127	915	8212
S12_3380	1968 Dodge Charger	9123	925	8198
S24_3151	1912 Ford Model T Delivery Wagon	9173	991	8182
S18_1589	1965 Aston Martin DB5	9042	914	8128
S18_3685	1948 Porsche Type 356 Roadster	8990	948	8042
S18_1889	1948 Porsche 356-A Roadster	8826	972	7854
S700_4002	American Airlines: MD-11S	8820	1085	7735
S18_1367	1936 Mercedes-Benz 500K Special...	8635	960	7675
S32_3207	1950's Chicago Surface Lines Stre...	8601	934	7667
S18_1342	1937 Lincoln Berline	8693	1111	7582

Are all the warehouses currently in use still necessary? How can we review warehouses that have low or inactive inventory?



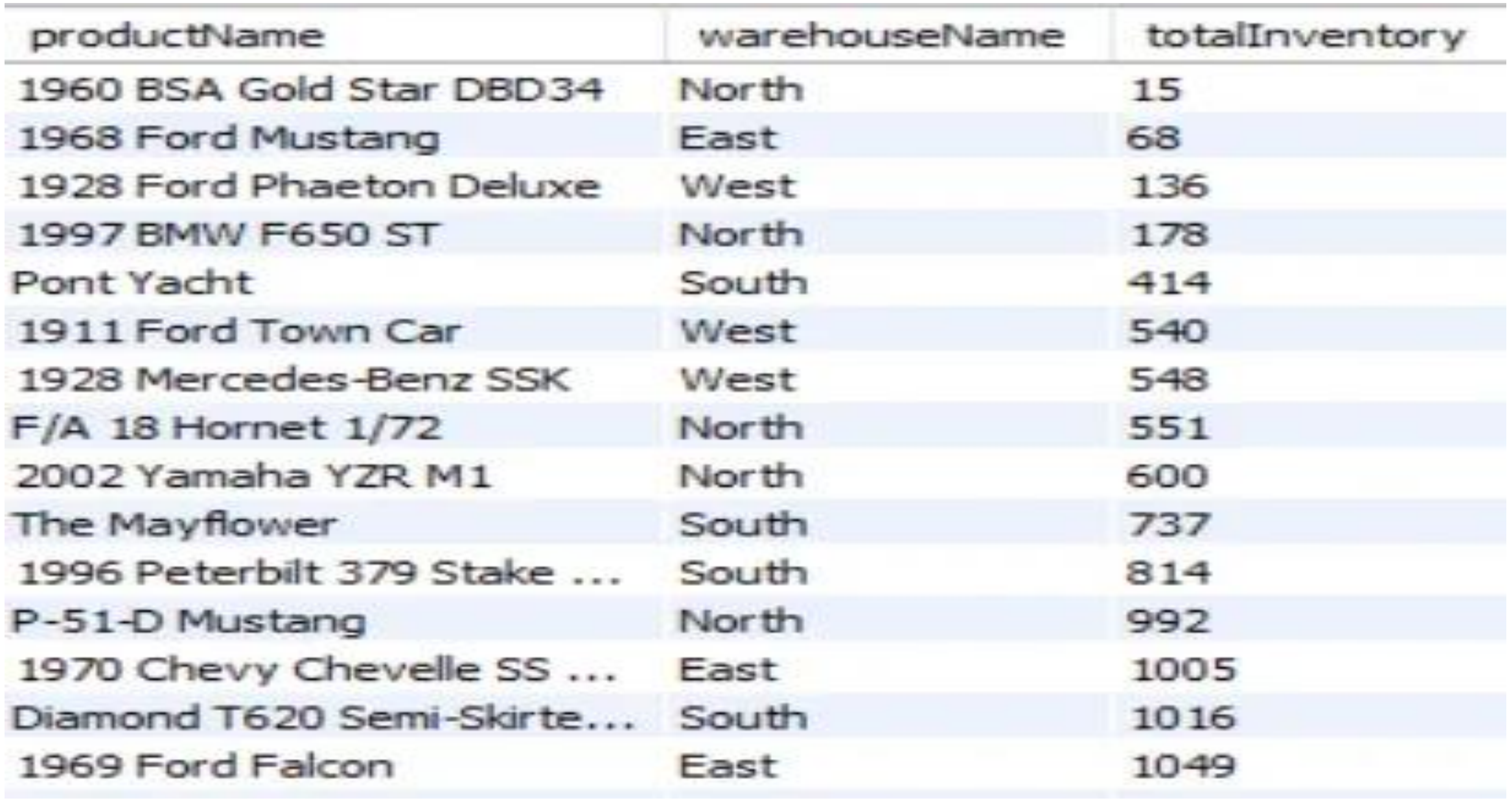
```sql
SELECT
    p.productName,
    w.warehouseName,
    SUM(p.quantityInStock) AS totalInventory
FROM
    mintclassics.products AS p
JOIN
    mintclassics.warehouses AS w ON p.warehouseCode = w.warehouseCode
GROUP BY
    p.productName, w.warehouseName
ORDER BY
    totalInventory asc
```

This query is used to retrieve the total inventory (totalInventory) of each product in each warehouse. The query performs a join between the “mintclassics.products” (products) table and the “mintclassics.warehouses” (warehouses) table based on the warehouse code (warehouseCode).

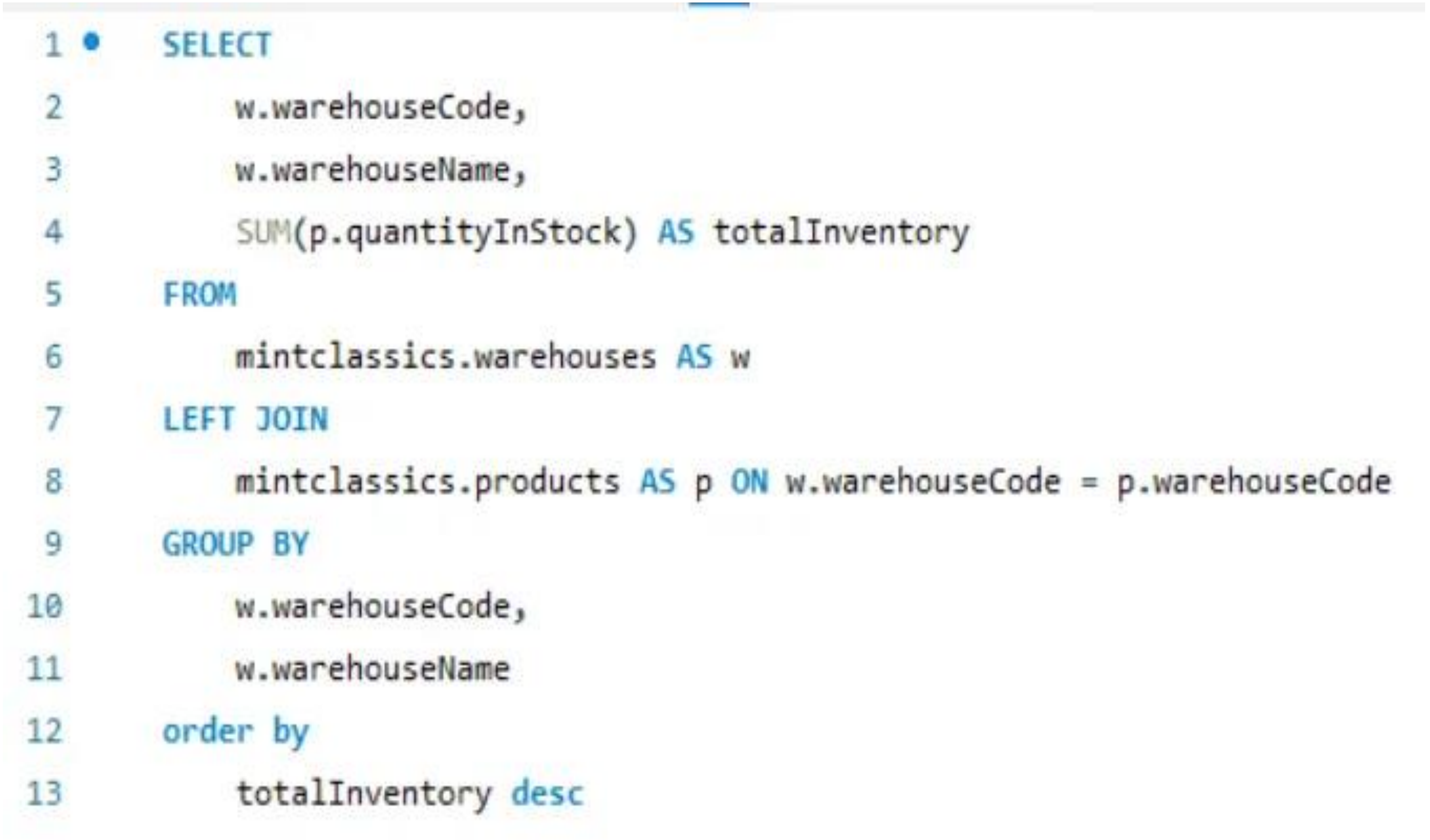
After the join, the data is grouped (GROUP BY) by product name (productName) and warehouse name (warehouseName). Then, the query calculates the total quantity of products available in stock (quantityInStock) for each combination of product and warehouse. The results of this calculation are stored in the “totalInventory” column.

The query’s results are then sorted (ORDER BY) in ascending order (from smallest to largest) based on totalInventory, resulting in a list of product-warehouse combinations with the lowest total inventory at the top.

In other words, this query helps you see the total inventory of each product in each warehouse and sorts it from the smallest to the largest.



productName	warehouseName	totalInventory
1960 BSA Gold Star DBD34	North	15
1968 Ford Mustang	East	68
1928 Ford Phaeton Deluxe	West	136
1997 BMW F650 ST	North	178
Pont Yacht	South	414
1911 Ford Town Car	West	540
1928 Mercedes-Benz SSK	West	548
F/A 18 Hornet 1/72	North	551
2002 Yamaha YZR M1	North	600
The Mayflower	South	737
1996 Peterbilt 379 Stake ...	South	814
P-51-D Mustang	North	992
1970 Chevy Chevelle SS ...	East	1005
Diamond T620 Semi-Skirte...	South	1016
1969 Ford Falcon	East	1049



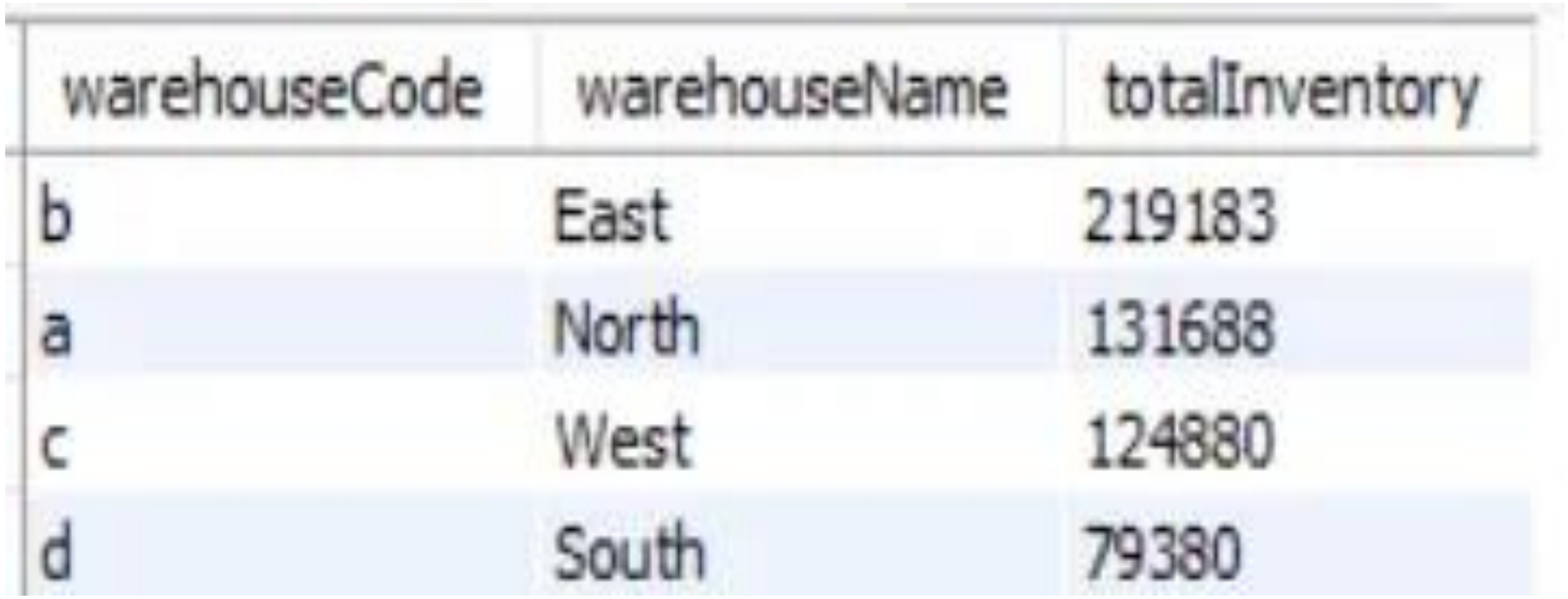
```sql
SELECT
    w.warehouseCode,
    w.warehouseName,
    SUM(p.quantityInStock) AS totalInventory
FROM
    mintclassics.warehouses AS w
LEFT JOIN
    mintclassics.products AS p ON w.warehouseCode = p.warehouseCode
GROUP BY
    w.warehouseCode,
    w.warehouseName
order by
    totalInventory desc
```

This query is intended to retrieve the total inventory data (totalInventory) for each warehouse in the "mintclassics.warehouses" table. The query performs a left join between the "mintclassics.warehouses" (warehouses) table and the "mintclassics.products" (products) table based on the warehouse code (warehouseCode).

After the join, the data is grouped (GROUP BY) by warehouse code (warehouseCode) and warehouse name (warehouseName). Then, the query calculates the total quantity of products available in stock (quantityInStock) for each warehouse. The results of this calculation are stored in the "totalInventory" column.

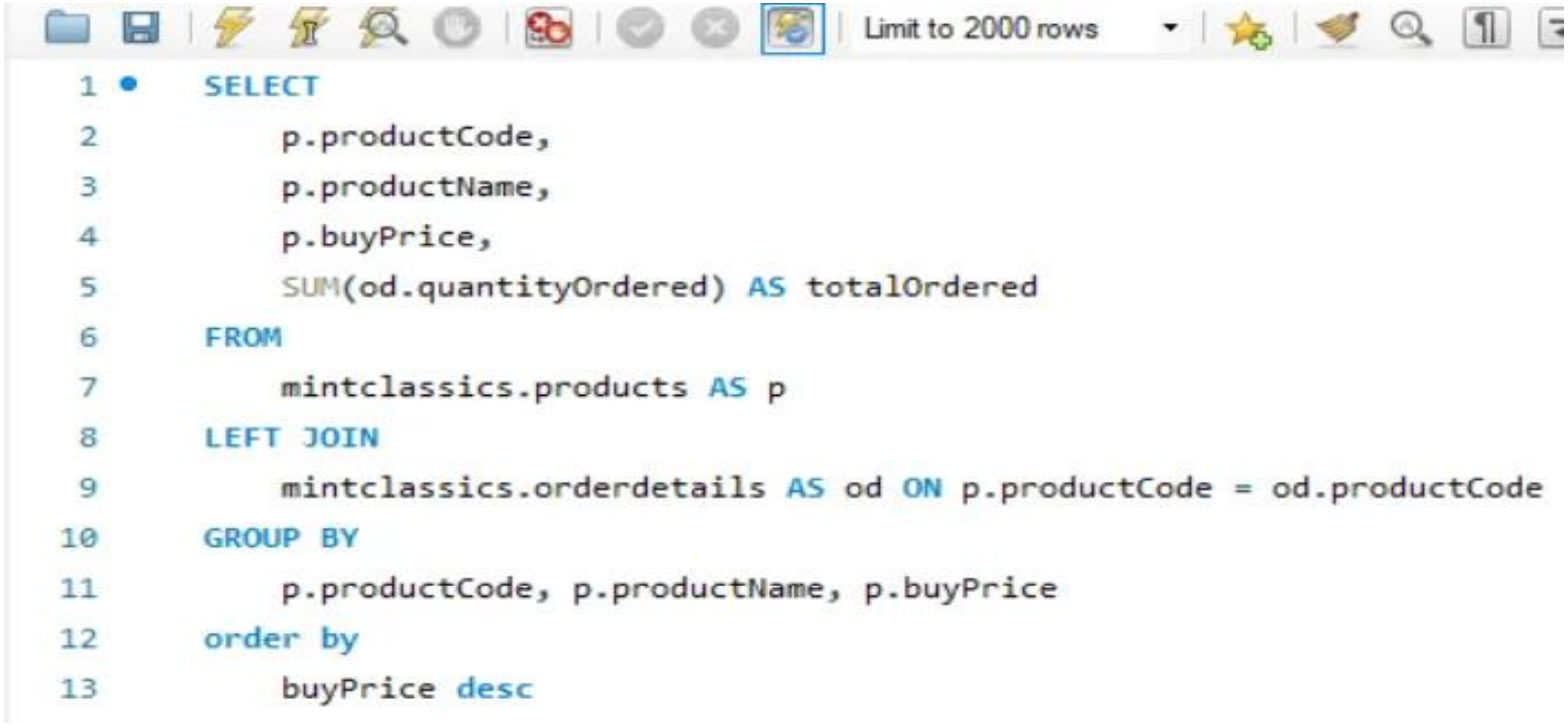
The query results are then sorted (ORDER BY) in descending order (from largest to smallest) based on totalInventory, resulting in a list of warehouses with the highest total inventory at the top.

In other words, this query helps you view the total inventory for each warehouse and sorts it from highest to lowest.



warehouseCode	warehouseName	totalInventory
b	East	219183
a	North	131688
c	West	124880
d	South	79380

3. Is there a relationship between product prices and their sales levels? How can price adjustments impact sales?



```sql
SELECT
    p.productCode,
    p.productName,
    p.buyPrice,
    SUM(od.quantityOrdered) AS totalOrdered
FROM
    mintclassics.products AS p
LEFT JOIN
    mintclassics.orderdetails AS od ON p.productCode = od.productCode
GROUP BY
    p.productCode, p.productName, p.buyPrice
order by
    buyPrice desc
```

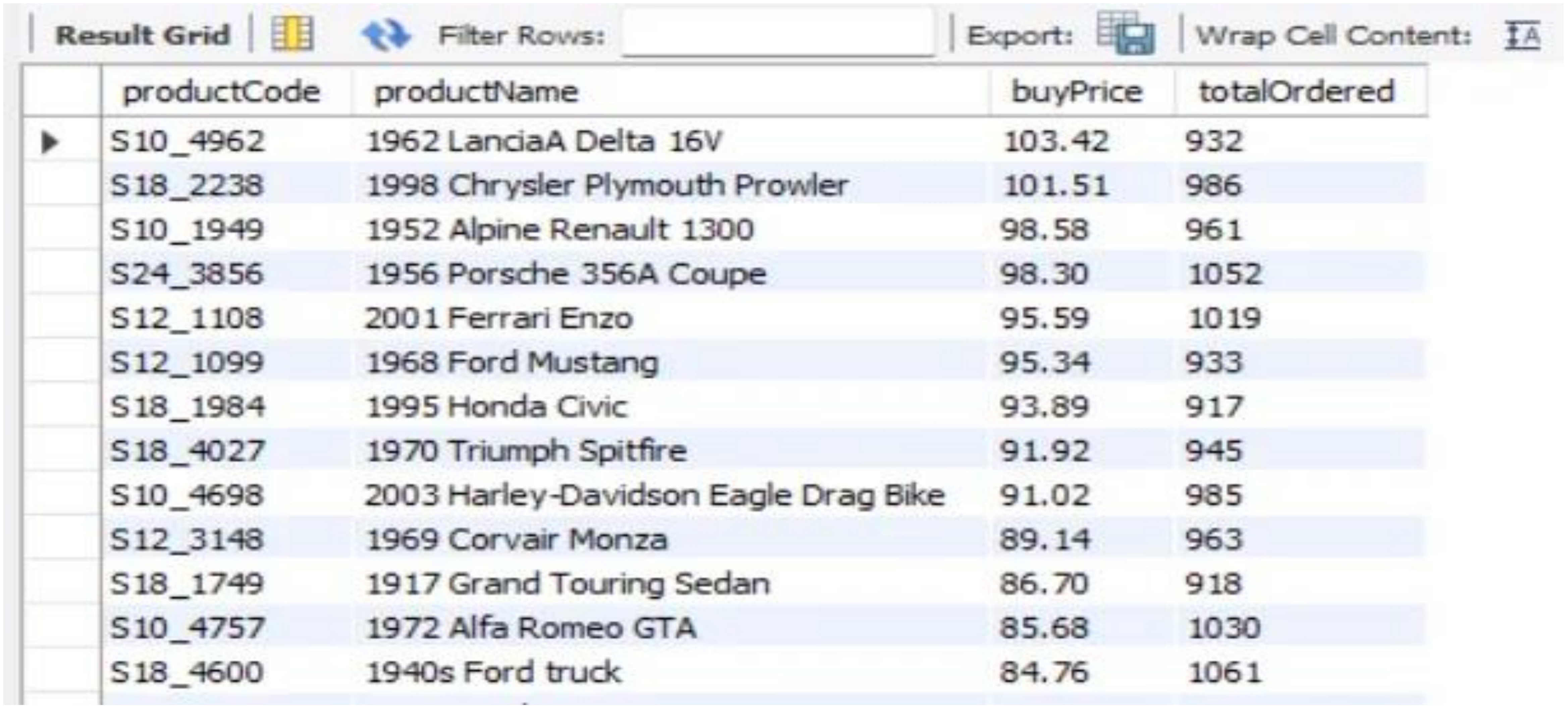
This query aims to retrieve product data such as the product code (productCode), product name (productName), purchase price (buyPrice), and the total quantity ordered (totalOrdered). It operates on the “mintclassics.products” table (products) and the “mintclassics.orderdetails” table (order details).

The process begins with a LEFT JOIN between the “mintclassics.products” (products) table and the “mintclassics.orderdetails” (order details) table based on the product code (productCode). This allows us to combine product information with the associated order quantity information.

Next, the data is grouped (GROUP BY) by the product code (productCode), product name (productName), and purchase price (buyPrice). Using GROUP BY enables us to calculate the total quantity of products ordered (totalOrdered) for each product, along with its purchase price and other details.

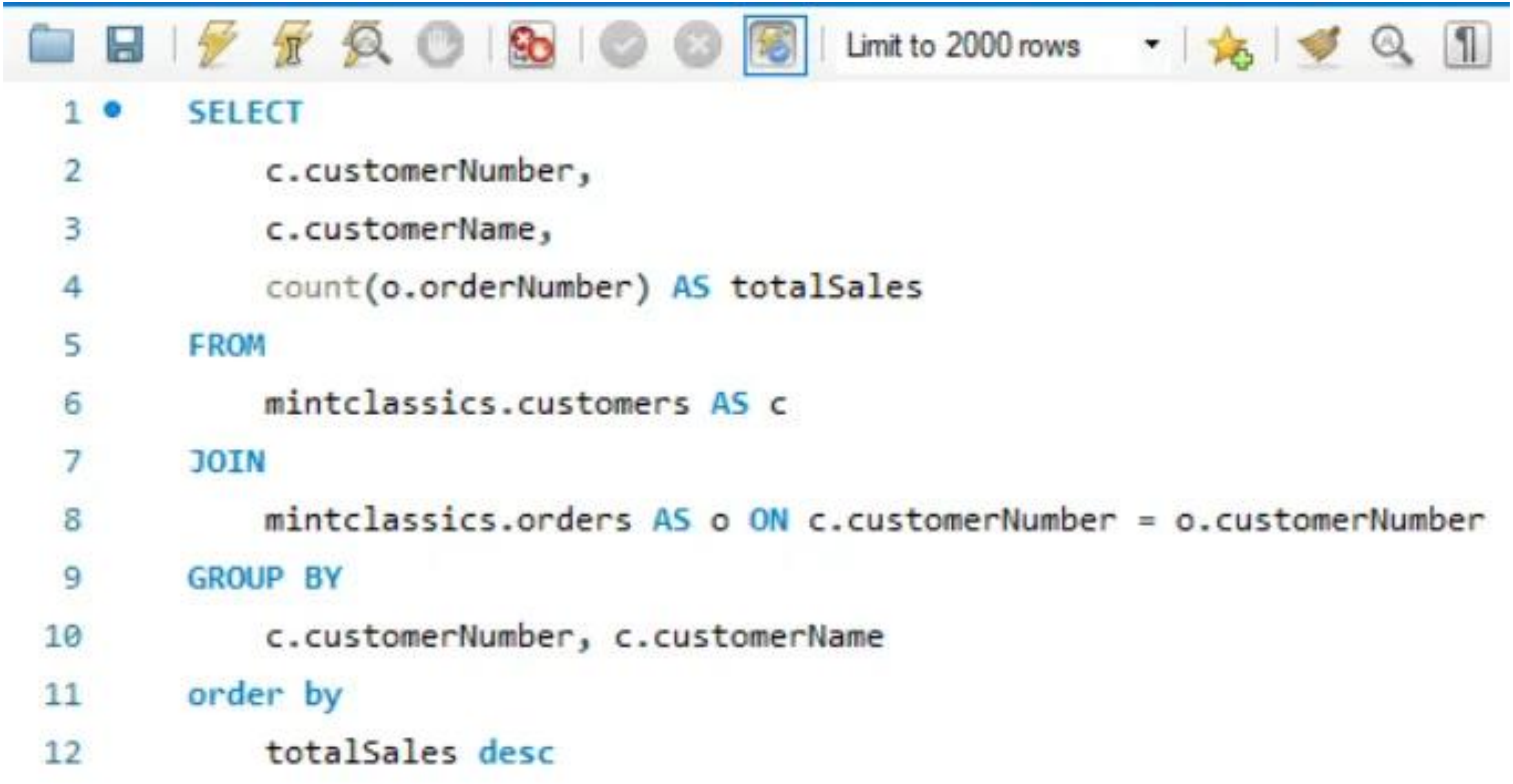
The query results are sorted (ORDER BY) in descending order (from highest to lowest) based on the purchase price (buyPrice), resulting in a list of products sorted by the highest purchase price at the top.

In other words, this query helps you obtain a list of products with the highest purchase prices, accompanied by the total quantity of products ordered for each of these products.



productCode	productName	buyPrice	totalOrdered
S10_4962	1962 LanciaA Delta 16V	103.42	932
S18_2238	1998 Chrysler Plymouth Prowler	101.51	986
S10_1949	1952 Alpine Renault 1300	98.58	961
S24_3856	1956 Porsche 356A Coupe	98.30	1052
S12_1108	2001 Ferrari Enzo	95.59	1019
S12_1099	1968 Ford Mustang	95.34	933
S18_1984	1995 Honda Civic	93.89	917
S18_4027	1970 Triumph Spitfire	91.92	945
S10_4698	2003 Harley-Davidson Eagle Drag Bike	91.02	985
S12_3148	1969 Corvair Monza	89.14	963
S18_1749	1917 Grand Touring Sedan	86.70	918
S10_4757	1972 Alfa Romeo GTA	85.68	1030
S18_4600	1940s Ford truck	84.76	1061

4. Who are the customers contributing the most to sales? How can sales efforts be focused on these valuable customers?


Limit to 2000 rows

```sql
SELECT
    c.customerNumber,
    c.customerName,
    count(o.orderNumber) AS totalSales
FROM
    mintclassics.customers AS c
JOIN
    mintclassics.orders AS o ON c.customerNumber = o.customerNumber
GROUP BY
    c.customerNumber, c.customerName
order by
    totalSales desc
```

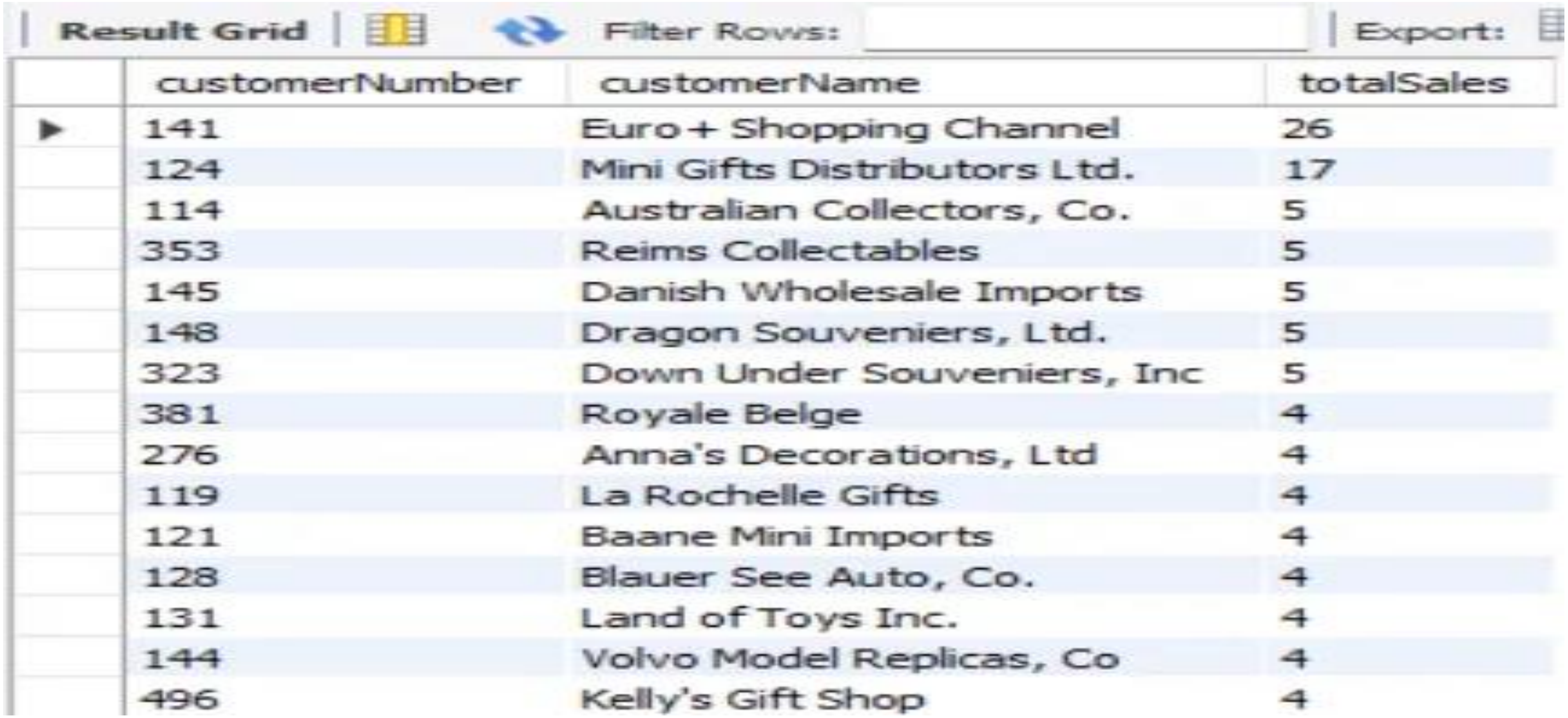
This query aims to retrieve customer data, including customer number (customerNumber), customer name (customerName), and total sales (totalSales) made by each customer. It operates on the “mintclassics.customers” (customers) table and the “mintclassics.orders” (orders) table.

The process starts with a JOIN between the “mintclassics.customers” (customers) and “mintclassics.orders” (orders) tables based on the customer number (customerNumber). This allows us to combine customer information with related order information.

Next, the data is grouped (GROUP BY) by customer number (customerNumber) and customer name (customerName). Using GROUP BY enables us to calculate the total sales (totalSales) made by each customer.

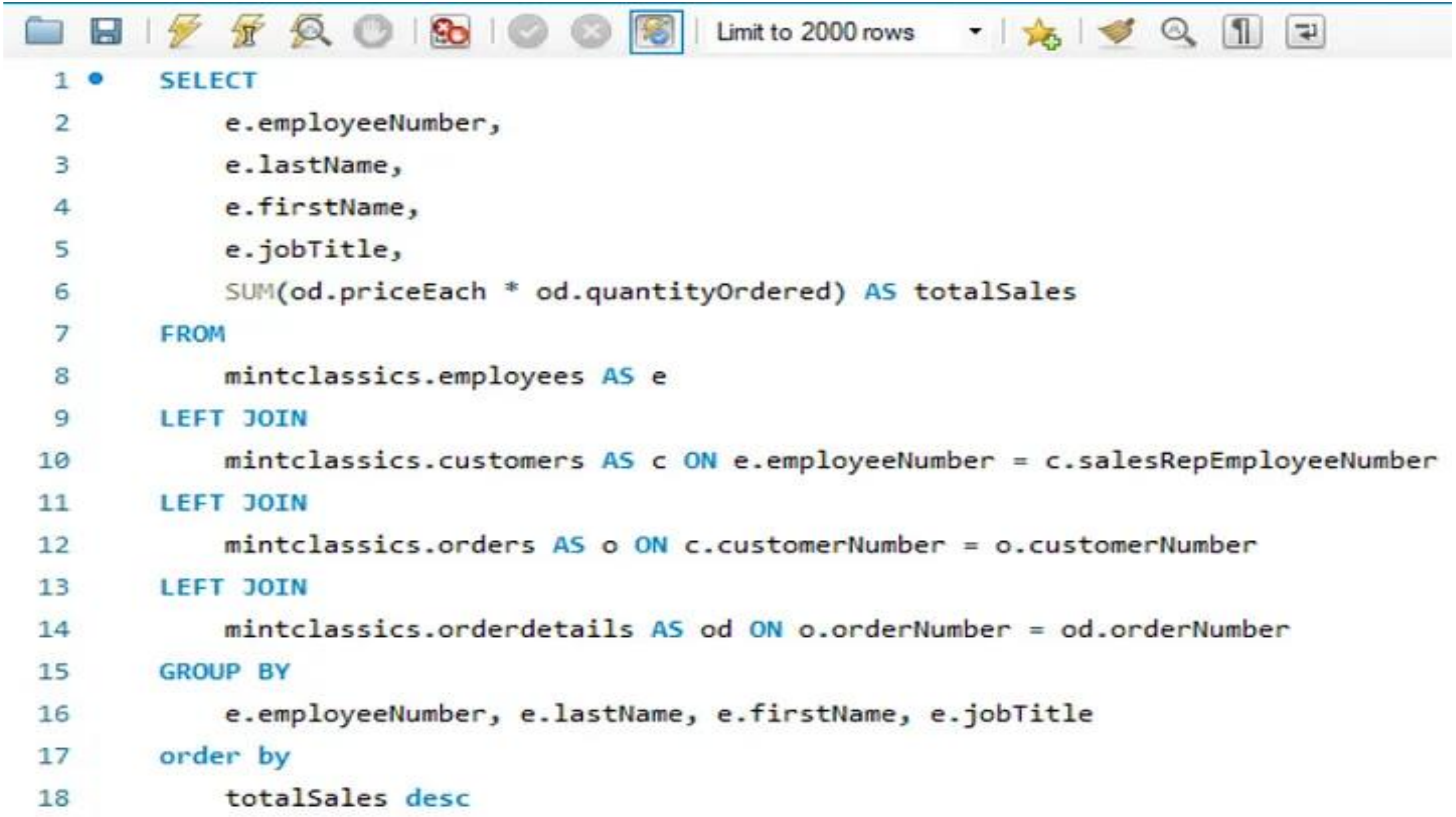
The query results are sorted (ORDER BY) in descending order (from highest to lowest) based on total sales (totalSales), resulting in a list of customers sorted by the highest total sales at the top.

In other words, this query helps you obtain a list of customers with the highest total sales, along with the total sales amount conducted by each of these customers.


Result Grid | Filter Rows: | Export:

customerNumber	customerName	totalSales
141	Euro+ Shopping Channel	26
124	Mini Gifts Distributors Ltd.	17
114	Australian Collectors, Co.	5
353	Reims Collectables	5
145	Danish Wholesale Imports	5
148	Dragon Souveniers, Ltd.	5
323	Down Under Souveniers, Inc	5
381	Royale Belge	4
276	Anna's Decorations, Ltd	4
119	La Rochelle Gifts	4
121	Baane Mini Imports	4
128	Blauer See Auto, Co.	4
131	Land of Toys Inc.	4
144	Volvo Model Replicas, Co	4
496	Kelly's Gift Shop	4

5. How can the performance of sales employees be evaluated using sales data?



```sql
SELECT
    e.employeeNumber,
    e.lastName,
    e.firstName,
    e.jobTitle,
    SUM(od.priceEach * od.quantityOrdered) AS totalSales
FROM
    mintclassics.employees AS e
LEFT JOIN
    mintclassics.customers AS c ON e.employeeNumber = c.salesRepEmployeeNumber
LEFT JOIN
    mintclassics.orders AS o ON c.customerNumber = o.customerNumber
LEFT JOIN
    mintclassics.orderdetails AS od ON o.orderNumber = od.orderNumber
GROUP BY
    e.employeeNumber, e.lastName, e.firstName, e.jobTitle
order by
    totalSales desc
```

This query is used to retrieve employee data (employees) with information such as employee number (employeeNumber), last name (lastName), first name (firstName), job title (jobTitle), and total sales (totalSales) associated with each employee. This query involves several tables, namely “mintclassics.employees” (employee table), “mintclassics.customers” (customer table), “mintclassics.orders” (order table), and “mintclassics.orderdetails” (order details).

The process begins with a LEFT JOIN between the “mintclassics.employees” (employee) table and the “mintclassics.customers” (customer) table based on the employee number (employeeNumber). This allows us to link employee information with customers who have the employee as their sales representative (salesRepEmployeeNumber).

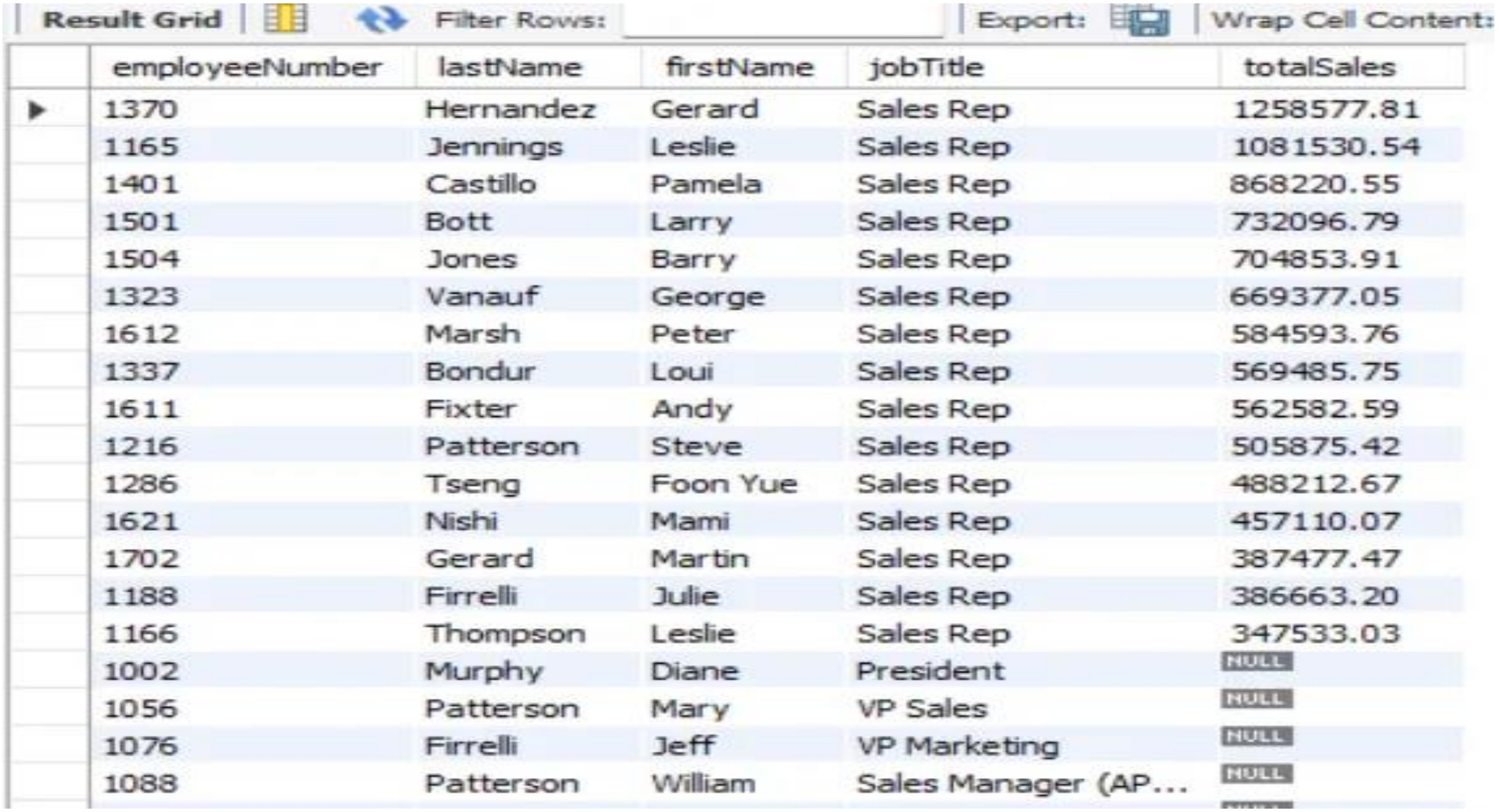
Next, there is a LEFT JOIN between the “mintclassics.customers” (customer) table and the “mintclassics.orders” (order) table based on the customer number (customerNumber). This enables us to combine customer information with the orders they have placed.

Then, there is a LEFT JOIN between the “mintclassics.orders” (order) table and the “mintclassics.orderdetails” (order details) table based on the order number (orderNumber). This allows us to connect order information with order details such as product price (priceEach) and quantity ordered (quantityOrdered).

After all the JOINs are performed, the data is grouped (GROUP BY) based on employee number (employeeNumber), last name (lastName), first name (firstName), and job title (jobTitle). The use of GROUP BY allows us to calculate the total sales (totalSales) associated with each employee.

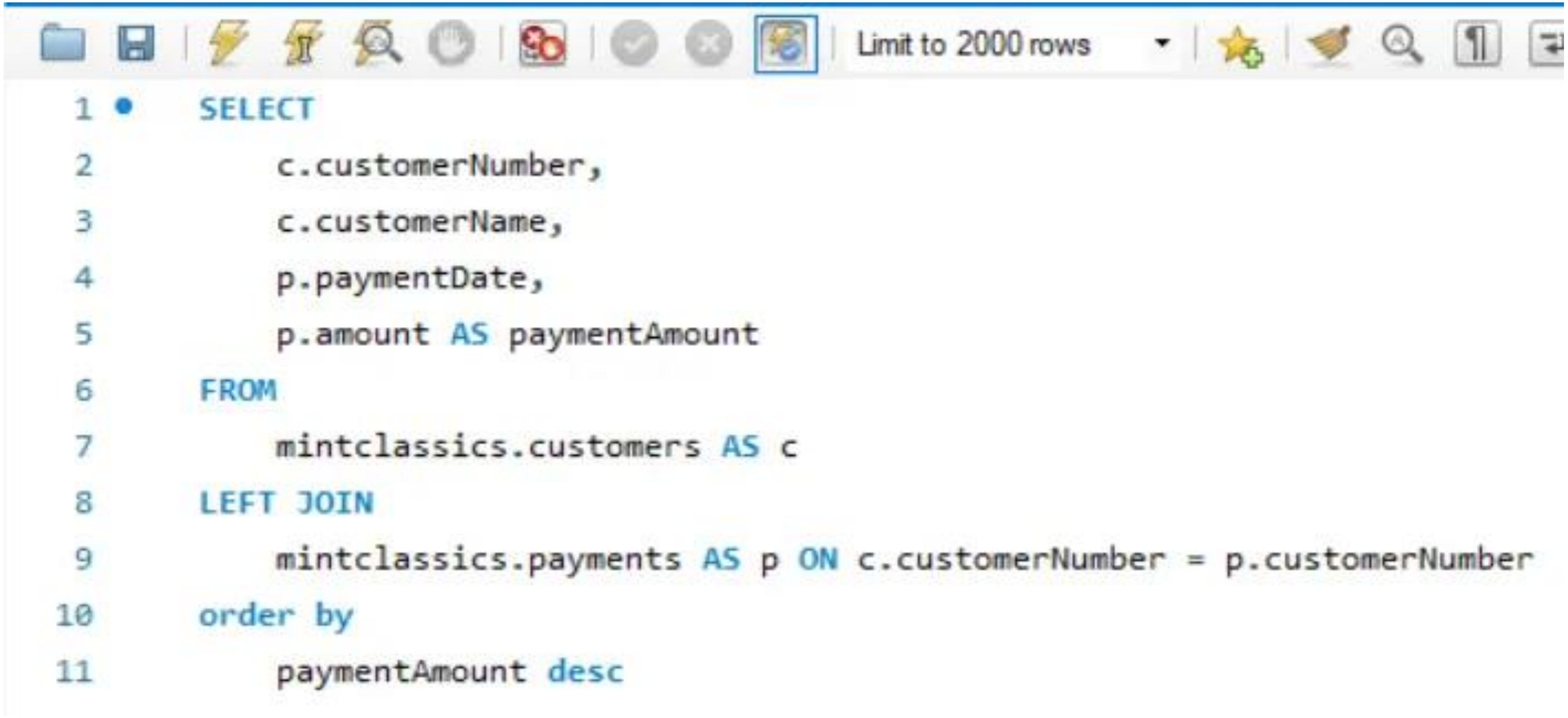
The query results are then sorted (ORDER BY) in descending order (from highest to lowest) based on total sales (totalSales), resulting in a list of employees with the highest total sales at the top.

In other words, this query helps you obtain a list of employees along with their job titles with the highest total sales, as well as the total sales amount associated with each employee.


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	employeeNumber	lastName	firstName	jobTitle	totalSales
▶	1370	Hernandez	Gerard	Sales Rep	1258577.81
	1165	Jennings	Leslie	Sales Rep	1081530.54
	1401	Castillo	Pamela	Sales Rep	868220.55
	1501	Bott	Larry	Sales Rep	732096.79
	1504	Jones	Barry	Sales Rep	704853.91
	1323	Vanauf	George	Sales Rep	669377.05
	1612	Marsh	Peter	Sales Rep	584593.76
	1337	Bondur	Loui	Sales Rep	569485.75
	1611	Fixter	Andy	Sales Rep	562582.59
	1216	Patterson	Steve	Sales Rep	505875.42
	1286	Tseng	Foon Yue	Sales Rep	488212.67
	1621	Nishi	Mami	Sales Rep	457110.07
	1702	Gerard	Martin	Sales Rep	387477.47
	1188	Firrelli	Julie	Sales Rep	386663.20
	1166	Thompson	Leslie	Sales Rep	347533.03
	1002	Murphy	Diane	President	NULL
	1056	Patterson	Mary	VP Sales	NULL
	1076	Firrelli	Jeff	VP Marketing	NULL
	1088	Patterson	William	Sales Manager (AP...	NULL

6. How can customer payment trends be analyzed? What credit risks need to be considered, and how can cash flow be managed?



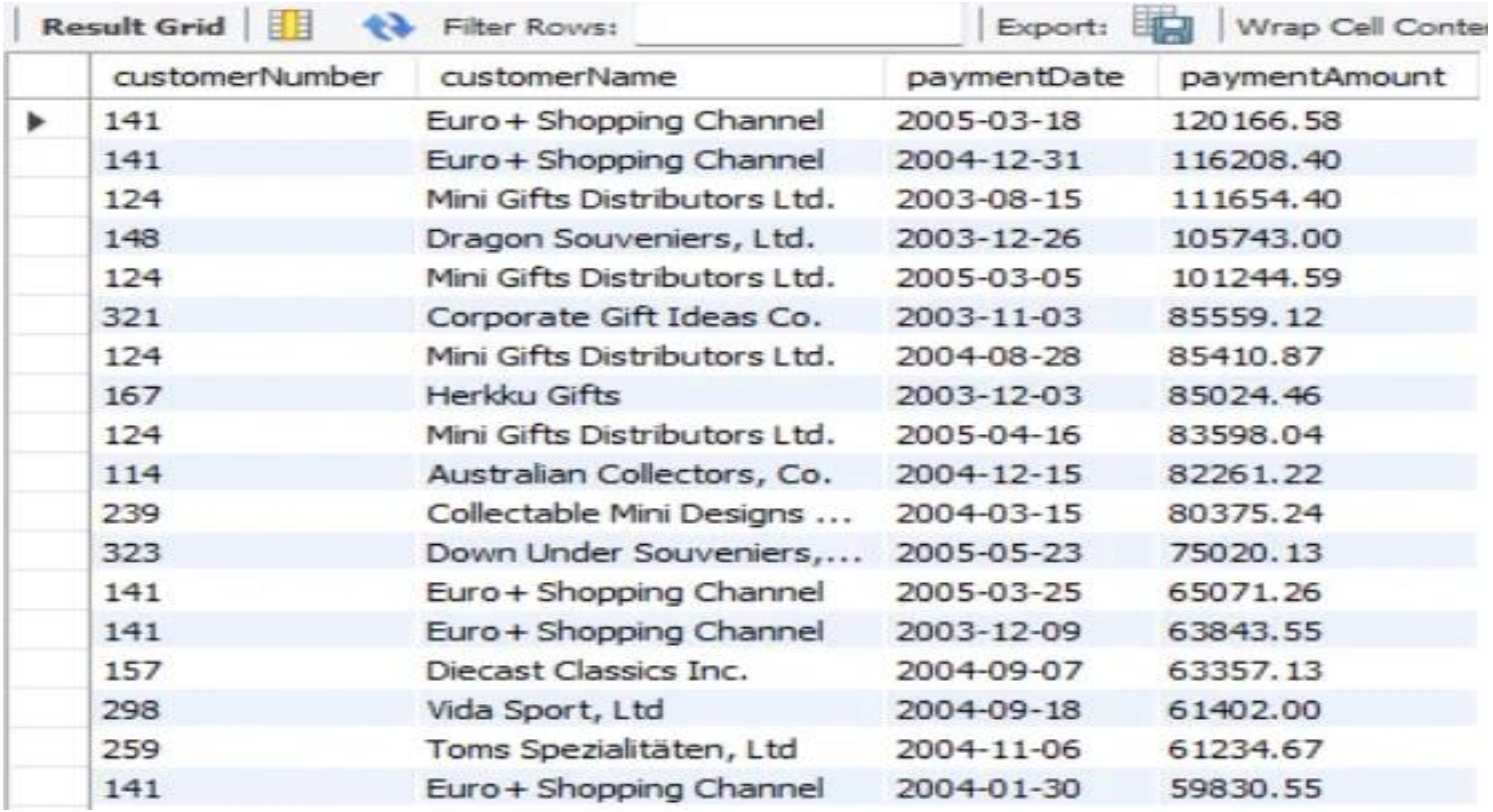
```sql
SELECT
    c.customerNumber,
    c.customerName,
    p.paymentDate,
    p.amount AS paymentAmount
FROM
    mintclassics.customers AS c
LEFT JOIN
    mintclassics.payments AS p ON c.customerNumber = p.customerNumber
order by
    paymentAmount desc
```

This query is used to retrieve customer data (customers) and payment information (payments) associated with each customer. The data retrieved includes customer number (customerNumber), customer name (customerName), payment date (paymentDate), and payment amount (paymentAmount).

The process begins by performing a LEFT JOIN between the “mintclassics.customers” table (customers) and the “mintclassics.payments” table (payments) based on the customer number (customerNumber). This allows us to combine customer information with payment data related to each customer.

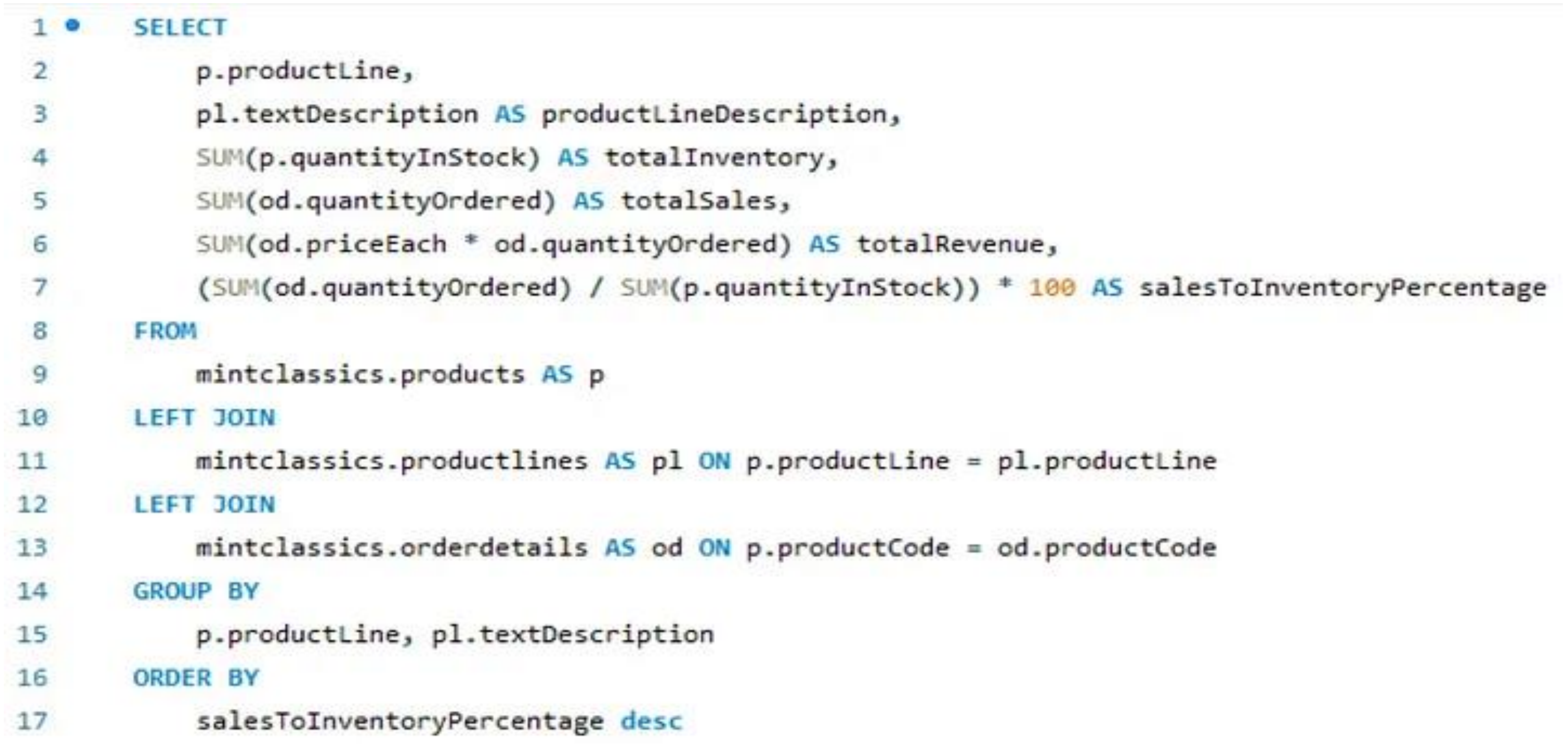
The query results are sorted (ORDER BY) in descending order (from highest to lowest) based on the payment amount (paymentAmount). In other words, the data is displayed in order from the largest payment to the smallest.

By using this query, you can view a list of customers along with the dates and the largest payment amounts they have made. This helps you analyze customer payment trends and identify customers with the highest payment amounts.



customerNumber	customerName	paymentDate	paymentAmount
141	Euro+ Shopping Channel	2005-03-18	120166.58
141	Euro+ Shopping Channel	2004-12-31	116208.40
124	Mini Gifts Distributors Ltd.	2003-08-15	111654.40
148	Dragon Souveniers, Ltd.	2003-12-26	105743.00
124	Mini Gifts Distributors Ltd.	2005-03-05	101244.59
321	Corporate Gift Ideas Co.	2003-11-03	85559.12
124	Mini Gifts Distributors Ltd.	2004-08-28	85410.87
167	Herkku Gifts	2003-12-03	85024.46
124	Mini Gifts Distributors Ltd.	2005-04-16	83598.04
114	Australian Collectors, Co.	2004-12-15	82261.22
239	Collectable Mini Designs ...	2004-03-15	80375.24
323	Down Under Souveniers,...	2005-05-23	75020.13
141	Euro+ Shopping Channel	2005-03-25	65071.26
141	Euro+ Shopping Channel	2003-12-09	63843.55
157	Diecast Classics Inc.	2004-09-07	63357.13
298	Vida Sport, Ltd	2004-09-18	61402.00
259	Toms Spezialitäten, Ltd	2004-11-06	61234.67
141	Euro+ Shopping Channel	2004-01-30	59830.55

7. How can the performance of various product lines be compared? Which products are the most successful, and which ones need improvement or removal?



```sql
SELECT
    p.productLine,
    pl.textDescription AS productLineDescription,
    SUM(p.quantityInStock) AS totalInventory,
    SUM(od.quantityOrdered) AS totalSales,
    SUM(od.priceEach * od.quantityOrdered) AS totalRevenue,
    (SUM(od.quantityOrdered) / SUM(p.quantityInStock)) * 100 AS salesToInventoryPercentage
FROM
    mintclassics.products AS p
LEFT JOIN
    mintclassics.productlines AS pl ON p.productLine = pl.productLine
LEFT JOIN
    mintclassics.orderdetails AS od ON p.productCode = od.productCode
GROUP BY
    p.productLine, pl.textDescription
ORDER BY
    salesToInventoryPercentage desc
```

This query is used to retrieve data about various product lines along with related information. The data includes the product line name (productLine), product line description (productLineDescription), total inventory (totalInventory), total sales (totalSales), total revenue (totalRevenue), and the percentage of sales to inventory (salesToInventoryPercentage).

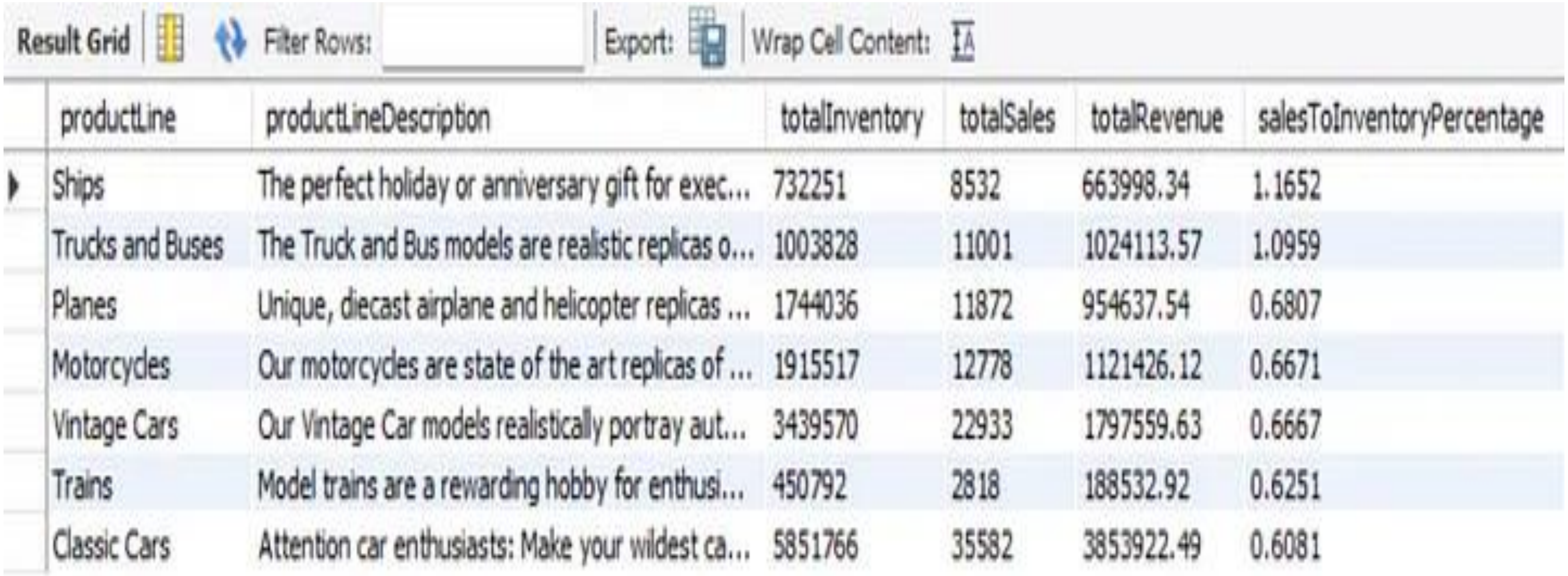
The process begins by joining the “mintclassics.products” (products) table with the “mintclassics.productlines” (product lines) table using a LEFT JOIN based on the “productLine” column. This allows us to combine product information with the corresponding product line descriptions.

Next, the resulting JOINed table is linked with the “mintclassics.orderdetails” (order details) table based on the “productCode” column. This enables us to merge product data with relevant sales data.

The query results are grouped (GROUP BY) based on the “productLine” and “productLineDescription” columns. This means that the data is grouped by product lines and their descriptions.

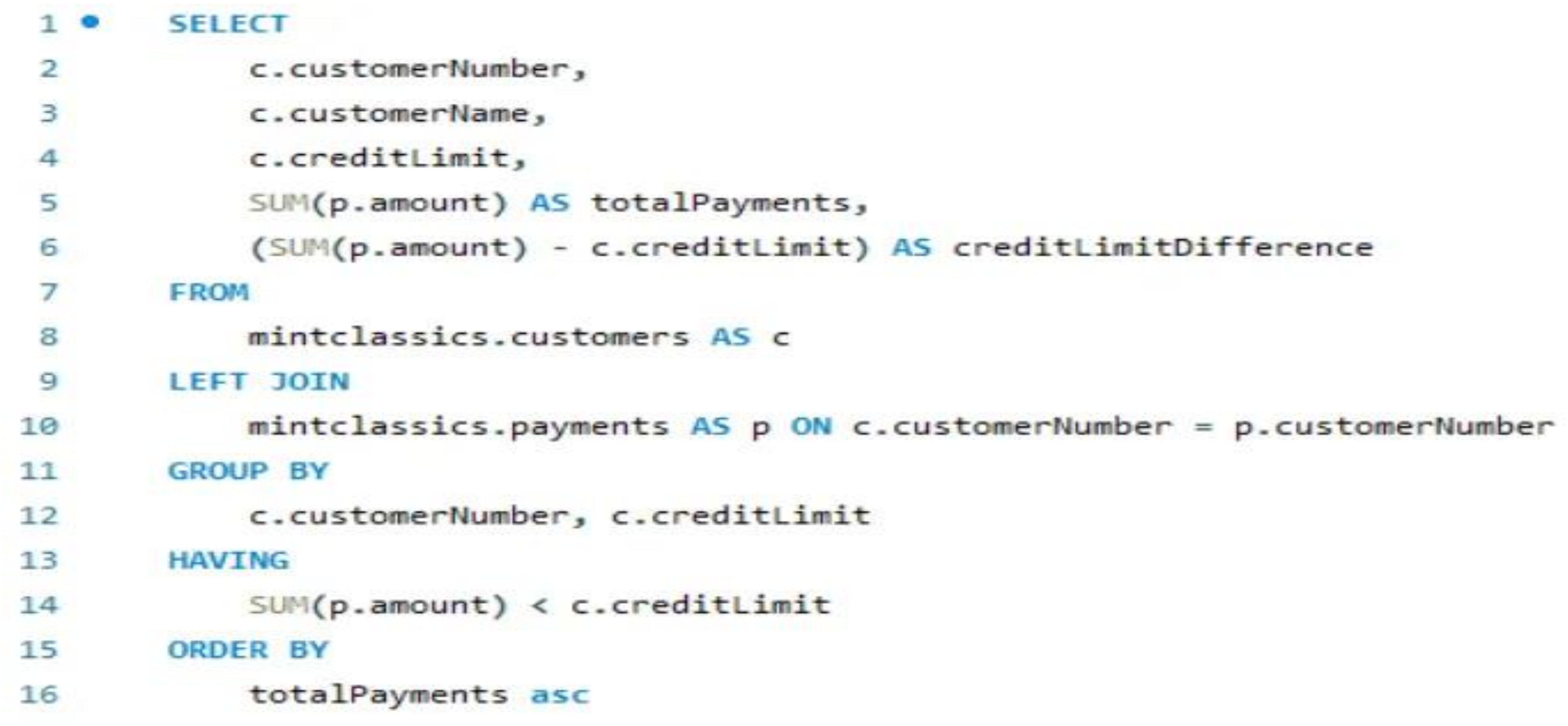
Subsequently, the data is sorted (ORDER BY) in descending order (from highest to lowest) based on the sales-to-inventory percentage (salesToInventoryPercentage). This will produce a list of product lines with the highest sales percentage first.

By using this query, you can analyze the performance of various product lines, identify which product lines have the highest sales percentage, and gain insights into how each product line performs in terms of inventory and sales.



productLine	productLineDescription	totalInventory	totalSales	totalRevenue	salesToInventoryPercentage
Ships	The perfect holiday or anniversary gift for exec...	732251	8532	663998.34	1.1652
Trucks and Buses	The Truck and Bus models are realistic replicas o...	1003828	11001	1024113.57	1.0959
Planes	Unique, diecast airplane and helicopter replicas ...	1744036	11872	954637.54	0.6807
Motorcycles	Our motorcycles are state of the art replicas of ...	1915517	12778	1121426.12	0.6671
Vintage Cars	Our Vintage Car models realistically portray aut...	3439570	22933	1797559.63	0.6667
Trains	Model trains are a rewarding hobby for enthusi...	450792	2818	188532.92	0.6251
Classic Cars	Attention car enthusiasts: Make your wildest ca...	5851766	35582	3853922.49	0.6081

8. How can the company’s credit policies be evaluated? Are there any customers with credit issues that need to be addressed?



```sql
SELECT
    c.customerNumber,
    c.customerName,
    c.creditLimit,
    SUM(p.amount) AS totalPayments,
    (SUM(p.amount) - c.creditLimit) AS creditLimitDifference
FROM
    mintclassics.customers AS c
LEFT JOIN
    mintclassics.payments AS p ON c.customerNumber = p.customerNumber
GROUP BY
    c.customerNumber, c.creditLimit
HAVING
    SUM(p.amount) < c.creditLimit
ORDER BY
    totalPayments asc
```

This query is used to retrieve customer data (customers) and related information, including customer number (customerNumber), customer name (customerName), credit limit (creditLimit), total payments made (totalPayments), and the difference between total payments and the credit limit (creditLimitDifference).

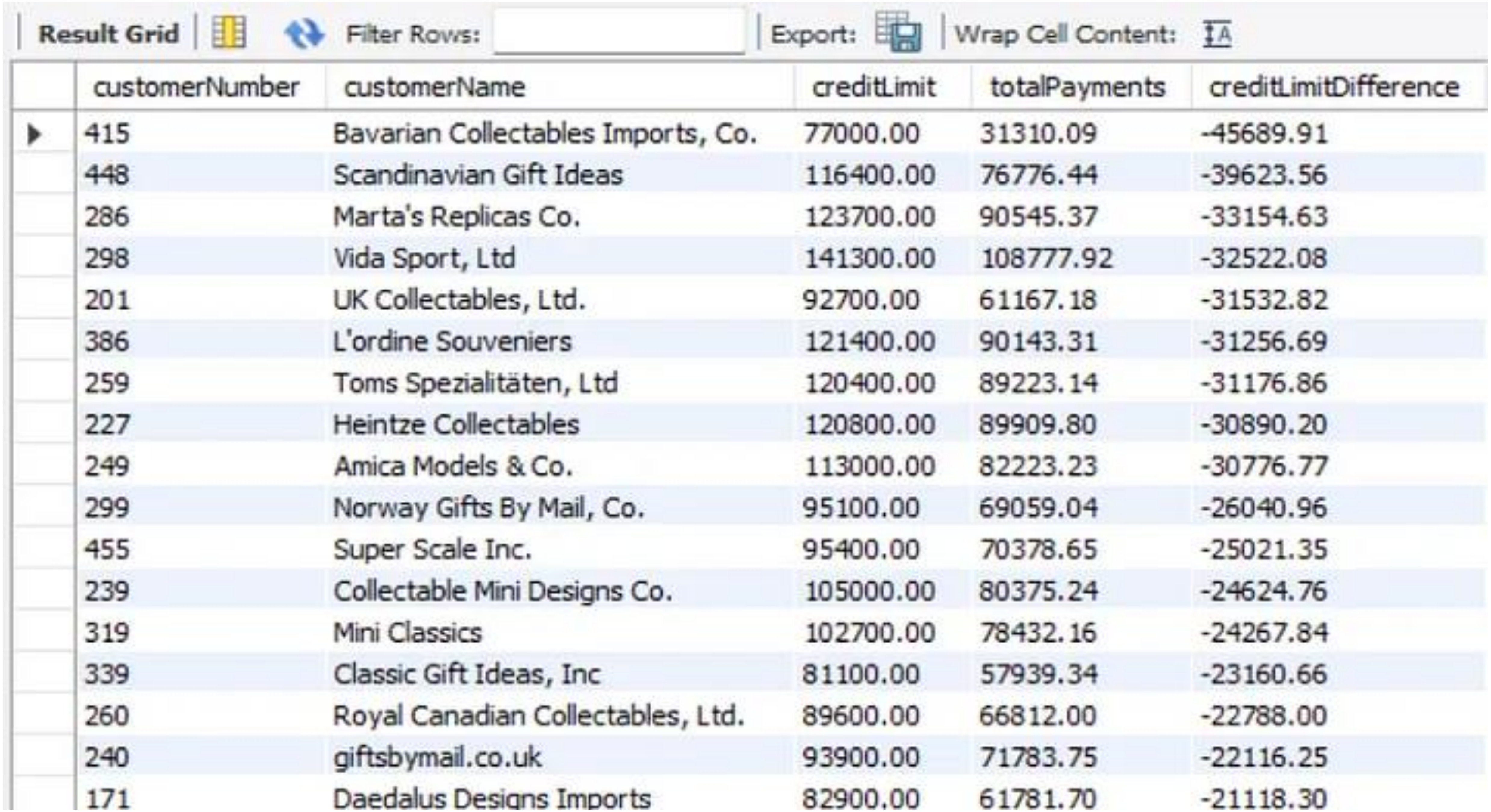
The query process begins by joining the “mintclassics.customers” table (customer table) with the “mintclassics.payments” table (payment table) using a LEFT JOIN. This joining is done based on the customer number (customerNumber), so customer data will be linked to the corresponding payment data.

Next, the data is grouped (GROUP BY) based on customer number (customerNumber) and credit limit (creditLimit). This means the data will be grouped for each customer and their credit limit value.

Then, the HAVING clause is used to filter the query results. Only customer data with total payments (totalPayments) less than the credit limit (creditLimit) will be retrieved. This means only customers who have not paid their entire credit limit will be displayed.

Finally, the query results are sorted (ORDER BY) in ascending order (from smallest to largest) based on the difference between total payments and the credit limit (creditLimitDifference). This will produce a list of customers with the smallest differences first.

By using this query, you can identify customers who have payments less than their credit limits, evaluate credit risk that needs attention, and manage the company’s cash flow.


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customerNumber	customerName	creditLimit	totalPayments	creditLimitDifference
415	Bavarian Collectables Imports, Co.	77000.00	31310.09	-45689.91
448	Scandinavian Gift Ideas	116400.00	76776.44	-39623.56
286	Marta's Replicas Co.	123700.00	90545.37	-33154.63
298	Vida Sport, Ltd	141300.00	108777.92	-32522.08
201	UK Collectables, Ltd.	92700.00	61167.18	-31532.82
386	L'ordine Souveniers	121400.00	90143.31	-31256.69
259	Toms Spezialitäten, Ltd	120400.00	89223.14	-31176.86
227	Heintze Collectables	120800.00	89909.80	-30890.20
249	Amica Models & Co.	113000.00	82223.23	-30776.77
299	Norway Gifts By Mail, Co.	95100.00	69059.04	-26040.96
455	Super Scale Inc.	95400.00	70378.65	-25021.35
239	Collectable Mini Designs Co.	105000.00	80375.24	-24624.76
319	Mini Classics	102700.00	78432.16	-24267.84
339	Classic Gift Ideas, Inc	81100.00	57939.34	-23160.66
260	Royal Canadian Collectables, Ltd.	89600.00	66812.00	-22788.00
240	giftsbymail.co.uk	93900.00	71783.75	-22116.25
171	Daedalus Designs Imports	82900.00	61781.70	-21118.30

Task 4 — Formulating Recommendations to Address Business Issues

In this task, I have conducted data analysis using SQL queries and formulated recommendations to address the inventory-related business problem. Here is the summary based on the questions posed:

1. **Inventory Optimization** : After analyzing the data, I found some products with high inventory but low sales. My recommendation is to reduce the inventory of these products. This can be achieved by either reducing the quantity ordered for these products or evaluating the actual demand for them. Reducing inventory will help in lowering storage costs and optimizing resource allocation.

2. **Warehouse Review**: Warehouse data indicates that there are warehouses with low or inactive inventory. I recommend conducting further reviews of these warehouses. Consider closing or consolidating inefficient or inactive warehouses. This will reduce warehouse rental costs and optimize inventory allocation.

3. **Product Pricing Evaluation**: The analysis of product prices reveals a relationship between price and sales performance. My recommendation is to carefully review product prices. Consider adjusting the prices of specific products with low sales. Price reductions can enhance the attractiveness of these products to customers and boost sales.

4. **Customer Analysis:** Customer data has helped identify valuable customers contributing significantly to sales. My recommendation is to focus sales efforts on these valuable customers. Provide special incentives to these customers and consider offering products that align with their preferences.

6. **Payment Analysis**: Payment trends of customers have been analyzed, and I recommend monitoring payments regularly. Identify customers with poor payment trends and take follow-up actions to mitigate credit risks. Additionally, manage cash flow carefully to ensure optimal liquidity.

7. **Product Line Review**: The analysis has shown the performance of various product lines. Products with less success need further evaluation. Consider product improvements or, if necessary, discontinuation of inefficient products. This will help in enhancing the profitability of the product portfolio.

8. **Credit Policy Evaluation**: Lastly, I recommend conducting a thorough evaluation of the company’s credit policies. Identify customers with credit issues and consider providing solutions or making changes in credit policies to reduce credit risk.

By following these recommendations and involving data analysis, the company can optimize its operations, improve profitability, and provide better customer service.

Task 5 — Crafting Conclusions and Recommendations with SQL Support

The final task is to compile conclusions about the analysis process, key findings, and the recommendations I have designed. I will explain the steps taken and why I made certain decisions. I will also include SQL queries that support my findings as part of the portfolio uploaded to GitHub.

By completing all these tasks, I have gained experience in data analysis and the use of SQL to address business problems. The results of my work will be integrated into my GitHub portfolio to showcase my data analysis and SQL skills to potential employers.

First use (Arsyuddin , 2023)

Subsequent use (Akinyemi, 2023)