



SQL CASE-STUDY



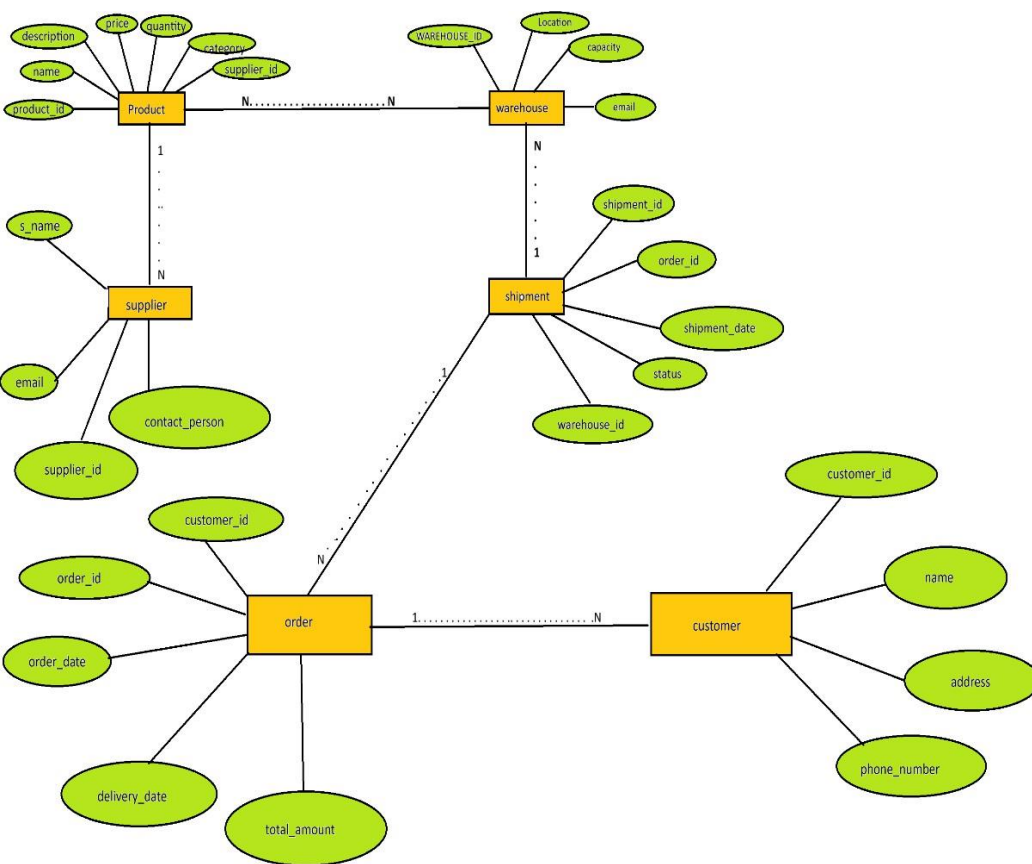
TOPIC:-INVENTORY MANAGEMENT SYSTEM

PROBLEM STATEMENT

P

The current inventory management system lacks real-time tracking and reporting capabilities, leading to inefficiencies such as overstocking, stockouts. This results in increased operational costs, decreased customer satisfaction, and difficulties in data analysis for strategic decision-making. An improved system is needed to streamline inventory control, enhance accuracy, and provide actionable insights for better resource allocation.





E-R DIAGRAM FOR INVENTORY MANAGEMENT SYSTEM










TABLES FOR ATTRIBUTES:-

Result Grid   Filter Rows:






	category_id	category_name
▶	1	Electronics
	2	Clothing
	3	Home Appliances
	4	Books
	5	Furniture
✱	NULL	NULL

Result Grid   Filter Rows: Edit:  

	supplier_id	supplier_name	contact_info
▶	1	Supplier A	contact@supplierA.com
	2	Supplier B	contact@supplierB.com
	3	Supplier C	contact@supplierC.com
✱	NULL	NULL	NULL

Result Grid   Filter Rows: Edit:    Export/Import:  

	product_id	product_name	category_id	supplier_id	unit_price	quantity_in_stock
▶	1	Laptop	1	1	999.99	50
	2	Smartphone	1	1	599.99	100
	3	T-shirt	2	2	19.99	200
	4	Jeans	2	2	39.99	150
	5	Blender	3	3	49.99	75
	6	Microwave	3	3	99.99	30
	7	Fiction Book	4	2	9.99	300
	8	Coffee Table	5	3	199.99	20
✱	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid   Filter Rows: Edit:   

	order_id	product_id	quantity_ordered	order_date
▶	1	1	2	2024-01-10
	2	2	5	2024-01-15
	3	4	3	2024-01-20
	4	3	10	2024-01-25
	5	5	1	2024-01-30
	6	6	2	2024-02-05
	7	1	1	2024-02-10
	8	2	4	2024-02-15
	9	8	1	2024-02-20
✱	NULL	NULL	NULL	NULL

Orders 69 ×

RAW DATA INFORMATION

```
create database db;
```

```
use db;
```

```
CREATE TABLE Categories (  
    category_id INT PRIMARY KEY,  
    category_name VARCHAR(50)  
);
```

```
INSERT INTO Categories (category_id, category_name) VALUES  
(1, 'Electronics'),  
(2, 'Clothing'),  
(3, 'Home Appliances'),  
(4, 'Books'),  
(5, 'Furniture');
```

```
describe Categories;
```

```
select * from Categories;
```

```
CREATE TABLE Suppliers (  
    supplier_id INT PRIMARY KEY,  
    supplier_name VARCHAR(100),  
    contact_info VARCHAR(100)  
);
```

```
drop table Suppliers;
```

```
INSERT INTO Suppliers (supplier_id, supplier_name, contact_info)  
VALUES
```

```
(1, 'Supplier A', 'contact@supplierA.com'),
```

```
(2, 'Supplier B', 'contact@supplierB.com'),
```

```
(3, 'Supplier C', 'contact@supplierC.com');
```

```
describe Suppliers;
```

```
select * from Suppliers;
```

```
CREATE TABLE Products (
```

```
    product_id INT PRIMARY KEY,
```

```
    product_name VARCHAR(100),
```

```
    category_id INT,
```

```
    supplier_id INT,
```

```
    unit_price DECIMAL(10, 2),
```

```
    quantity_in_stock INT,
```

```
    FOREIGN KEY (category_id) REFERENCES Categories(category_id),
```

```
    FOREIGN KEY (supplier_id) REFERENCES Suppliers(supplier_id)
```

```
);
```

```
INSERT INTO Products (product_id, product_name, category_id,  
supplier_id, unit_price, quantity_in_stock) VALUES
```

```
(1, 'Laptop', 1, 1, 999.99, 50),
```

```
(2, 'Smartphone', 1, 1, 599.99, 100),
```

```
(3, 'T-shirt', 2, 2, 19.99, 200),
```

```
(4, 'Jeans', 2, 2, 39.99, 150),
```

```
(5, 'Blender', 3, 3, 49.99, 75),
```

```
(6, 'Microwave', 3, 3, 99.99, 30),
```

```
(7, 'Fiction Book', 4, 2, 9.99, 300),
```

```
(8, 'Coffee Table', 5, 3, 199.99, 20);
```

```
describe Products;
```

```
select * from Products;
```

```
CREATE TABLE Orders (  
    order_id INT PRIMARY KEY,  
    product_id INT,  
    quantity_ordered INT,  
    order_date DATE,  
    FOREIGN KEY (product_id) REFERENCES Products(product_id)  
);
```

```
INSERT INTO Orders (order_id, product_id, quantity_ordered,  
order_date) VALUES
```

```
(1, 1, 2, '2024-01-10'),
```

```
(2, 2, 5, '2024-01-15'),
```

```
(3, 4, 3, '2024-01-20'),
```

```
(4, 3, 10, '2024-01-25'),
```

```
(5, 5, 1, '2024-01-30'),
```

```
(6, 6, 2, '2024-02-05'),
```

```
(7, 1, 1, '2024-02-10'),
```

```
(8, 2, 4, '2024-02-15'),
```

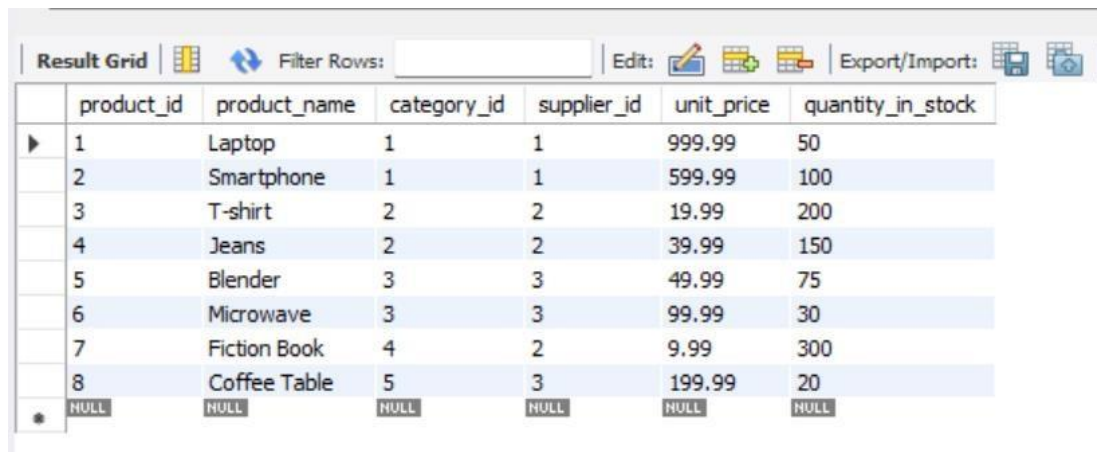
```
(9, 8, 1, '2024-02-20');
```

```
describe Orders;
```

```
select * from Orders;
```

1. List all products.

```
SELECT * FROM Products;
```

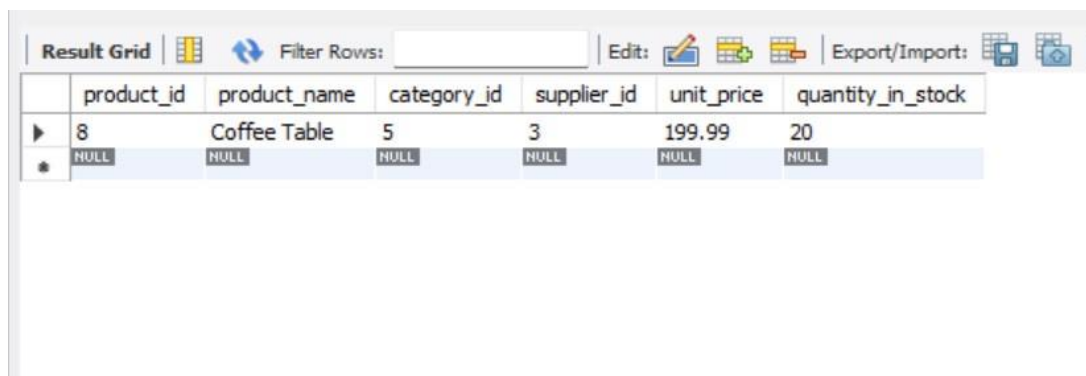


The screenshot shows a database application interface with a 'Result Grid' tab. The grid contains 8 rows of data. The first row is highlighted with a blue background. The columns are: product_id, product_name, category_id, supplier_id, unit_price, and quantity_in_stock. The data is as follows:

	product_id	product_name	category_id	supplier_id	unit_price	quantity_in_stock
▶	1	Laptop	1	1	999.99	50
	2	Smartphone	1	1	599.99	100
	3	T-shirt	2	2	19.99	200
	4	Jeans	2	2	39.99	150
	5	Blender	3	3	49.99	75
	6	Microwave	3	3	99.99	30
	7	Fiction Book	4	2	9.99	300
	8	Coffee Table	5	3	199.99	20
★	NULL	NULL	NULL	NULL	NULL	NULL

2. Find products with a quantity less than 20.

```
SELECT * FROM Products WHERE quantity_in_stock < 20;
```





The screenshot shows the same database application interface, but the result grid now only contains 1 row of data. The first row is highlighted with a blue background. The columns are: product_id, product_name, category_id, supplier_id, unit_price, and quantity_in_stock. The data is as follows:

	product_id	product_name	category_id	supplier_id	unit_price	quantity_in_stock
▶	8	Coffee Table	5	3	199.99	20
★	NULL	NULL	NULL	NULL	NULL	NULL



3. Get the total number of products in stock.

```
SELECT SUM(quantity_in_stock) AS total_stock FROM Products;
```

Result Grid			 Filter Rows: <input type="text"/>
	total_stock		
▶	925		

4. Find the average unit price of products.

```
SELECT AVG(unit_price) AS average_price FROM Products;
```

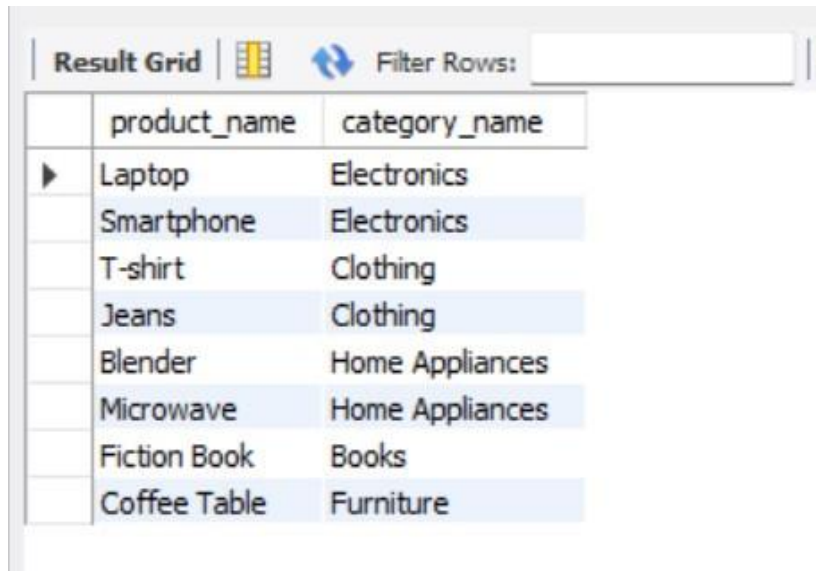
Result Grid			 Filter Rows: <input type="text"/>
	total_stock		
▶	925		

5. List products and their categories.

```
SELECT p.product_name, c.category_name
```

```
FROM Products p
```

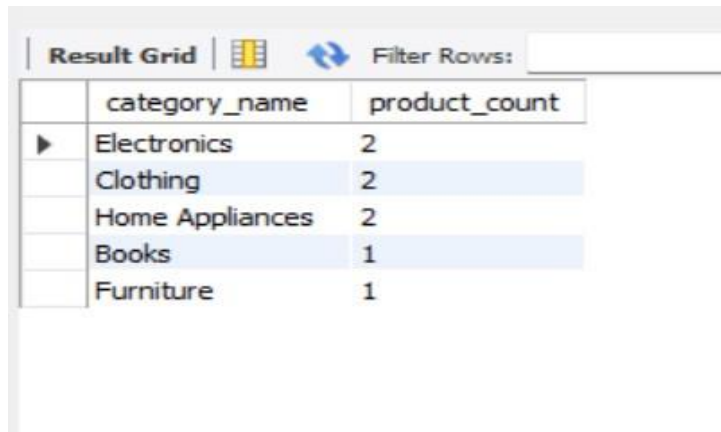
```
JOIN Categories c ON p.category_id = c.category_id;
```



	product_name	category_name
▶	Laptop	Electronics
	Smartphone	Electronics
	T-shirt	Clothing
	Jeans	Clothing
	Blender	Home Appliances
	Microwave	Home Appliances
	Fiction Book	Books
	Coffee Table	Furniture

6. Get the number of products per category.

```
SELECT c.category_name, COUNT(p.product_id) AS product_count  
FROM Categories c  
LEFT JOIN Products p ON c.category_id = p.category_id  
GROUP BY c.category_name;
```

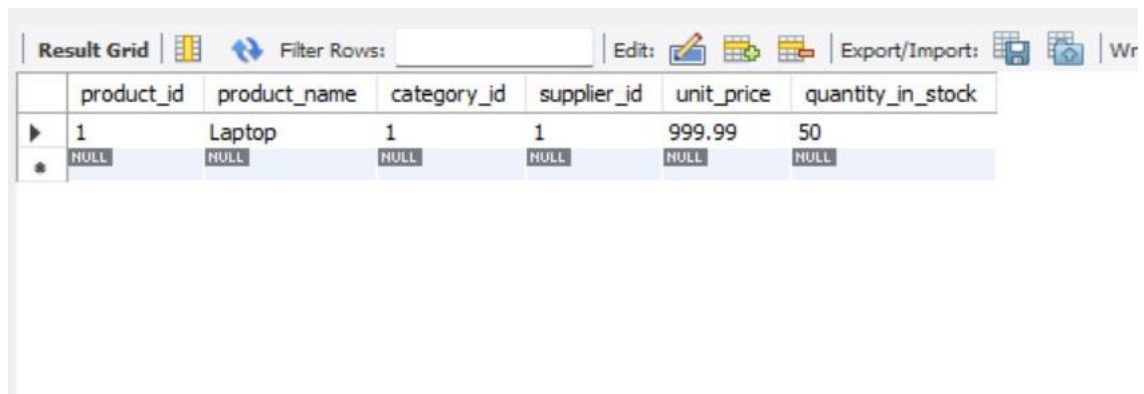


The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the results of the SQL query, showing the category name and the count of products for each category. The categories are Electronics, Clothing, Home Appliances, Books, and Furniture, with product counts of 2, 2, 2, 1, and 1 respectively. The 'Filter Rows' field is empty.

	category_name	product_count
▶	Electronics	2
	Clothing	2
	Home Appliances	2
	Books	1
	Furniture	1

7. Find the highest priced product.

```
SELECT * FROM Products WHERE unit_price = (SELECT MAX(unit_price)
FROM Products);
```

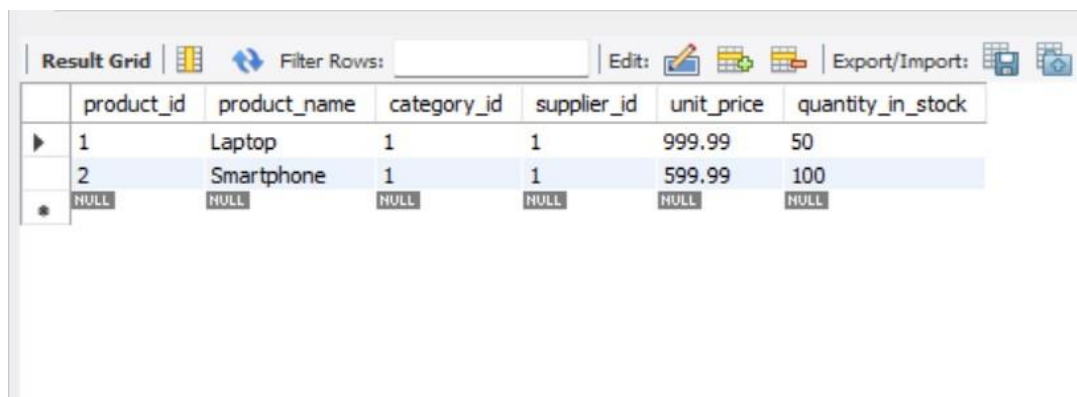


The screenshot shows a database query result grid. The grid has a toolbar at the top with icons for 'Result Grid', 'Filter Rows', 'Edit', and 'Export/Import'. Below the toolbar is a table with the following data:

	product_id	product_name	category_id	supplier_id	unit_price	quantity_in_stock
▶	1	Laptop	1	1	999.99	50
*	NULL	NULL	NULL	NULL	NULL	NULL

8. Get products supplied by a specific supplier.

```
SELECT * FROM Products WHERE supplier_id = (SELECT supplier_id
FROM Suppliers WHERE supplier_name = 'Supplier A');
```

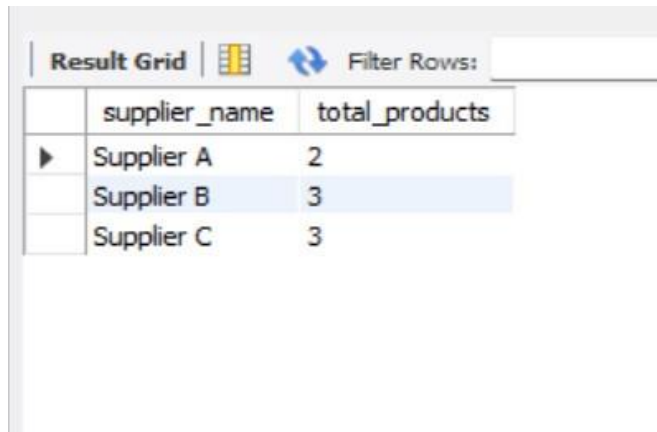


The screenshot shows a database query result grid. The grid has a toolbar at the top with icons for 'Result Grid', 'Filter Rows', 'Edit', and 'Export/Import'. Below the toolbar is a table with the following data:

	product_id	product_name	category_id	supplier_id	unit_price	quantity_in_stock
▶	1	Laptop	1	1	999.99	50
	2	Smartphone	1	1	599.99	100
*	NULL	NULL	NULL	NULL	NULL	NULL

9. List suppliers and their total products supplied.

```
SELECT s.supplier_name, COUNT(p.product_id) AS total_products
FROM Suppliers s
LEFT JOIN Products p ON s.supplier_id = p.supplier_id
GROUP BY s.supplier_name;
```

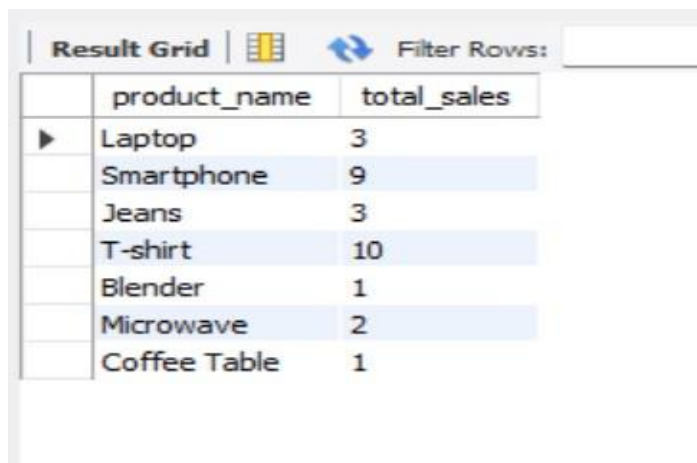


The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the results of a SQL query. The columns are 'supplier_name' and 'total_products'. The data is as follows:

	supplier_name	total_products
▶	Supplier A	2
	Supplier B	3
	Supplier C	3

10. Get the total sales for each product.

```
SELECT p.product_name, SUM(o.quantity_ordered) AS total_sales
FROM Products p
JOIN Orders o ON p.product_id = o.product_id
GROUP BY p.product_name;
```

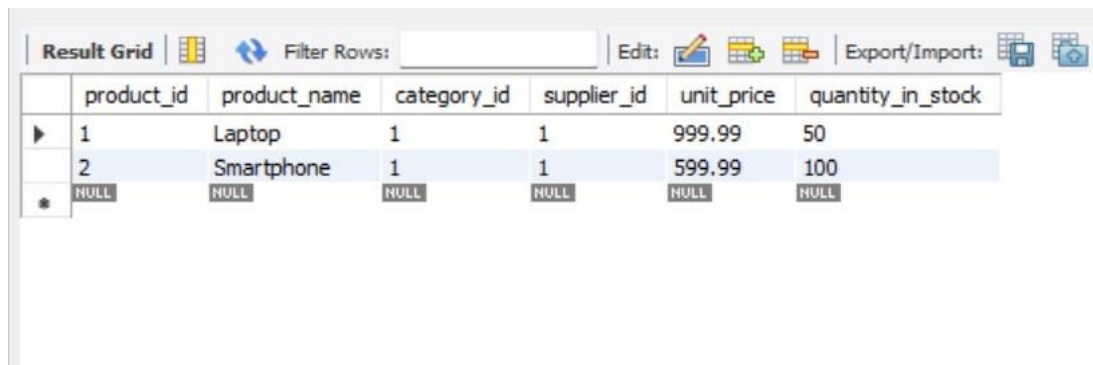


The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the results of a SQL query. The columns are 'product_name' and 'total_sales'. The data is as follows:

	product_name	total_sales
▶	Laptop	3
	Smartphone	9
	Jeans	3
	T-shirt	10
	Blender	1
	Microwave	2
	Coffee Table	1

11. Find products with a unit price higher than the average.

```
SELECT * FROM Products WHERE unit_price > (SELECT AVG(unit_price)
FROM Products);
```

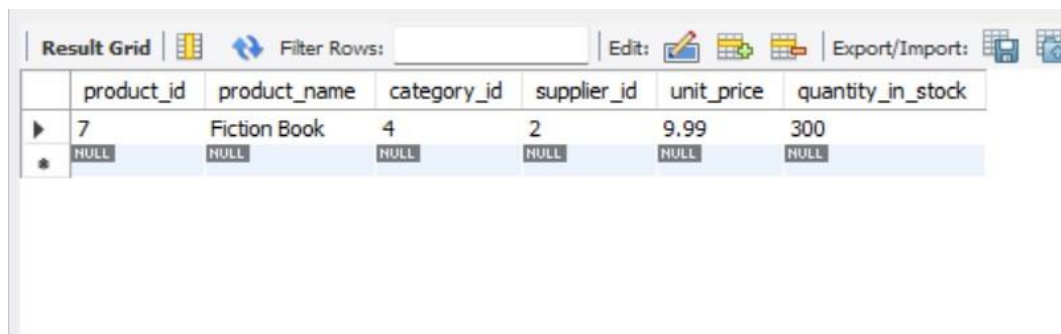


The screenshot shows a database query result grid with the following columns: product_id, product_name, category_id, supplier_id, unit_price, and quantity_in_stock. The results are as follows:

	product_id	product_name	category_id	supplier_id	unit_price	quantity_in_stock
▶	1	Laptop	1	1	999.99	50
	2	Smartphone	1	1	599.99	100
★	NULL	NULL	NULL	NULL	NULL	NULL

12. List products that have never been ordered.

```
SELECT * FROM Products WHERE product_id NOT IN (SELECT product_id
FROM Orders);
```



The screenshot shows a database query result grid with the following columns: product_id, product_name, category_id, supplier_id, unit_price, and quantity_in_stock. The results are as follows:

	product_id	product_name	category_id	supplier_id	unit_price	quantity_in_stock
▶	7	Fiction Book	4	2	9.99	300
★	NULL	NULL	NULL	NULL	NULL	NULL

13. Get the category with the most products.

```
SELECT category_id  
FROM Products  
GROUP BY category_id  
ORDER BY COUNT(*) DESC  
LIMIT 1;
```

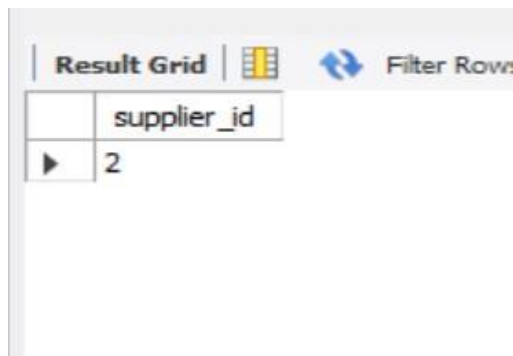


The screenshot shows a database query result grid. At the top, there is a tab labeled 'Result Grid' and a 'Filter Rows' button. The grid has two columns: 'category_id' and an unnamed column. The first row contains the value '1' in the unnamed column.

	category_id
▶	1

14. Find the supplier with the most products.

```
SELECT supplier_id  
FROM Products  
GROUP BY supplier_id  
ORDER BY COUNT(*) DESC  
LIMIT 1;
```

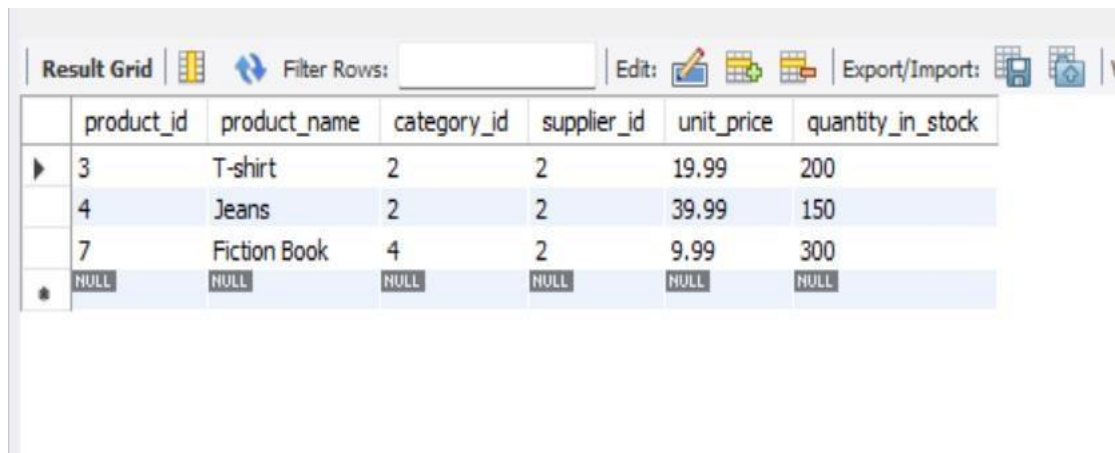


The screenshot shows a database query result grid. At the top, there is a tab labeled 'Result Grid' and a 'Filter Rows' button. The grid has two columns: 'supplier_id' and an unnamed column. The first row contains the value '2' in the unnamed column.

	supplier_id
▶	2

15. Get products with a quantity greater than the average quantity.

```
SELECT * FROM Products WHERE quantity_in_stock > (SELECT  
AVG(quantity_in_stock) FROM Products);
```

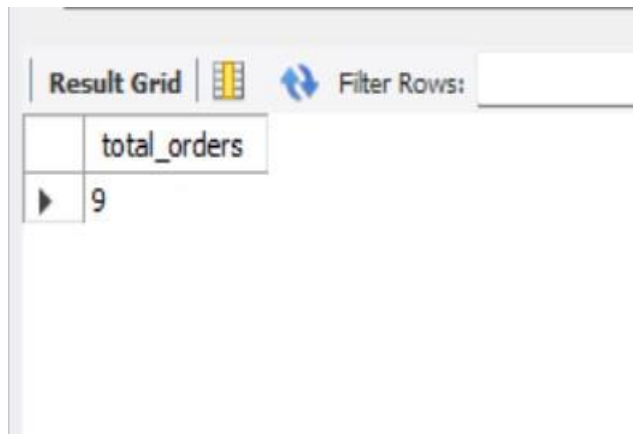


The screenshot shows a database query result grid. The grid has a toolbar at the top with icons for 'Result Grid', 'Filter Rows', 'Edit', and 'Export/Import'. The grid contains the following data:

	product_id	product_name	category_id	supplier_id	unit_price	quantity_in_stock
▶	3	T-shirt	2	2	19.99	200
	4	Jeans	2	2	39.99	150
	7	Fiction Book	4	2	9.99	300
★	NULL	NULL	NULL	NULL	NULL	NULL

16. Find the total number of orders.

```
SELECT COUNT(*) AS total_orders FROM Orders;
```

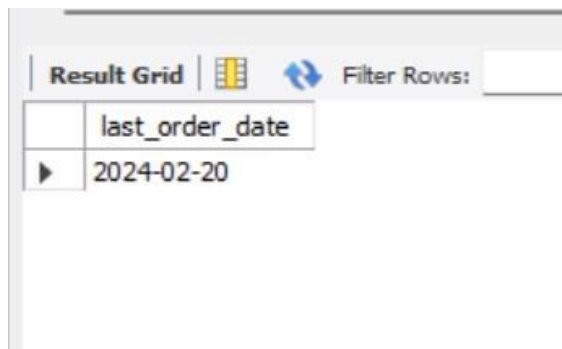


The screenshot shows a database query result grid. The grid has a toolbar at the top with icons for 'Result Grid', 'Filter Rows', and 'Edit'. The grid contains the following data:

	total_orders
▶	9

17. Get the most recent order date.

```
SELECT MAX(order_date) AS last_order_date FROM Orders;
```

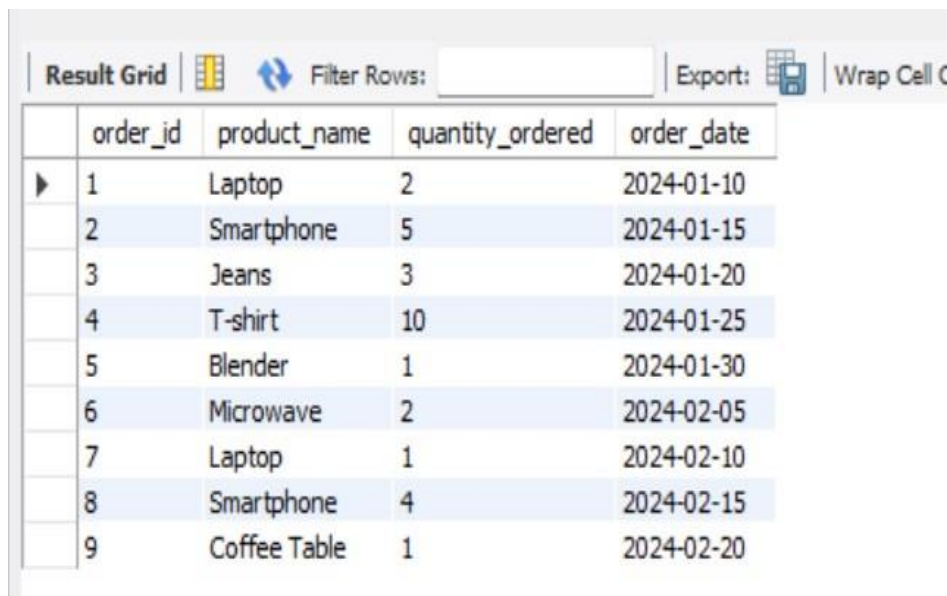


The screenshot shows a database interface with a 'Result Grid' tab. The grid contains a single row with the column name 'last_order_date' and its value '2024-02-20'. Above the grid, there are icons for a grid, a refresh button, and a 'Filter Rows' input field.

last_order_date
2024-02-20

18. List orders with product names.

```
SELECT o.order_id, p.product_name, o.quantity_ordered, o.order_date  
FROM Orders o  
JOIN Products p ON o.product_id = p.product_id;
```

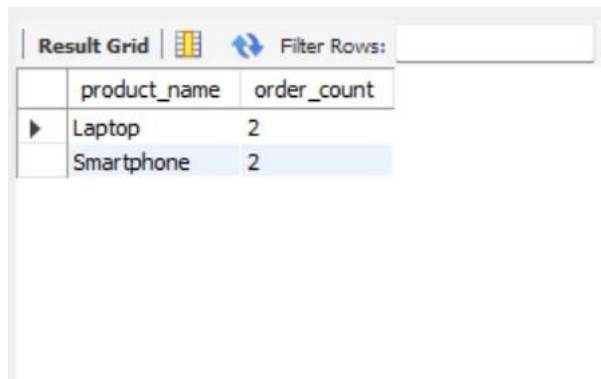


The screenshot shows a database interface with a 'Result Grid' tab. The grid displays a list of orders with columns: order_id, product_name, quantity_ordered, and order_date. The data is as follows:

order_id	product_name	quantity_ordered	order_date
1	Laptop	2	2024-01-10
2	Smartphone	5	2024-01-15
3	Jeans	3	2024-01-20
4	T-shirt	10	2024-01-25
5	Blender	1	2024-01-30
6	Microwave	2	2024-02-05
7	Laptop	1	2024-02-10
8	Smartphone	4	2024-02-15
9	Coffee Table	1	2024-02-20

19. Find products that have been ordered more than 1 times.

```
SELECT p.product_name, COUNT(o.order_id) AS order_count  
FROM Products p  
JOIN Orders o ON p.product_id = o.product_id  
GROUP BY p.product_name  
HAVING COUNT(o.order_id) > 1;
```

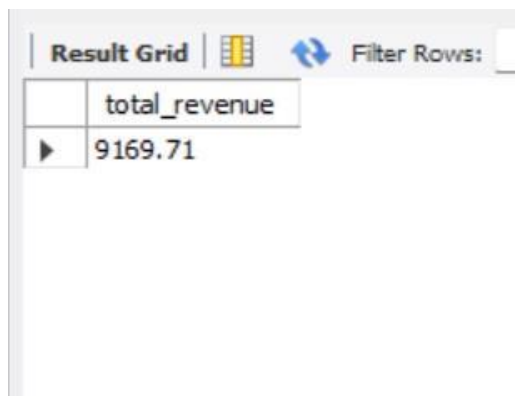


The screenshot shows a 'Result Grid' window with a 'Filter Rows' input field. The grid contains two columns: 'product_name' and 'order_count'. There are two rows of data: 'Laptop' with an order count of 2, and 'Smartphone' with an order count of 2. The 'Smartphone' row is highlighted in blue.

	product_name	order_count
▶	Laptop	2
	Smartphone	2

20. Get the total revenue generated from orders.

```
SELECT SUM(p.unit_price * o.quantity_ordered) AS total_revenue  
FROM Orders o  
JOIN Products p ON o.product_id = p.product_id;
```

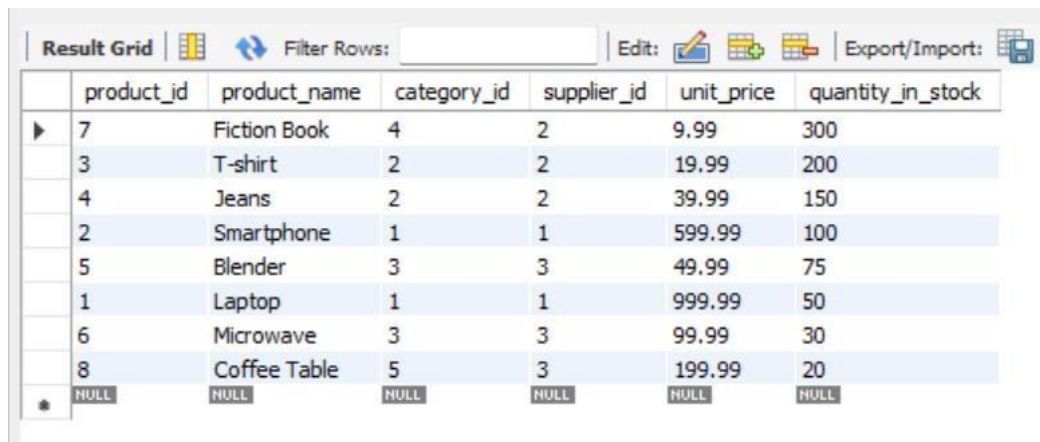


The screenshot shows a 'Result Grid' window with a 'Filter Rows' input field. The grid contains one column: 'total_revenue'. There is one row of data showing the total revenue as 9169.71.

	total_revenue
▶	9169.71

21. List products sorted by quantity in stock descending.

```
SELECT * FROM Products ORDER BY quantity_in_stock DESC;
```

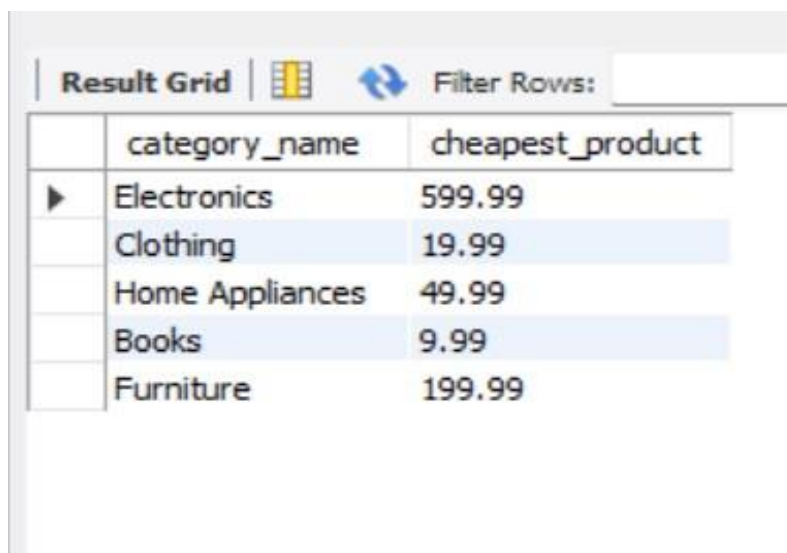


The screenshot shows a database interface with a 'Result Grid' tab. The grid displays a list of products sorted by 'quantity_in_stock' in descending order. The columns are: product_id, product_name, category_id, supplier_id, unit_price, and quantity_in_stock. The data is as follows:

product_id	product_name	category_id	supplier_id	unit_price	quantity_in_stock
7	Fiction Book	4	2	9.99	300
3	T-shirt	2	2	19.99	200
4	Jeans	2	2	39.99	150
2	Smartphone	1	1	599.99	100
5	Blender	3	3	49.99	75
1	Laptop	1	1	999.99	50
6	Microwave	3	3	99.99	30
8	Coffee Table	5	3	199.99	20
NULL	NULL	NULL	NULL	NULL	NULL

22. Find the cheapest product in each category.

```
SELECT c.category_name, MIN(p.unit_price) AS cheapest_product  
FROM Categories c  
JOIN Products p ON c.category_id = p.category_id  
GROUP BY c.category_name;
```

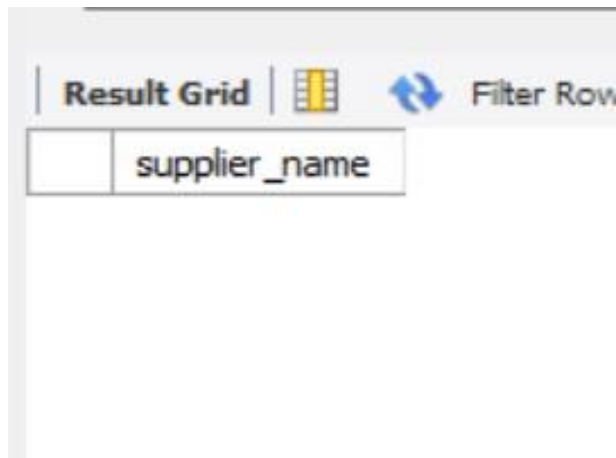


The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the cheapest product for each category. The columns are: category_name and cheapest_product. The data is as follows:

category_name	cheapest_product
Electronics	599.99
Clothing	19.99
Home Appliances	49.99
Books	9.99
Furniture	199.99

23. Get suppliers with no products.

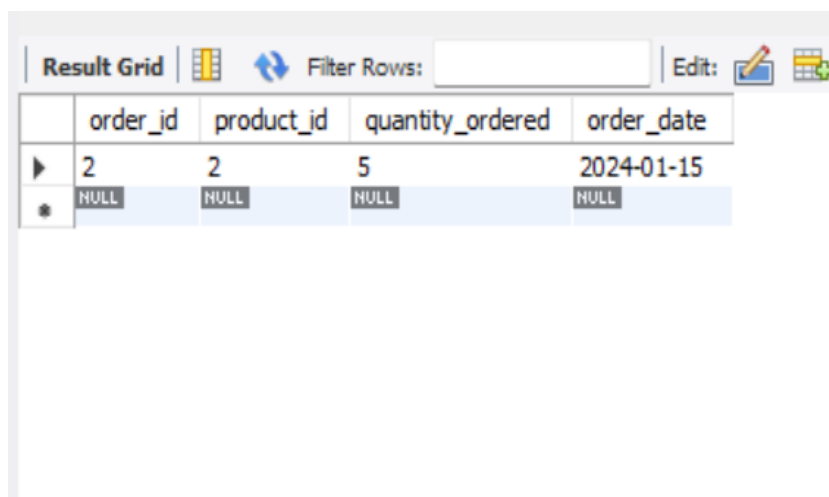
```
SELECT s.supplier_name  
FROM Suppliers s  
LEFT JOIN Products p ON s.supplier_id = p.supplier_id  
WHERE p.product_id IS NULL;
```



The screenshot shows a database interface with a 'Result Grid' tab. The grid has one column labeled 'supplier_name'. The interface includes icons for a grid, a refresh button, and a 'Filter Rows' button.

24. List all orders from a specific date.

```
SELECT * FROM Orders WHERE order_date = '2024-01-15';
```

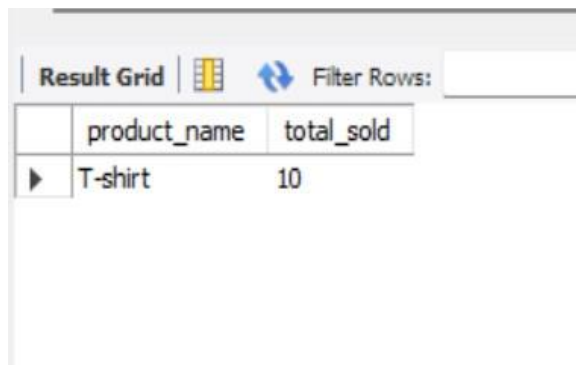


The screenshot shows a database interface with a 'Result Grid' tab. The grid has four columns: 'order_id', 'product_id', 'quantity_ordered', and 'order_date'. It contains one data row and one null row. The interface includes icons for a grid, a refresh button, a 'Filter Rows' input field, and an 'Edit' button.

	order_id	product_id	quantity_ordered	order_date
▶	2	2	5	2024-01-15
*	NULL	NULL	NULL	NULL

25. Find the top-selling product.

```
SELECT p.product_name, SUM(o.quantity_ordered) AS total_sold
FROM Products p
JOIN Orders o ON p.product_id = o.product_id
GROUP BY p.product_name
ORDER BY total_sold DESC
LIMIT 1;
```

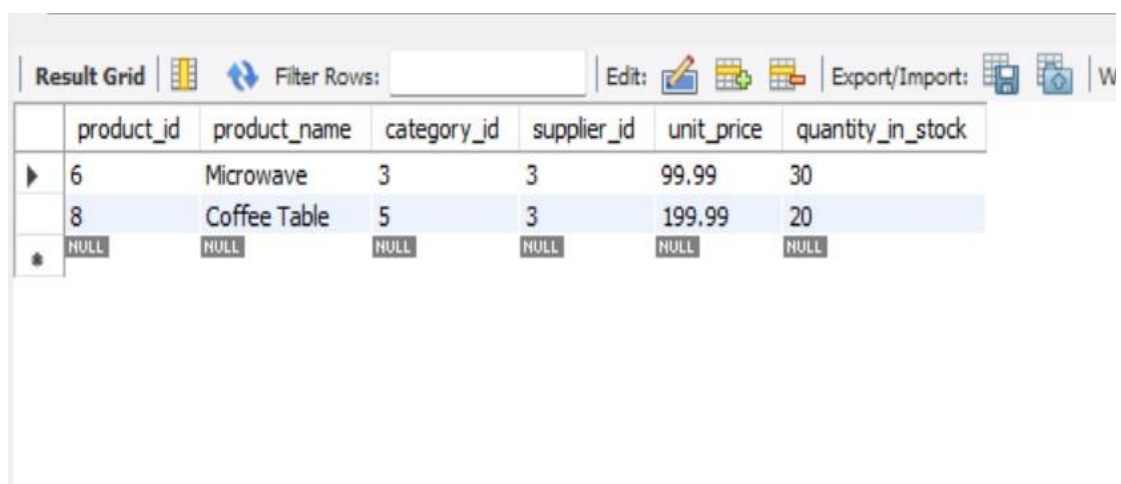


The screenshot shows a database query result grid. At the top, there is a tab labeled 'Result Grid' and a 'Filter Rows:' input field. The grid has two columns: 'product_name' and 'total_sold'. The first row shows 'T-shirt' with a 'total_sold' value of 10.

	product_name	total_sold
▶	T-shirt	10

26. List products with a stock level below a certain threshold.

```
SELECT * FROM Products WHERE quantity_in_stock < 50;
```

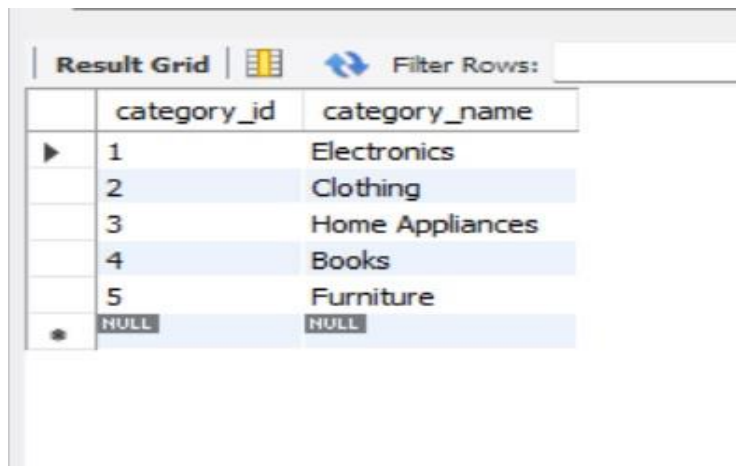


The screenshot shows a database query result grid. At the top, there is a tab labeled 'Result Grid', a 'Filter Rows:' input field, and buttons for 'Edit', 'Export/Import', and 'W'. The grid has seven columns: 'product_id', 'product_name', 'category_id', 'supplier_id', 'unit_price', and 'quantity_in_stock'. The first two rows are highlighted in blue. The first row shows product_id 6, Microwave, category_id 3, supplier_id 3, unit_price 99.99, and quantity_in_stock 30. The second row shows product_id 8, Coffee Table, category_id 5, supplier_id 3, unit_price 199.99, and quantity_in_stock 20. The third row is a summary row with all NULL values.

	product_id	product_name	category_id	supplier_id	unit_price	quantity_in_stock
▶	6	Microwave	3	3	99.99	30
	8	Coffee Table	5	3	199.99	20
*	NULL	NULL	NULL	NULL	NULL	NULL

27. List all unique product categories.

```
SELECT DISTINCT category_id, category_name FROM Categories;
```

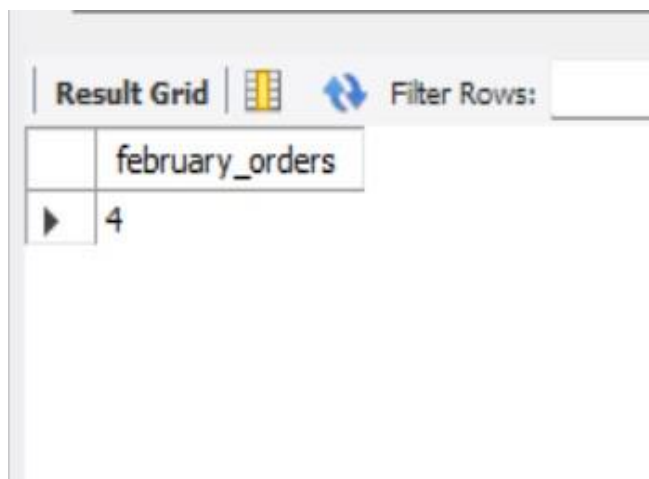


The screenshot shows a 'Result Grid' interface with a 'Filter Rows' input field. The grid contains a table with two columns: 'category_id' and 'category_name'. The data is as follows:

	category_id	category_name
▶	1	Electronics
	2	Clothing
	3	Home Appliances
	4	Books
	5	Furniture
✱	NULL	NULL

28. Count how many orders were placed in February 2024.

```
SELECT COUNT(*) AS february_orders FROM Orders WHERE order_date  
BETWEEN '2024-02-01' AND '2024-02-29';
```

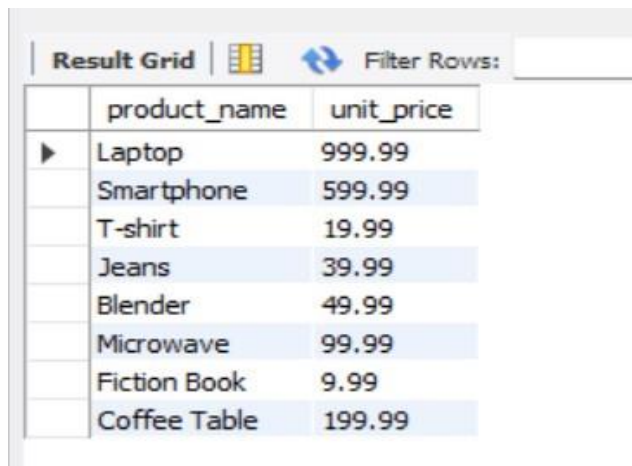


The screenshot shows a 'Result Grid' interface with a 'Filter Rows' input field. The grid contains a table with one column: 'february_orders'. The data is as follows:

	february_orders
▶	4

29. Get the product names along with their unit prices.

```
SELECT product_name, unit_price FROM Products;
```



The screenshot shows a database interface with a 'Result Grid' tab. The grid displays two columns: 'product_name' and 'unit_price'. There are nine rows of data, each with a small triangle icon in the first column. The rows are: Laptop (999.99), Smartphone (599.99), T-shirt (19.99), Jeans (39.99), Blender (49.99), Microwave (99.99), Fiction Book (9.99), and Coffee Table (199.99). The 'Filter Rows' field is empty.

	product_name	unit_price
▶	Laptop	999.99
	Smartphone	599.99
	T-shirt	19.99
	Jeans	39.99
	Blender	49.99
	Microwave	99.99
	Fiction Book	9.99
	Coffee Table	199.99

30. List products that belong to the 'Electronics' category.

```
SELECT p.product_name FROM Products p
```

```
JOIN Categories c ON p.category_id = c.category_id
```

```
WHERE c.category_name = 'Electronics';
```



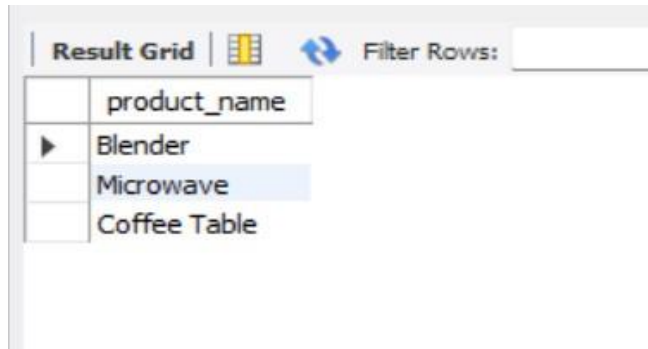
The screenshot shows a database interface with a 'Result Grid' tab. The grid displays one column: 'product_name'. There are two rows of data, each with a small triangle icon in the first column. The rows are: Laptop and Smartphone. The 'Filter Rows' field is empty.

	product_name
▶	Laptop
	Smartphone

31. Find all products supplied by 'Supplier C'.

```
SELECT p.product_name FROM Products p
```

```
WHERE p.supplier_id = (SELECT supplier_id FROM Suppliers WHERE  
supplier_name = 'Supplier C');
```



The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the results of a query. The first column is labeled 'product_name'. The rows listed are 'Blender', 'Microwave', and 'Coffee Table'. The 'Microwave' row is currently selected, highlighted in blue. Above the grid, there is a 'Filter Rows:' input field.

	product_name
▶	Blender
	Microwave
	Coffee Table

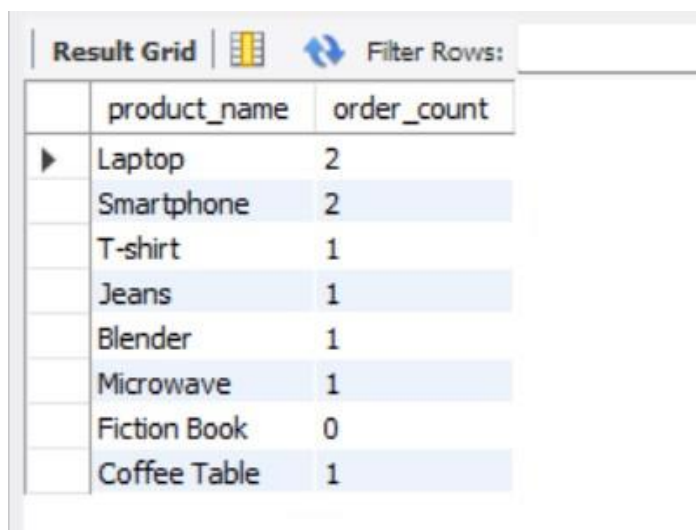
32. Count the number of orders placed for each product.

```
SELECT p.product_name, COUNT(o.order_id) AS order_count
```

```
FROM Products p
```

```
LEFT JOIN Orders o ON p.product_id = o.product_id
```

```
GROUP BY p.product_name;
```

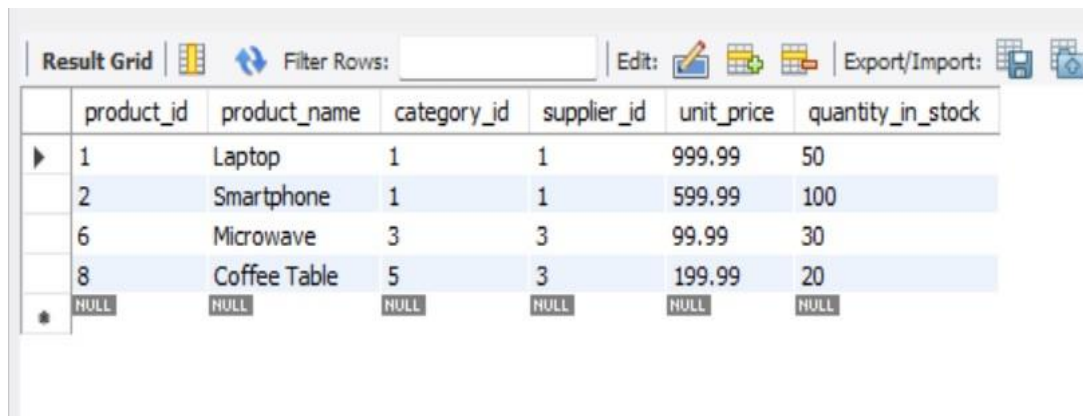


The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the results of a query. The first column is labeled 'product_name' and the second column is labeled 'order_count'. The rows listed are 'Laptop' (2), 'Smartphone' (2), 'T-shirt' (1), 'Jeans' (1), 'Blender' (1), 'Microwave' (1), 'Fiction Book' (0), and 'Coffee Table' (1). The 'Smartphone' row is currently selected, highlighted in blue. Above the grid, there is a 'Filter Rows:' input field.

	product_name	order_count
▶	Laptop	2
	Smartphone	2
	T-shirt	1
	Jeans	1
	Blender	1
	Microwave	1
	Fiction Book	0
	Coffee Table	1

33. Find products with a unit price greater than \$50.

```
SELECT * FROM Products WHERE unit_price > 50;
```

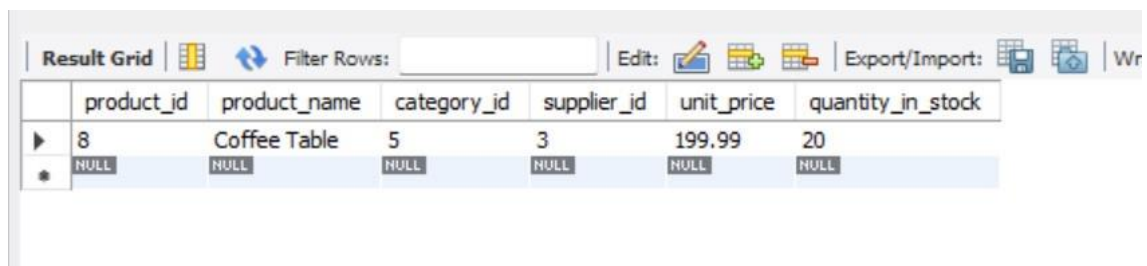


The screenshot shows a database query result grid with the following columns: product_id, product_name, category_id, supplier_id, unit_price, and quantity_in_stock. The results are as follows:

	product_id	product_name	category_id	supplier_id	unit_price	quantity_in_stock
▶	1	Laptop	1	1	999.99	50
	2	Smartphone	1	1	599.99	100
	6	Microwave	3	3	99.99	30
	8	Coffee Table	5	3	199.99	20
✱	NULL	NULL	NULL	NULL	NULL	NULL

34. Find the latest product added (highest product ID).

```
SELECT * FROM Products WHERE product_id = (SELECT MAX(product_id) FROM Products);
```

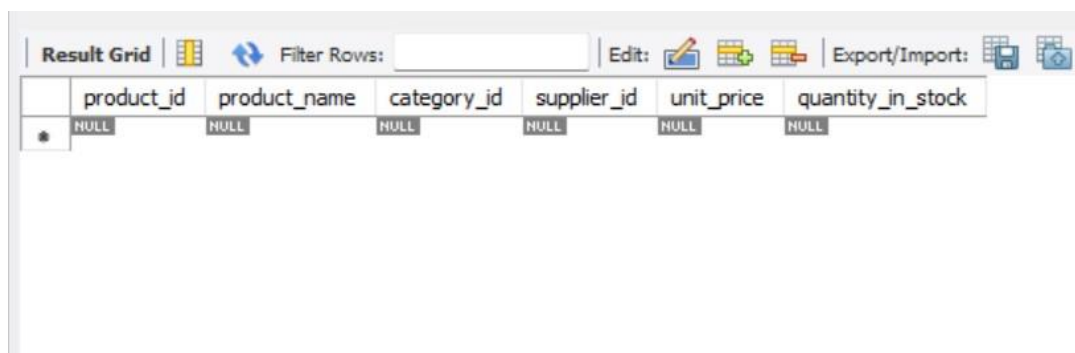


The screenshot shows a database query result grid with the following columns: product_id, product_name, category_id, supplier_id, unit_price, and quantity_in_stock. The results are as follows:

	product_id	product_name	category_id	supplier_id	unit_price	quantity_in_stock
▶	8	Coffee Table	5	3	199.99	20
✱	NULL	NULL	NULL	NULL	NULL	NULL

35. Find products that are out of stock.

```
SELECT * FROM Products WHERE quantity_in_stock = 0;
```

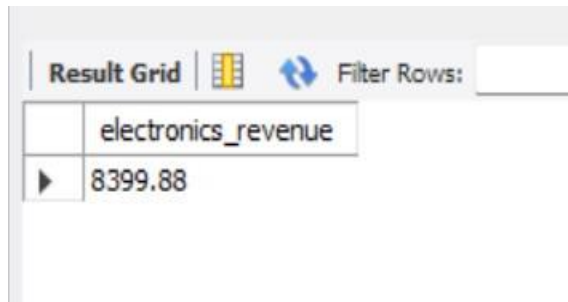


The screenshot shows a database query result grid with the following columns: product_id, product_name, category_id, supplier_id, unit_price, and quantity_in_stock. The results are as follows:

	product_id	product_name	category_id	supplier_id	unit_price	quantity_in_stock
✱	NULL	NULL	NULL	NULL	NULL	NULL

36. Get the total revenue generated from 'Electronics' products.

```
SELECT SUM(p.unit_price * o.quantity_ordered) AS electronics_revenue  
FROM Orders o  
JOIN Products p ON o.product_id = p.product_id  
WHERE p.category_id = (SELECT category_id FROM Categories WHERE  
category_name = 'Electronics');
```

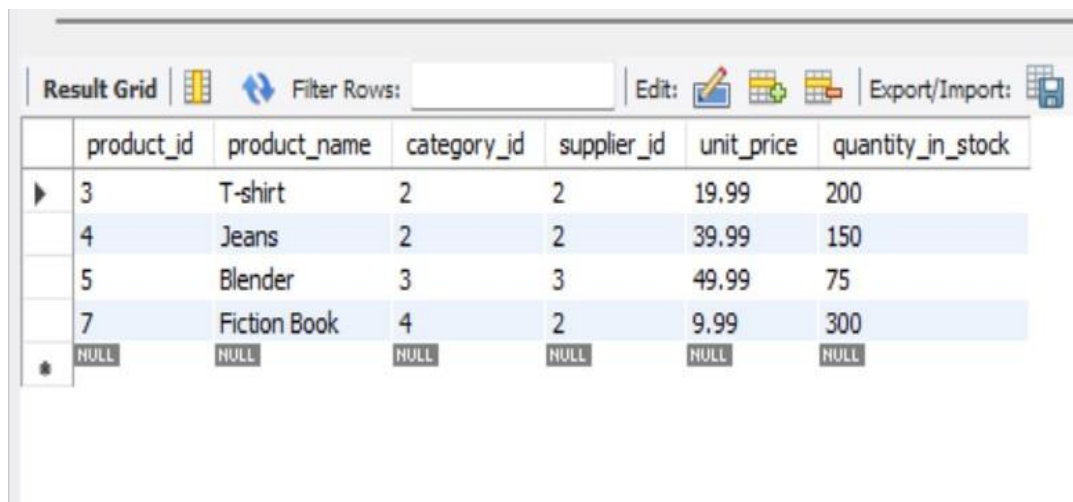


The screenshot shows a database interface with a 'Result Grid' tab. The grid has two columns: 'electronics_revenue' and a value '8399.88'. Above the grid, there are icons for 'Filter Rows' and a search bar.

	electronics_revenue
▶	8399.88

37. List products that have a quantity greater than 20 and a unit price less than \$50.

```
SELECT * FROM Products WHERE quantity_in_stock > 20 AND unit_price  
< 50;
```



The screenshot shows a database interface with a 'Result Grid' tab. The grid has columns: 'product_id', 'product_name', 'category_id', 'supplier_id', 'unit_price', and 'quantity_in_stock'. The data rows are: (3, T-shirt, 2, 2, 19.99, 200), (4, Jeans, 2, 2, 39.99, 150), (5, Blender, 3, 3, 49.99, 75), (7, Fiction Book, 4, 2, 9.99, 300), and a row of NULL values. Above the grid, there are icons for 'Filter Rows', 'Edit', and 'Export/Import'.

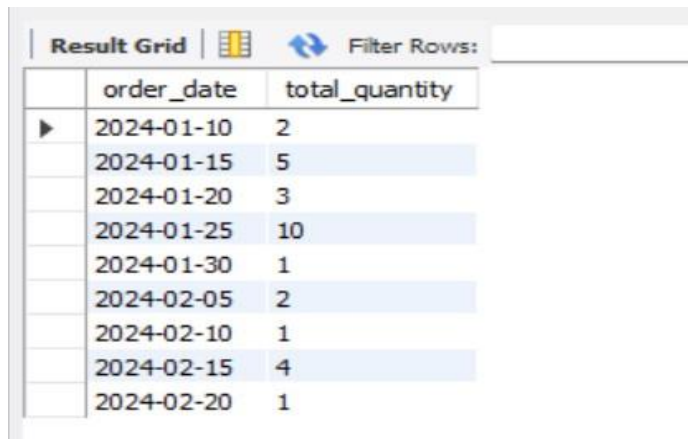
	product_id	product_name	category_id	supplier_id	unit_price	quantity_in_stock
▶	3	T-shirt	2	2	19.99	200
	4	Jeans	2	2	39.99	150
	5	Blender	3	3	49.99	75
	7	Fiction Book	4	2	9.99	300
*	NULL	NULL	NULL	NULL	NULL	NULL

38. Find the total quantity of orders per day.

```
SELECT order_date, SUM(quantity_ordered) AS total_quantity
```

```
FROM Orders
```

```
GROUP BY order_date;
```



The screenshot shows a 'Result Grid' window with a 'Filter Rows' input field. The grid contains two columns: 'order_date' and 'total_quantity'. The data is as follows:

	order_date	total_quantity
▶	2024-01-10	2
	2024-01-15	5
	2024-01-20	3
	2024-01-25	10
	2024-01-30	1
	2024-02-05	2
	2024-02-10	1
	2024-02-15	4
	2024-02-20	1

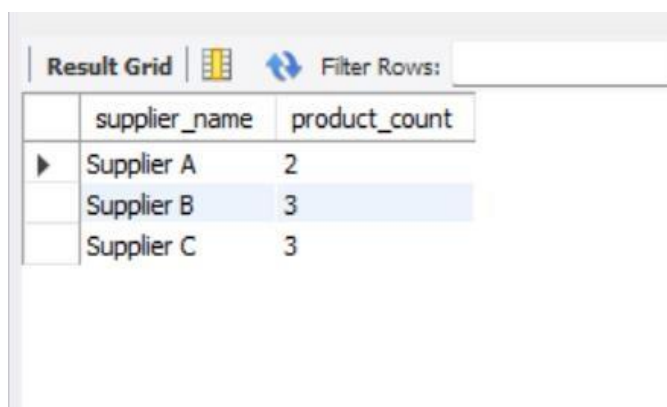
39. Count the number of products in each supplier inventory.

```
SELECT s.supplier_name, COUNT(p.product_id) AS product_count
```

```
FROM Suppliers s
```

```
LEFT JOIN Products p ON s.supplier_id = p.supplier_id
```

```
GROUP BY s.supplier_name;
```



The screenshot shows a 'Result Grid' window with a 'Filter Rows' input field. The grid contains two columns: 'supplier_name' and 'product_count'. The data is as follows:

	supplier_name	product_count
▶	Supplier A	2
	Supplier B	3
	Supplier C	3

40. Find the supplier with the lowest unit price product.

```
SELECT s.supplier_name, MIN(p.unit_price) AS lowest_price
FROM Suppliers s
JOIN Products p ON s.supplier_id = p.supplier_id
GROUP BY s.supplier_name
ORDER BY lowest_price ASC
LIMIT 1;
```

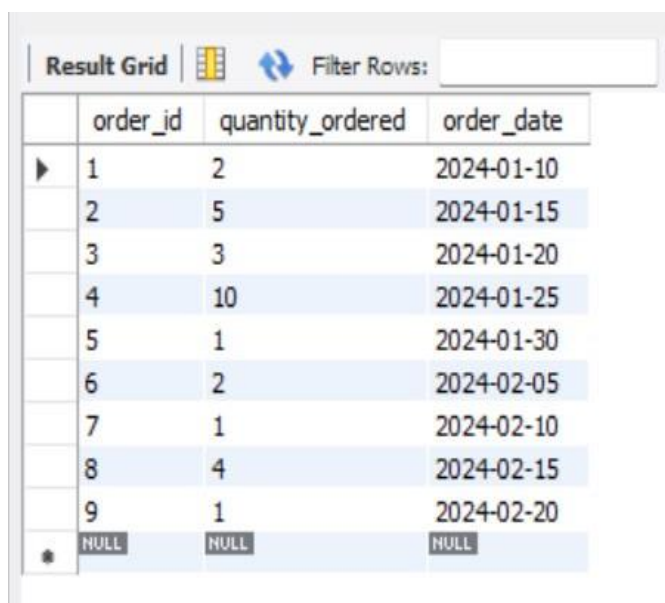


The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the results of a query. The first row is the header with columns 'supplier_name' and 'lowest_price'. The second row shows 'Supplier B' with a 'lowest_price' of 9.99. There is a 'Filter Rows' button and a search bar above the grid.

	supplier_name	lowest_price
▶	Supplier B	9.99

41. List all orders and their corresponding quantities and dates.

```
SELECT o.order_id, o.quantity_ordered, o.order_date
FROM Orders o;
```

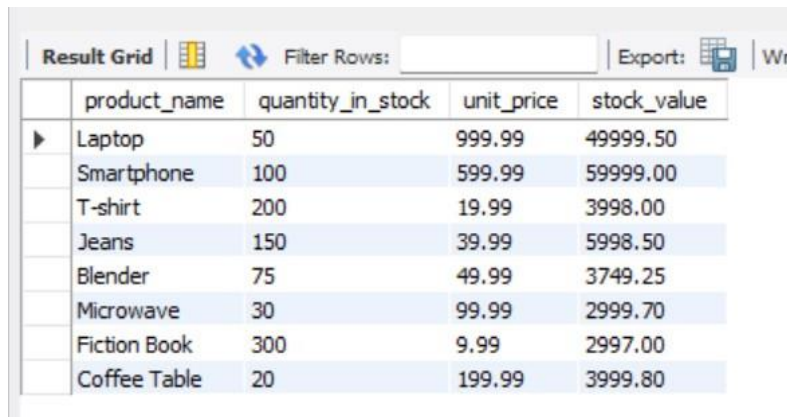


The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the results of a query. The first row is the header with columns 'order_id', 'quantity_ordered', and 'order_date'. The subsequent rows show order details for order IDs 1 through 9, and a final row with NULL values. There is a 'Filter Rows' button and a search bar above the grid.

	order_id	quantity_ordered	order_date
▶	1	2	2024-01-10
	2	5	2024-01-15
	3	3	2024-01-20
	4	10	2024-01-25
	5	1	2024-01-30
	6	2	2024-02-05
	7	1	2024-02-10
	8	4	2024-02-15
	9	1	2024-02-20
✱	NULL	NULL	NULL

42. View for Total Stock Value by Product

```
CREATE VIEW ProductStockValue AS  
  
SELECT product_name, quantity_in_stock, unit_price,  
       (quantity_in_stock * unit_price) AS stock_value  
  
FROM Products;
```

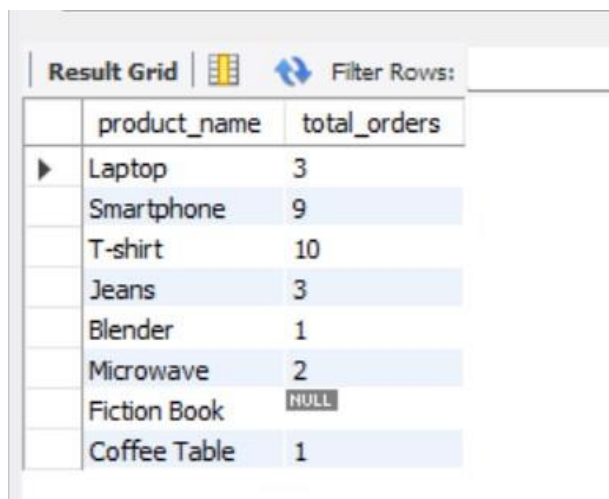


The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the output of the SQL query for the ProductStockValue view. It has four columns: product_name, quantity_in_stock, unit_price, and stock_value. The data is as follows:

	product_name	quantity_in_stock	unit_price	stock_value
▶	Laptop	50	999.99	49999.50
	Smartphone	100	599.99	59999.00
	T-shirt	200	19.99	3998.00
	Jeans	150	39.99	5998.50
	Blender	75	49.99	3749.25
	Microwave	30	99.99	2999.70
	Fiction Book	300	9.99	2997.00
	Coffee Table	20	199.99	3999.80

43. View for Total Orders by Product

```
CREATE VIEW ProductOrderSummary AS  
  
SELECT p.product_name, SUM(o.quantity_ordered) AS total_orders  
  
FROM Products p  
  
LEFT JOIN Orders o ON p.product_id = o.product_id  
  
GROUP BY p.product_name;
```



The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the output of the SQL query for the ProductOrderSummary view. It has two columns: product_name and total_orders. The data is as follows:

	product_name	total_orders
▶	Laptop	3
	Smartphone	9
	T-shirt	10
	Jeans	3
	Blender	1
	Microwave	2
	Fiction Book	NULL
	Coffee Table	1

44. Trigger to Update Stock After an Order is Placed

delimiter //

```
CREATE TRIGGER UpdateStockAfterOrder
```

```
AFTER INSERT ON Orders
```

```
FOR EACH ROW
```

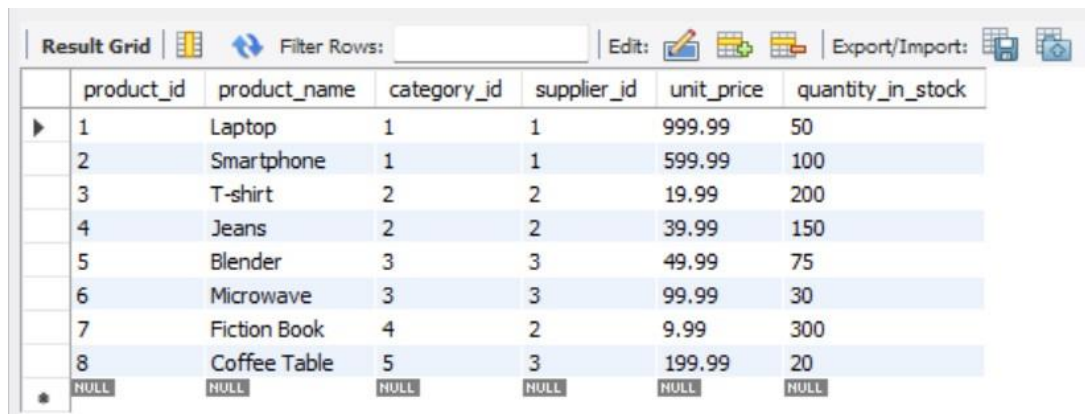
```
BEGIN
```

```
    UPDATE Products
```

```
    SET quantity_in_stock = quantity_in_stock - NEW.quantity_ordered
```

```
    WHERE product_id = NEW.product_id;
```

```
END;
```



The screenshot shows a database application interface with a 'Result Grid' tab. The grid displays a list of products with columns for product_id, product_name, category_id, supplier_id, unit_price, and quantity_in_stock. The data is as follows:

	product_id	product_name	category_id	supplier_id	unit_price	quantity_in_stock
▶	1	Laptop	1	1	999.99	50
	2	Smartphone	1	1	599.99	100
	3	T-shirt	2	2	19.99	200
	4	Jeans	2	2	39.99	150
	5	Blender	3	3	49.99	75
	6	Microwave	3	3	99.99	30
	7	Fiction Book	4	2	9.99	300
	8	Coffee Table	5	3	199.99	20
✱	NULL	NULL	NULL	NULL	NULL	NULL

45. Cursor to Display Low Stock Products

```
DROP PROCEDURE ShowLowStockProducts();

DELIMITER //

CREATE PROCEDURE LowStockProducts()

BEGIN

    DECLARE done INT DEFAULT FALSE;

    DECLARE prod_name VARCHAR(100);

    DECLARE stock_quantity INT;


    DECLARE low_stock_cursor CURSOR FOR

        SELECT product_name, quantity_in_stock

        FROM Products

        WHERE quantity_in_stock < 20;


    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;


    OPEN low_stock_cursor;


    read_loop: LOOP

        FETCH low_stock_cursor INTO prod_name, stock_quantity;

        IF done THEN

            LEAVE read_loop;

        END IF;

        SELECT CONCAT('Product: ', prod_name, ', Stock: ', stock_quantity) AS

LowStockProducts;

    END LOOP;


    CLOSE low_stock_cursor;

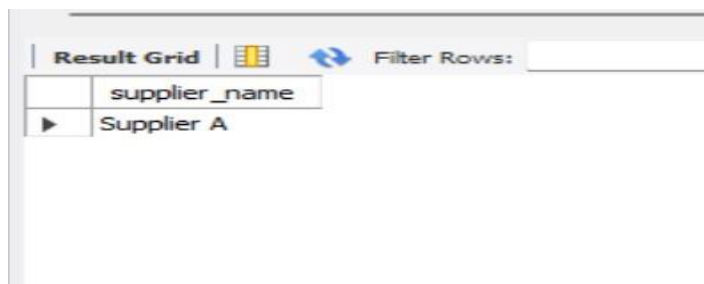
END //

DELIMITER ;

call LowStockProducts()
```


46. Get the Supplier of the Most Expensive Product

```
SELECT s.supplier_name  
FROM Suppliers s  
WHERE s.supplier_id = (  
    SELECT supplier_id  
    FROM Products  
    WHERE unit_price = (SELECT MAX(unit_price) FROM Products)  
);
```

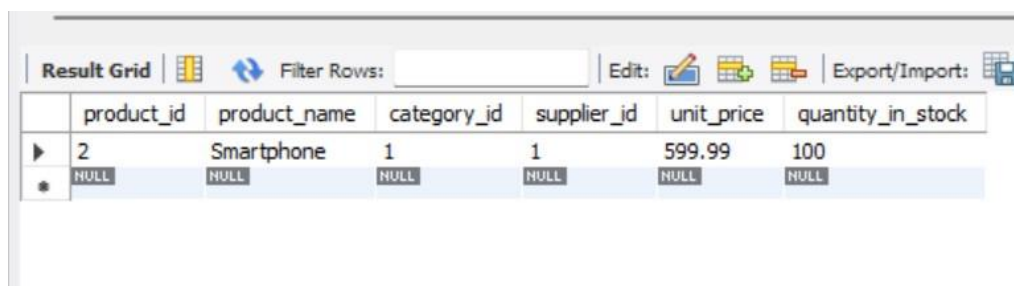


The screenshot shows a 'Result Grid' window with a 'Filter Rows' field. The grid contains one column labeled 'supplier_name' and one row with the value 'Supplier A'.

supplier_name
Supplier A

47. Get Products Ordered on a Specific Date

```
SELECT *  
FROM Products  
WHERE product_id IN (  
    SELECT product_id  
    FROM Orders  
    WHERE order_date = '2024-01-15'  
);
```



The screenshot shows a 'Result Grid' window with a 'Filter Rows' field and buttons for 'Edit', 'Export/Import', and 'Filter Rows'. The grid contains seven columns: 'product_id', 'product_name', 'category_id', 'supplier_id', 'unit_price', and 'quantity_in_stock'. The first row shows a product with 'product_id' 2, 'product_name' 'Smartphone', 'category_id' 1, 'supplier_id' 1, 'unit_price' 599.99, and 'quantity_in_stock' 100. The second row is a placeholder with all 'NULL' values.

product_id	product_name	category_id	supplier_id	unit_price	quantity_in_stock
2	Smartphone	1	1	599.99	100
NULL	NULL	NULL	NULL	NULL	NULL

48. Total Revenue Generated by Each Supplier

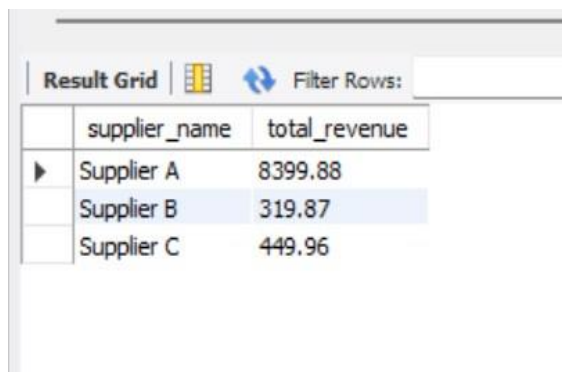
```
SELECT s.supplier_name, SUM(p.unit_price * o.quantity_ordered) AS  
total_revenue
```

```
FROM Suppliers s
```

```
JOIN Products p ON s.supplier_id = p.supplier_id
```

```
JOIN Orders o ON p.product_id = o.product_id
```

```
GROUP BY s.supplier_name;
```



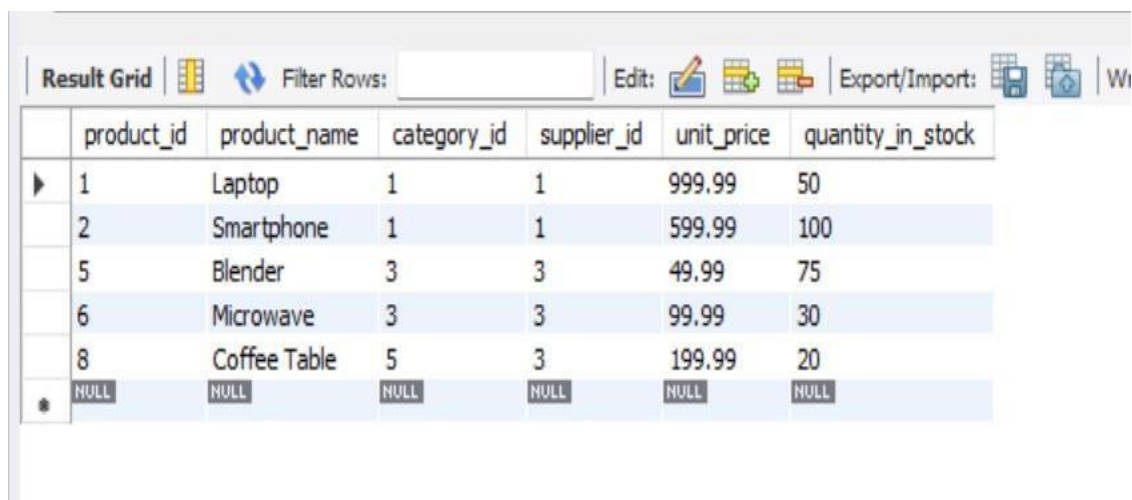
The screenshot shows a 'Result Grid' window with a toolbar containing icons for grid view, refresh, and a filter input field. The table has two columns: 'supplier_name' and 'total_revenue'. The data is as follows:

	supplier_name	total_revenue
▶	Supplier A	8399.88
	Supplier B	319.87
	Supplier C	449.96

49. Find Products Below Average Stock

```
SELECT * FROM Products
```

```
WHERE quantity_in_stock < (SELECT AVG(quantity_in_stock) FROM  
Products);
```



The screenshot shows a 'Result Grid' window with a toolbar containing icons for grid view, refresh, filter, edit, export/import, and window management. The table has seven columns: 'product_id', 'product_name', 'category_id', 'supplier_id', 'unit_price', and 'quantity_in_stock'. The data is as follows:

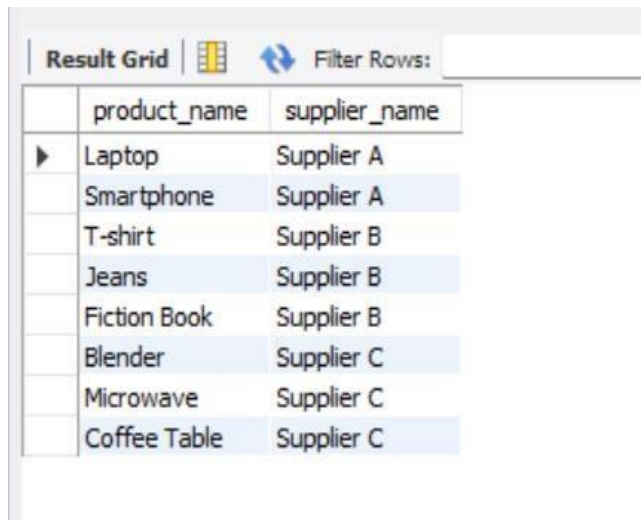
	product_id	product_name	category_id	supplier_id	unit_price	quantity_in_stock
▶	1	Laptop	1	1	999.99	50
	2	Smartphone	1	1	599.99	100
	5	Blender	3	3	49.99	75
	6	Microwave	3	3	99.99	30
	8	Coffee Table	5	3	199.99	20
✱	NULL	NULL	NULL	NULL	NULL	NULL

50. List products with their corresponding supplier names.

```
SELECT p.product_name, s.supplier_name
```

```
FROM Products p
```

```
JOIN Suppliers s ON p.supplier_id = s.supplier_id;
```



	product_name	supplier_name
▶	Laptop	Supplier A
	Smartphone	Supplier A
	T-shirt	Supplier B
	Jeans	Supplier B
	Fiction Book	Supplier B
	Blender	Supplier C
	Microwave	Supplier C
	Coffee Table	Supplier C

