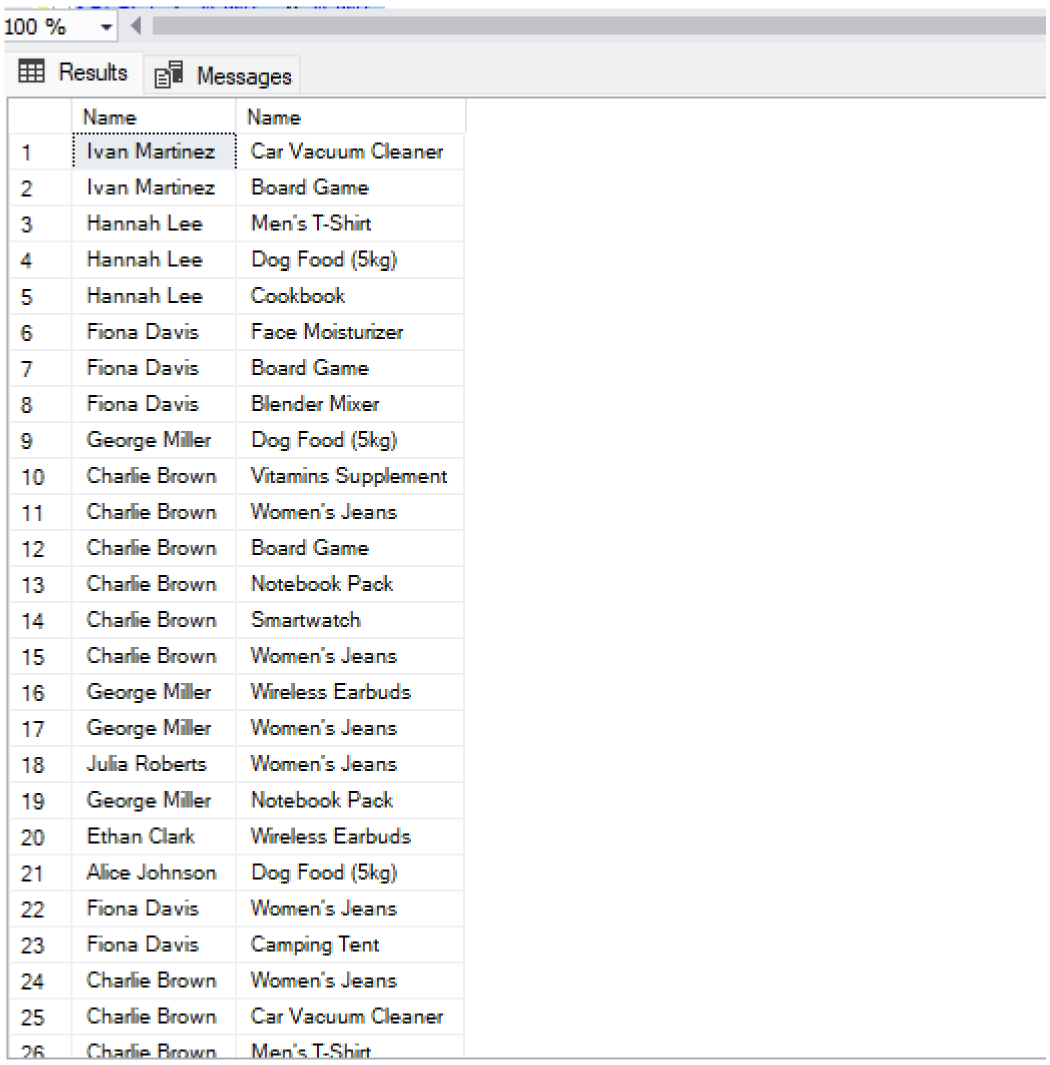
1. List all customer names along with the product names they purchased.

```
SELECT c.Name, p.Name
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
JOIN OrderDetails od ON o.OrderID = od.OrderID
JOIN Products p ON od.ProductID = p.ProductID;
```



100 %

Results Messages

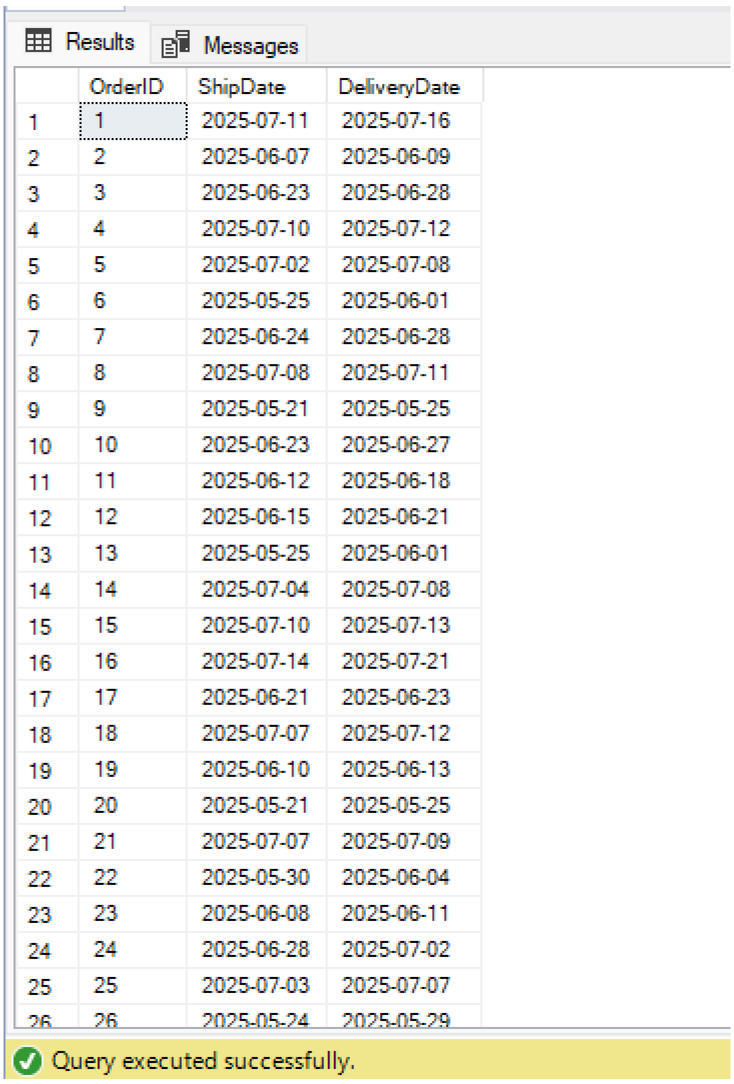
	Name	Name
1	Ivan Martinez	Car Vacuum Cleaner
2	Ivan Martinez	Board Game
3	Hannah Lee	Men's T-Shirt
4	Hannah Lee	Dog Food (5kg)
5	Hannah Lee	Cookbook
6	Fiona Davis	Face Moisturizer
7	Fiona Davis	Board Game
8	Fiona Davis	Blender Mixer
9	George Miller	Dog Food (5kg)
10	Charlie Brown	Vitamins Supplement
11	Charlie Brown	Women's Jeans
12	Charlie Brown	Board Game
13	Charlie Brown	Notebook Pack
14	Charlie Brown	Smartwatch
15	Charlie Brown	Women's Jeans
16	George Miller	Wireless Earbuds
17	George Miller	Women's Jeans
18	Julia Roberts	Women's Jeans
19	George Miller	Notebook Pack
20	Ethan Clark	Wireless Earbuds
21	Alice Johnson	Dog Food (5kg)
22	Fiona Davis	Women's Jeans
23	Fiona Davis	Camping Tent
24	Charlie Brown	Women's Jeans
25	Charlie Brown	Car Vacuum Cleaner
26	Charlie Brown	Men's T-Shirt

2. **Show order IDs with shipping status (date of delivery).**

SELECT o.OrderID, s.ShipDate, s.DeliveryDate

FROM Orders o

LEFT JOIN Shipping s ON o.OrderID = s.OrderID;



	OrderID	ShipDate	DeliveryDate
1	1	2025-07-11	2025-07-16
2	2	2025-06-07	2025-06-09
3	3	2025-06-23	2025-06-28
4	4	2025-07-10	2025-07-12
5	5	2025-07-02	2025-07-08
6	6	2025-05-25	2025-06-01
7	7	2025-06-24	2025-06-28
8	8	2025-07-08	2025-07-11
9	9	2025-05-21	2025-05-25
10	10	2025-06-23	2025-06-27
11	11	2025-06-12	2025-06-18
12	12	2025-06-15	2025-06-21
13	13	2025-05-25	2025-06-01
14	14	2025-07-04	2025-07-08
15	15	2025-07-10	2025-07-13
16	16	2025-07-14	2025-07-21
17	17	2025-06-21	2025-06-23
18	18	2025-07-07	2025-07-12
19	19	2025-06-10	2025-06-13
20	20	2025-05-21	2025-05-25
21	21	2025-07-07	2025-07-09
22	22	2025-05-30	2025-06-04
23	23	2025-06-08	2025-06-11
24	24	2025-06-28	2025-07-02
25	25	2025-07-03	2025-07-07
26	26	2025-05-24	2025-05-29

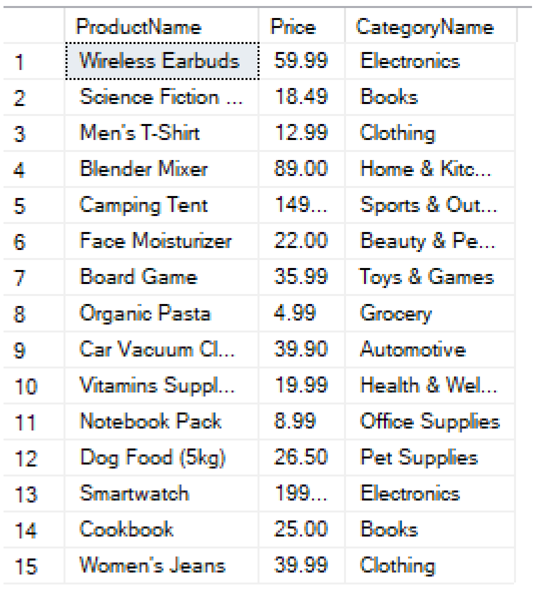
Query executed successfully.

3. Get product names, prices, and associated category names.

   SELECT p.Name AS ProductName, p.Price, c.CategoryName

   FROM Products p

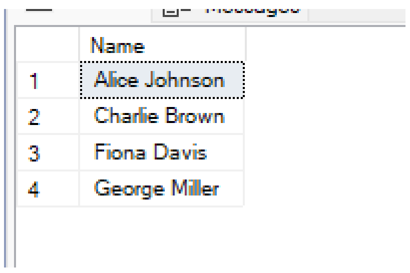
   JOIN Categories c ON p.CategoryID = c.CategoryID;



	ProductName	Price	CategoryName
1	Wireless Earbuds	59.99	Electronics
2	Science Fiction ...	18.49	Books
3	Men's T-Shirt	12.99	Clothing
4	Blender Mixer	89.00	Home & Kitc...
5	Camping Tent	149...	Sports & Out...
6	Face Moisturizer	22.00	Beauty & Pe...
7	Board Game	35.99	Toys & Games
8	Organic Pasta	4.99	Grocery
9	Car Vacuum Cl...	39.90	Automotive
10	Vitamins Suppl...	19.99	Health & Wel...
11	Notebook Pack	8.99	Office Supplies
12	Dog Food (5kg)	26.50	Pet Supplies
13	Smartwatch	199...	Electronics
14	Cookbook	25.00	Books
15	Women's Jeans	39.99	Clothing

4. Find all customers who bought a 'Smartwatch'.

```
SELECT DISTINCT cu.Name
FROM Customers cu
JOIN Orders o ON cu.CustomerID = o.CustomerID
JOIN OrderDetails od ON o.OrderID = od.OrderID
JOIN Products p ON od.ProductID = p.ProductID
WHERE p.Name = 'Smartwatch';
```



	Name
1	Alice Johnson
2	Charlie Brown
3	Fiona Davis
4	George Miller

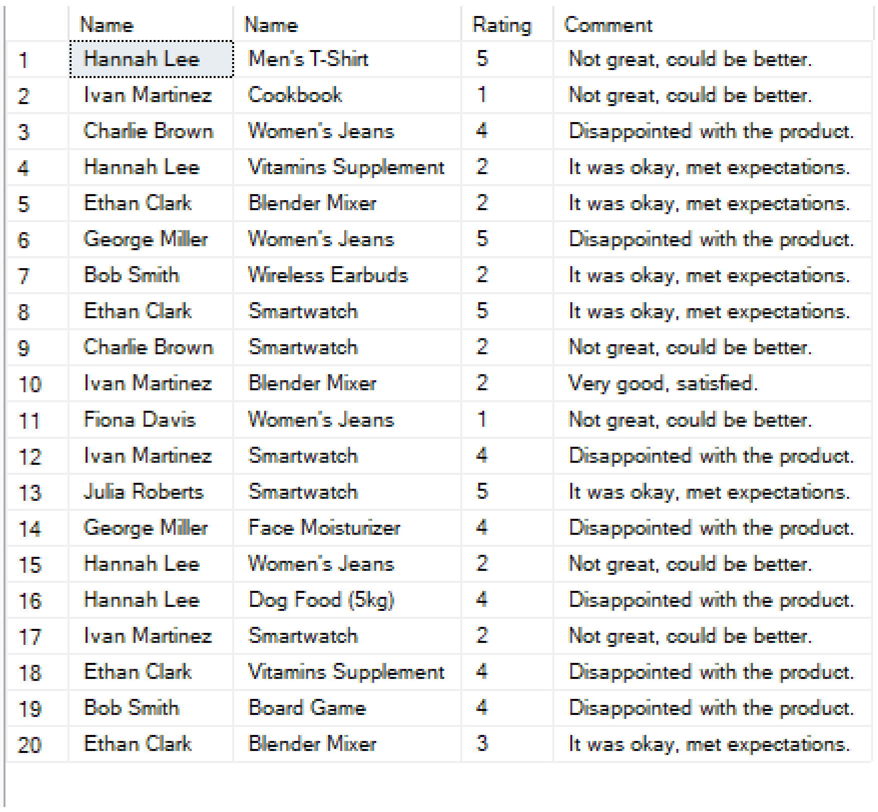
5. List all reviews with customer names and product names.

SELECT cu.Name, p.Name, r.Rating, r.Comment

FROM Reviews r

JOIN Customers cu ON r.CustomerID = cu.CustomerID

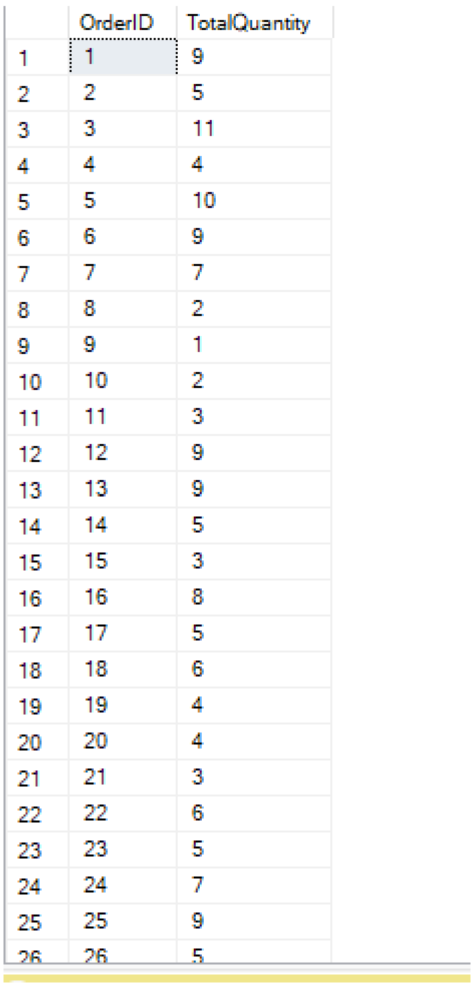
JOIN Products p ON r.ProductID = p.ProductID;



	Name	Name	Rating	Comment
1	Hannah Lee	Men's T-Shirt	5	Not great, could be better.
2	Ivan Martinez	Cookbook	1	Not great, could be better.
3	Charlie Brown	Women's Jeans	4	Disappointed with the product.
4	Hannah Lee	Vitamins Supplement	2	It was okay, met expectations.
5	Ethan Clark	Blender Mixer	2	It was okay, met expectations.
6	George Miller	Women's Jeans	5	Disappointed with the product.
7	Bob Smith	Wireless Earbuds	2	It was okay, met expectations.
8	Ethan Clark	Smartwatch	5	It was okay, met expectations.
9	Charlie Brown	Smartwatch	2	Not great, could be better.
10	Ivan Martinez	Blender Mixer	2	Very good, satisfied.
11	Fiona Davis	Women's Jeans	1	Not great, could be better.
12	Ivan Martinez	Smartwatch	4	Disappointed with the product.
13	Julia Roberts	Smartwatch	5	It was okay, met expectations.
14	George Miller	Face Moisturizer	4	Disappointed with the product.
15	Hannah Lee	Women's Jeans	2	Not great, could be better.
16	Hannah Lee	Dog Food (5kg)	4	Disappointed with the product.
17	Ivan Martinez	Smartwatch	2	Not great, could be better.
18	Ethan Clark	Vitamins Supplement	4	Disappointed with the product.
19	Bob Smith	Board Game	4	Disappointed with the product.
20	Ethan Clark	Blender Mixer	3	It was okay, met expectations.

6. Show all orders with their total quantity.

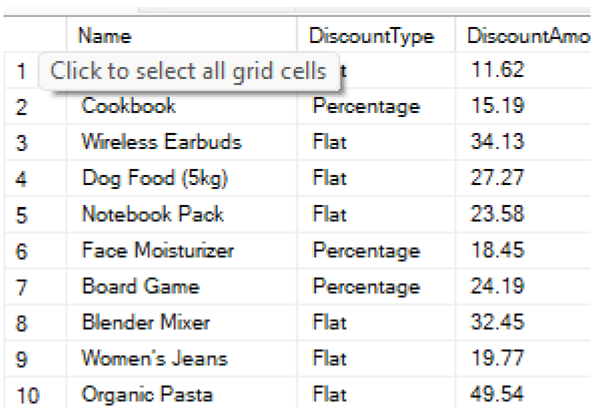
```sql
SELECT o.OrderID, SUM(od.Quantity) AS TotalQuantity
FROM Orders o
JOIN OrderDetails od ON o.OrderID = od.OrderID
GROUP BY o.OrderID;
```



	OrderID	TotalQuantity
1	1	9
2	2	5
3	3	11
4	4	4
5	5	10
6	6	9
7	7	7
8	8	2
9	9	1
10	10	2
11	11	3
12	12	9
13	13	9
14	14	5
15	15	3
16	16	8
17	17	5
18	18	6
19	19	4
20	20	4
21	21	3
22	22	6
23	23	5
24	24	7
25	25	9
26	26	5

7. List products and their discount types.

```
SELECT p.Name, d.DiscountType, d.DiscountAmount
FROM Products p
JOIN Discounts d ON p.ProductID = d.ProductID;
```



	Name	DiscountType	DiscountAmo
1	Click to select all grid cells	t	11.62
2	Cookbook	Percentage	15.19
3	Wireless Earbuds	Flat	34.13
4	Dog Food (5kg)	Flat	27.27
5	Notebook Pack	Flat	23.58
6	Face Moisturizer	Percentage	18.45
7	Board Game	Percentage	24.19
8	Blender Mixer	Flat	32.45
9	Women's Jeans	Flat	19.77
10	Organic Pasta	Flat	49.54

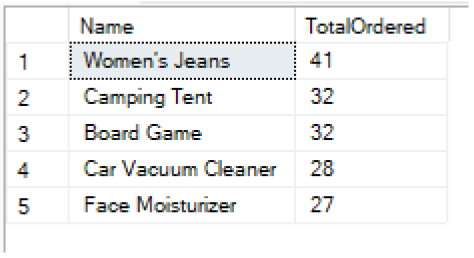
8. Show the top 5 most ordered products.

SELECT p.Name, SUM(od.Quantity) AS TotalOrdered

FROM OrderDetails od

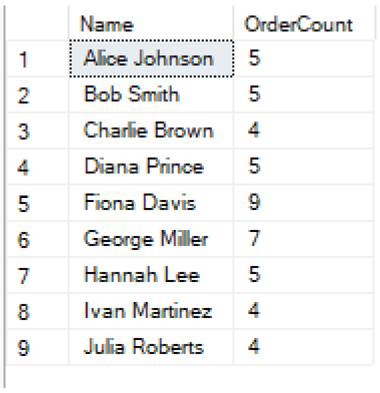
JOIN Products p ON od.ProductID = p.ProductID

GROUP BY p.Name
ORDER BY TotalOrdered DESC
OFFSET 0 ROWS FETCH NEXT 5 ROWS ONLY;



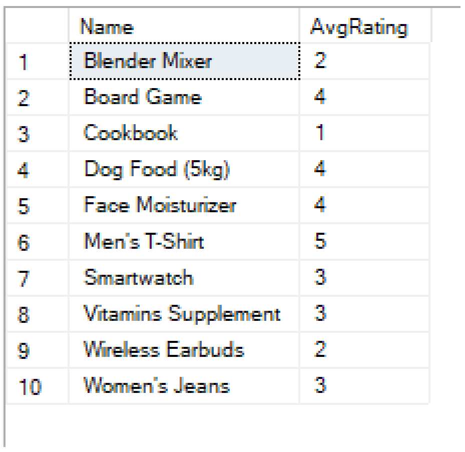
	Name	TotalOrdered
1	Women's Jeans	41
2	Camping Tent	32
3	Board Game	32
4	Car Vacuum Cleaner	28
5	Face Moisturizer	27

9. List customers who placed more than 2 orders.
SELECT c.Name, COUNT(o.OrderID) AS OrderCount
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
GROUP BY c.Name
HAVING COUNT(o.OrderID) > 2;



	Name	OrderCount
1	Alice Johnson	5
2	Bob Smith	5
3	Charlie Brown	4
4	Diana Prince	5
5	Fiona Davis	9
6	George Miller	7
7	Hannah Lee	5
8	Ivan Martinez	4
9	Julia Roberts	4

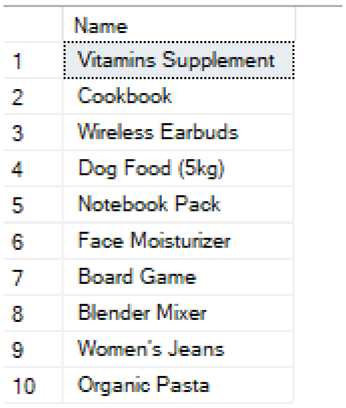
10. **Find average rating for each product.**
```
SELECT p.Name, AVG(r.Rating) AS AvgRating
FROM Products p
JOIN Reviews r ON p.ProductID = r.ProductID
GROUP BY p.Name;
```



	Name	AvgRating
1	Blender Mixer	2
2	Board Game	4
3	Cookbook	1
4	Dog Food (5kg)	4
5	Face Moisturizer	4
6	Men's T-Shirt	5
7	Smartwatch	3
8	Vitamins Supplement	3
9	Wireless Earbuds	2
10	Women's Jeans	3

11. List product names with active discounts today.

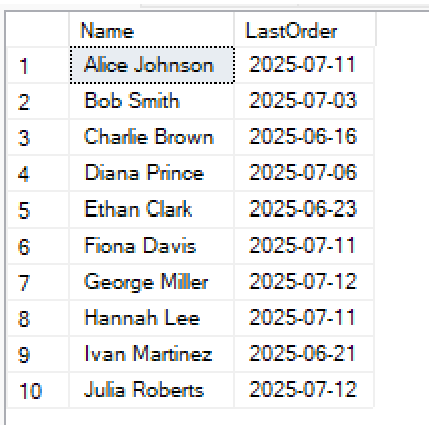
```
SELECT p.Name
FROM Products p
JOIN Discounts d ON p.ProductID = d.ProductID
WHERE GETDATE() BETWEEN d.StartDate AND d.EndDate;
```



	Name
1	Vitamins Supplement
2	Cookbook
3	Wireless Earbuds
4	Dog Food (5kg)
5	Notebook Pack
6	Face Moisturizer
7	Board Game
8	Blender Mixer
9	Women's Jeans
10	Organic Pasta

12. Get each customer’s most recent order.

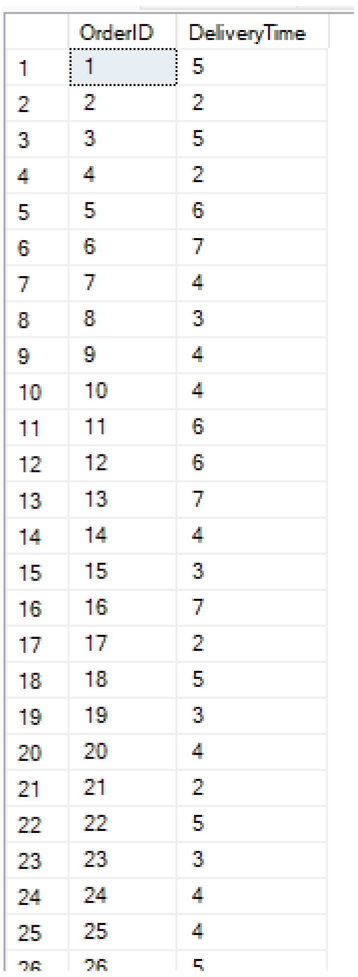
```
SELECT c.Name, MAX(o.OrderDate) AS LastOrder
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
GROUP BY c.Name;
```



	Name	LastOrder
1	Alice Johnson	2025-07-11
2	Bob Smith	2025-07-03
3	Charlie Brown	2025-06-16
4	Diana Prince	2025-07-06
5	Ethan Clark	2025-06-23
6	Fiona Davis	2025-07-11
7	George Miller	2025-07-12
8	Hannah Lee	2025-07-11
9	Ivan Martinez	2025-06-21
10	Julia Roberts	2025-07-12

13. List orders along with the shipping duration in days

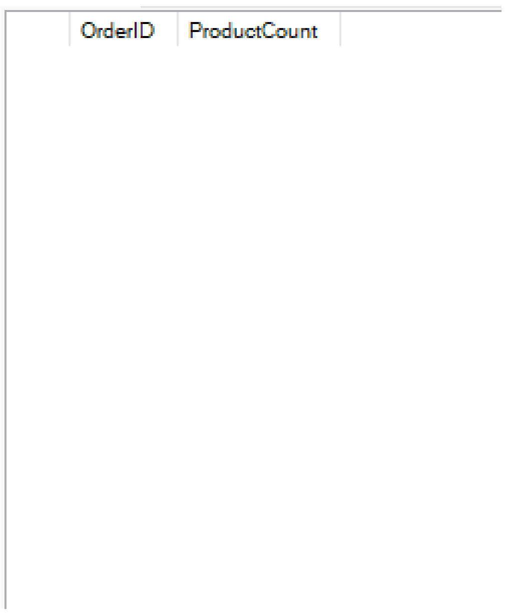
```
SELECT o.OrderID, DATEDIFF(DAY, s.ShipDate, s.DeliveryDate) AS DeliveryTime
FROM Orders o
JOIN Shipping s ON o.OrderID = s.OrderID;
```



	OrderID	DeliveryTime
1	1	5
2	2	2
3	3	5
4	4	2
5	5	6
6	6	7
7	7	4
8	8	3
9	9	4
10	10	4
11	11	6
12	12	6
13	13	7
14	14	4
15	15	3
16	16	7
17	17	2
18	18	5
19	19	3
20	20	4
21	21	2
22	22	5
23	23	3
24	24	4
25	25	4
26	26	5

14. Show orders that include more than 3 products

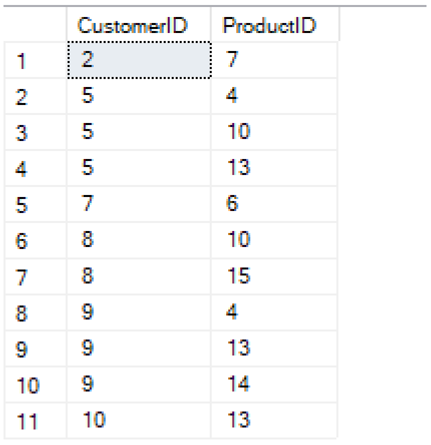
```
SELECT o.OrderID, COUNT(od.ProductID) AS ProductCount
FROM Orders o
JOIN OrderDetails od ON o.OrderID = od.OrderID
GROUP BY o.OrderID
HAVING COUNT(od.ProductID) > 3;
```



OrderID	ProductCount

15. Find customers who reviewed products they didn't purchase.

```
SELECT DISTINCT r.CustomerID, r.ProductID
FROM Reviews r
WHERE NOT EXISTS (
  SELECT 1
  FROM Orders o
  JOIN OrderDetails od ON o.OrderID = od.OrderID
  WHERE o.CustomerID = r.CustomerID AND od.ProductID = r.ProductID
);
```

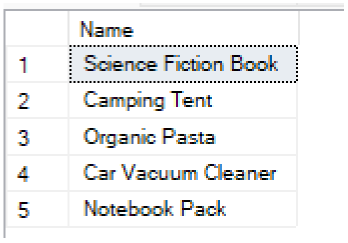


	CustomerID	ProductID
1	2	7
2	5	4
3	5	10
4	5	13
5	7	6
6	8	10
7	8	15
8	9	4
9	9	13
10	9	14
11	10	13

16. List products never reviewed.

```
SELECT p.Name
FROM Products p
LEFT JOIN Reviews r ON p.ProductID = r.ProductID
```

WHERE r.ProductID IS NULL;



	Name
1	Science Fiction Book
2	Camping Tent
3	Organic Pasta
4	Car Vacuum Cleaner
5	Notebook Pack

17. Get total sales value per category

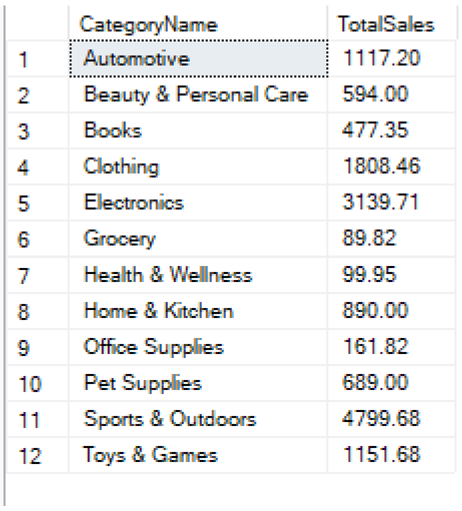
    SELECT c.CategoryName, SUM(p.Price * od.Quantity) AS TotalSales

    FROM Categories c

    JOIN Products p ON c.CategoryID = p.CategoryID

    JOIN OrderDetails od ON p.ProductID = od.ProductID

    GROUP BY c.CategoryName;



	CategoryName	TotalSales
1	Automotive	1117.20
2	Beauty & Personal Care	594.00
3	Books	477.35
4	Clothing	1808.46
5	Electronics	3139.71
6	Grocery	89.82
7	Health & Wellness	99.95
8	Home & Kitchen	890.00
9	Office Supplies	161.82
10	Pet Supplies	689.00
11	Sports & Outdoors	4799.68
12	Toys & Games	1151.68

18. Find customers who ordered from more than 3 different categories

    SELECT c.Name

    FROM Customers c

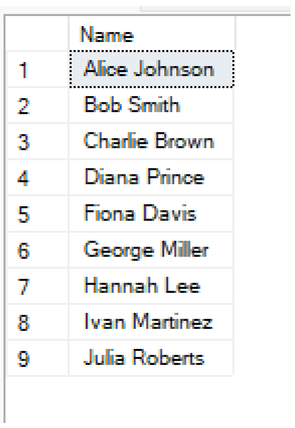
    JOIN Orders o ON c.CustomerID = o.CustomerID

    JOIN OrderDetails od ON o.OrderID = od.OrderID

    JOIN Products p ON od.ProductID = p.ProductID

    GROUP BY c.Name

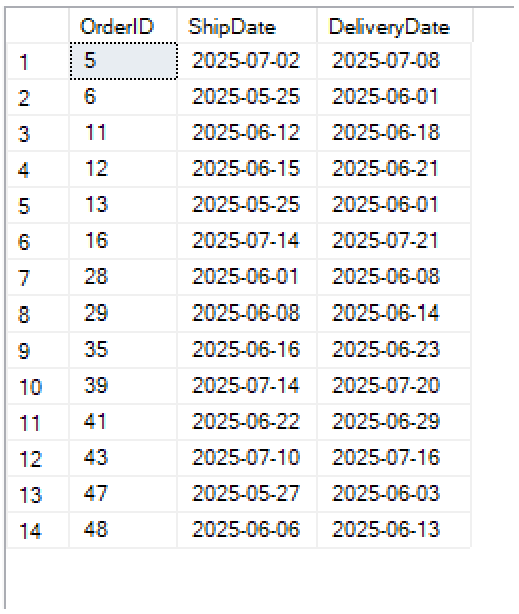
    HAVING COUNT(DISTINCT p.CategoryID) > 3;



	Name
1	Alice Johnson
2	Bob Smith
3	Charlie Brown
4	Diana Prince
5	Fiona Davis
6	George Miller
7	Hannah Lee
8	Ivan Martinez
9	Julia Roberts

19. List all orders that were delivered late (after 5 days of ship date).

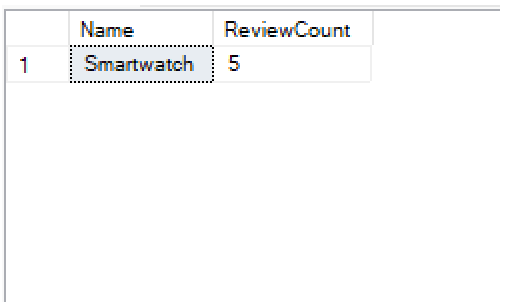
```
SELECT o.OrderID, s.ShipDate, s.DeliveryDate
FROM Orders o
JOIN Shipping s ON o.OrderID = s.OrderID
WHERE DATEDIFF(DAY, s.ShipDate, s.DeliveryDate) > 5;
```



	OrderID	ShipDate	DeliveryDate
1	5	2025-07-02	2025-07-08
2	6	2025-05-25	2025-06-01
3	11	2025-06-12	2025-06-18
4	12	2025-06-15	2025-06-21
5	13	2025-05-25	2025-06-01
6	16	2025-07-14	2025-07-21
7	28	2025-06-01	2025-06-08
8	29	2025-06-08	2025-06-14
9	35	2025-06-16	2025-06-23
10	39	2025-07-14	2025-07-20
11	41	2025-06-22	2025-06-29
12	43	2025-07-10	2025-07-16
13	47	2025-05-27	2025-06-03
14	48	2025-06-06	2025-06-13

20. Show the most reviewed product.

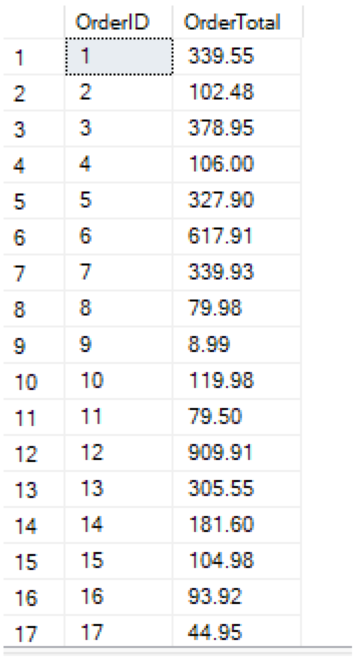
```
SELECT TOP 1 p.Name, COUNT(r.ReviewID) AS ReviewCount
FROM Products p
JOIN Reviews r ON p.ProductID = r.ProductID
GROUP BY p.Name
ORDER BY ReviewCount DESC; SELECT TOP 1 p.Name, COUNT(r.ReviewID) AS
ReviewCount
FROM Products p
JOIN Reviews r ON p.ProductID = r.ProductID
GROUP BY p.Name
ORDER BY ReviewCount DESC;
```



	Name	ReviewCount
1	Smartwatch	5

21. Get orders with total order value.

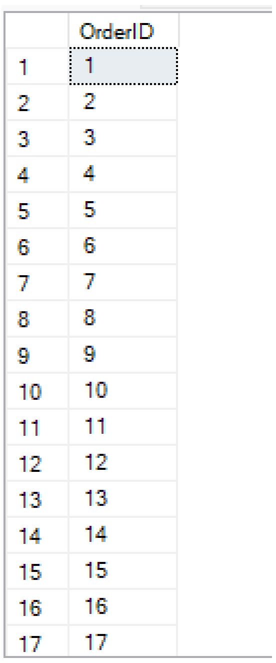
```
SELECT o.OrderID, SUM(p.Price * od.Quantity) AS OrderTotal
FROM Orders o
JOIN OrderDetails od ON o.OrderID = od.OrderID
JOIN Products p ON od.ProductID = p.ProductID
GROUP BY o.OrderID;
```



	OrderID	OrderTotal
1	1	339.55
2	2	102.48
3	3	378.95
4	4	106.00
5	5	327.90
6	6	617.91
7	7	339.93
8	8	79.98
9	9	8.99
10	10	119.98
11	11	79.50
12	12	909.91
13	13	305.55
14	14	181.60
15	15	104.98
16	16	93.92
17	17	44.95

22. List orders with products having discounts.

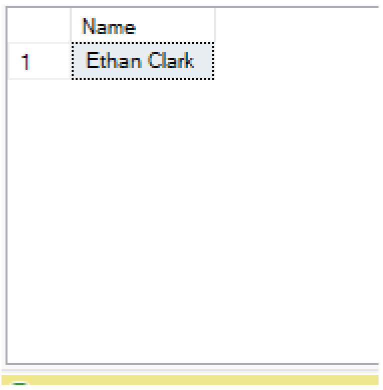
```
SELECT DISTINCT o.OrderID
FROM Orders o
JOIN OrderDetails od ON o.OrderID = od.OrderID
JOIN Discounts d ON od.ProductID = d.ProductID;
```



	OrderID
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17

23. Find customers who purchased discounted products only.

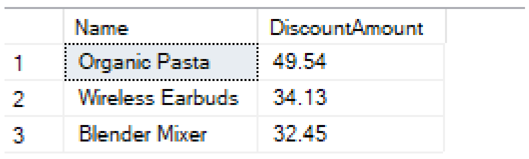
```
SELECT DISTINCT c.Name
FROM Customers c
WHERE NOT EXISTS (
  SELECT 1
  FROM Orders o
  JOIN OrderDetails od ON o.OrderID = od.OrderID
  JOIN Products p ON od.ProductID = p.ProductID
  LEFT JOIN Discounts d ON p.ProductID = d.ProductID
  WHERE o.CustomerID = c.CustomerID AND d.ProductID IS NULL
);
```



	Name
1	Ethan Clark

24. Show top 3 highest discount amounts and their product names.

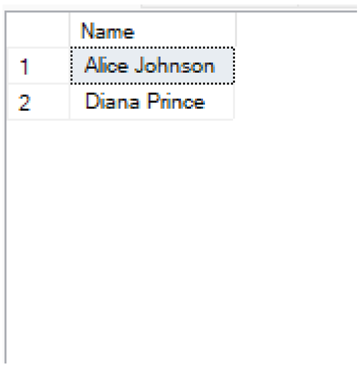
```
SELECT TOP 3 p.Name, d.DiscountAmount
FROM Discounts d
JOIN Products p ON d.ProductID = p.ProductID
ORDER BY d.DiscountAmount DESC;
```



	Name	DiscountAmount
1	Organic Pasta	49.54
2	Wireless Earbuds	34.13
3	Blender Mixer	32.45

25. Show customers who ordered but never reviewed any product.

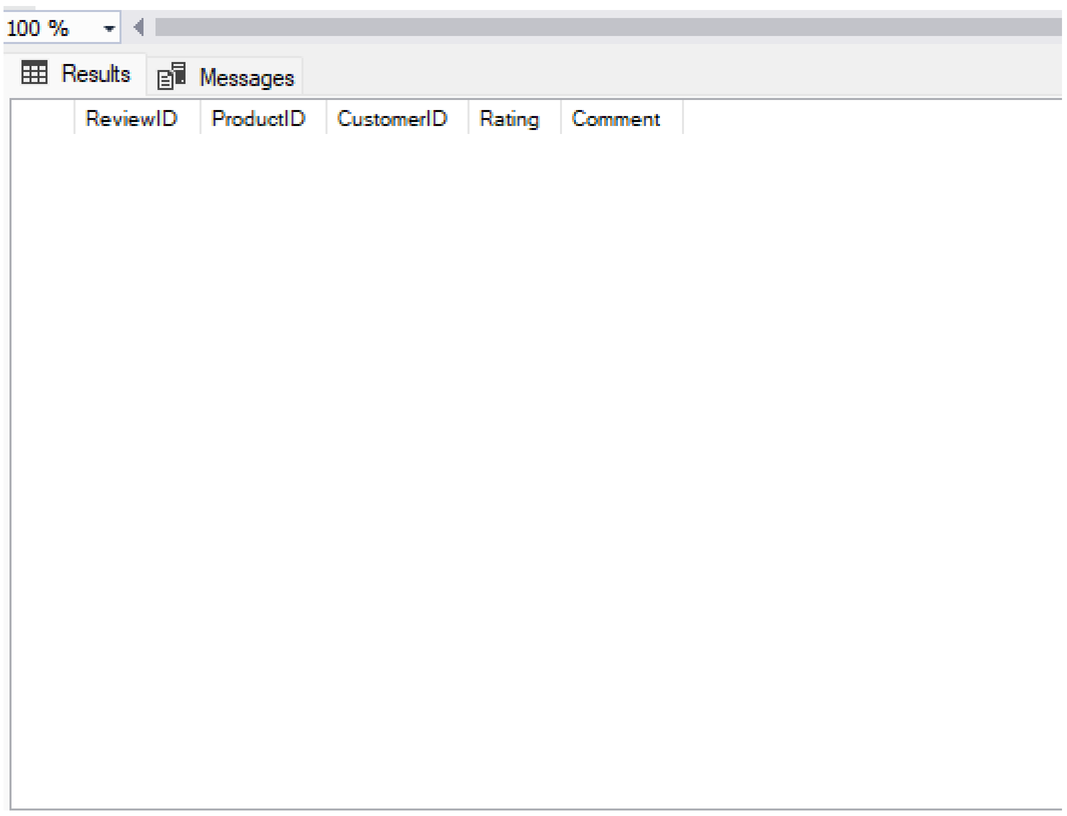
```
SELECT DISTINCT c.Name

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

WHERE c.CustomerID NOT IN (

  SELECT DISTINCT CustomerID FROM Reviews

);
```



	Name
1	Alice Johnson
2	Diana Prince

26. Check if all ratings fall between 1 and 5.

```
SELECT * FROM Reviews WHERE Rating < 1 OR Rating > 5;
```



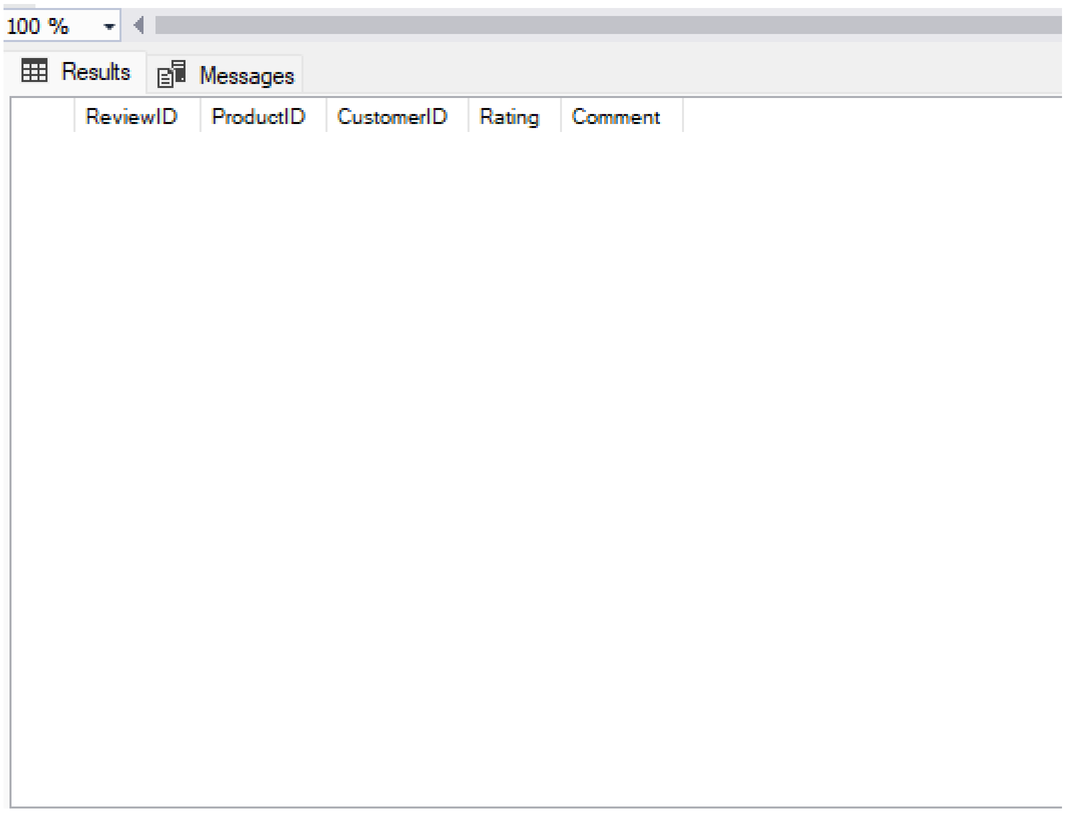
100 %

Results | Messages

	ReviewID	ProductID	CustomerID	Rating	Comment

27. List products with stock quantity < 0 (data anomaly check).

SELECT * FROM Products WHERE StockQuantity < 0;



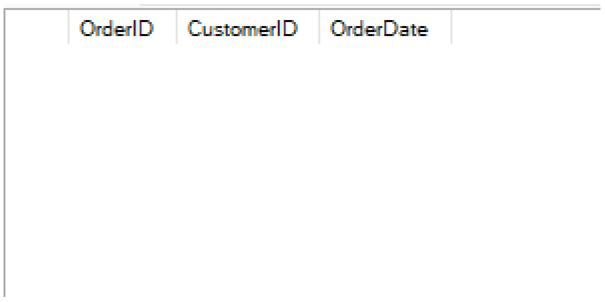
100 %

Results | Messages

	ReviewID	ProductID	CustomerID	Rating	Comment

28. Identify any Orders not in Shipping (violates uniqueness).

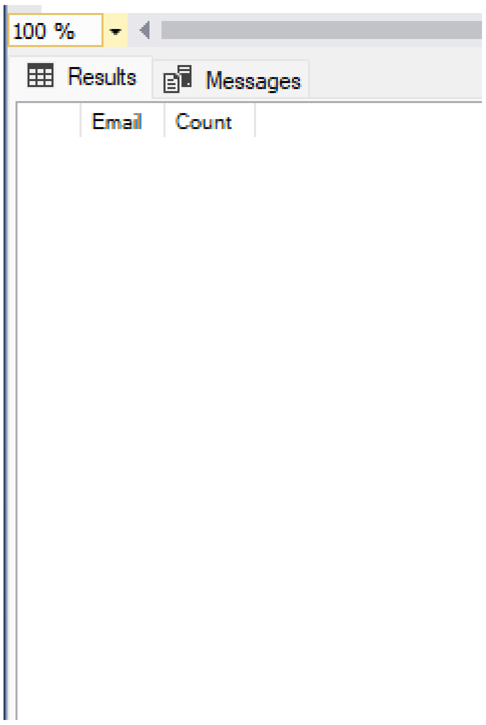
```sql
SELECT * FROM Orders

WHERE OrderID NOT IN (SELECT OrderID FROM Shipping);
```



```
OrderID   CustomerID   OrderDate
```

29. List duplicate emails in Customers (violates UNIQUE).

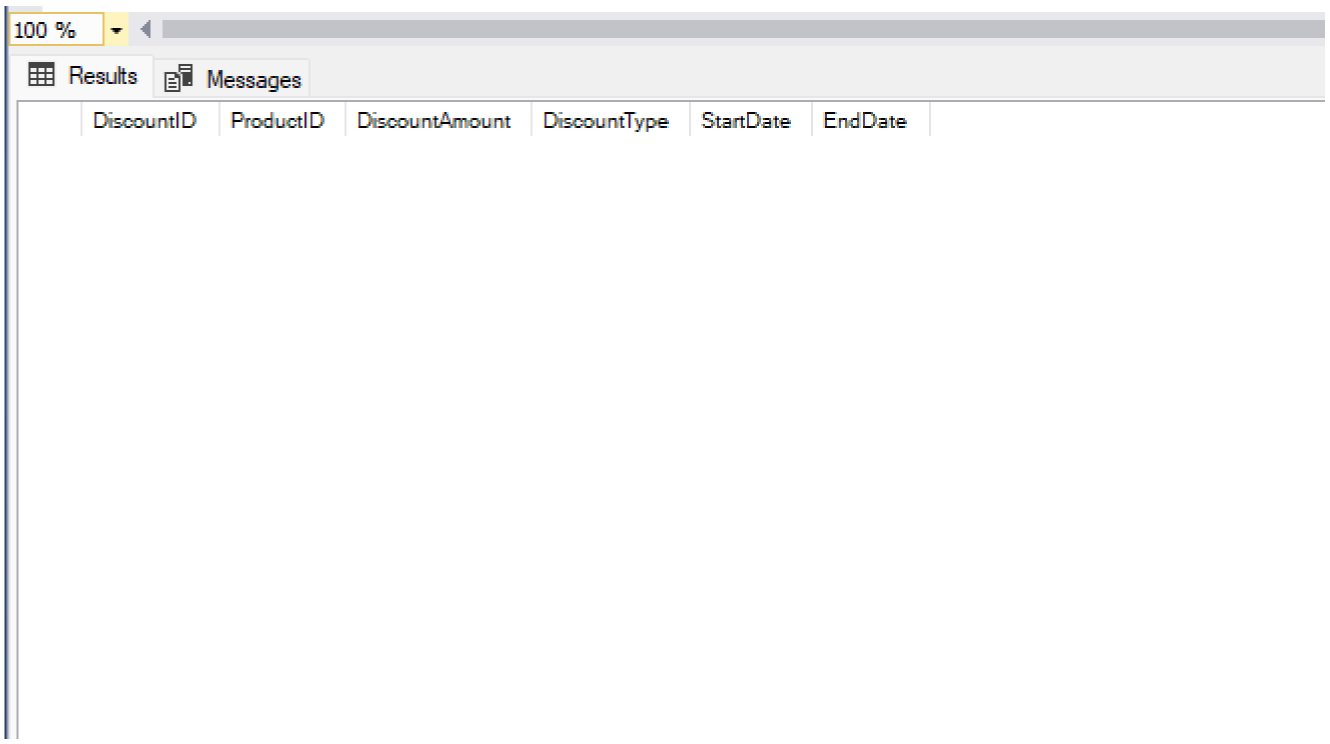
```sql
SELECT Email, COUNT(*) AS Count

FROM Customers

GROUP BY Email

HAVING COUNT(*) > 1;
```



```
Email   Count
```

30. Find discounts with EndDate earlier than StartDate.

```sql
SELECT * FROM Discounts

WHERE EndDate < StartDate;
```


100 %

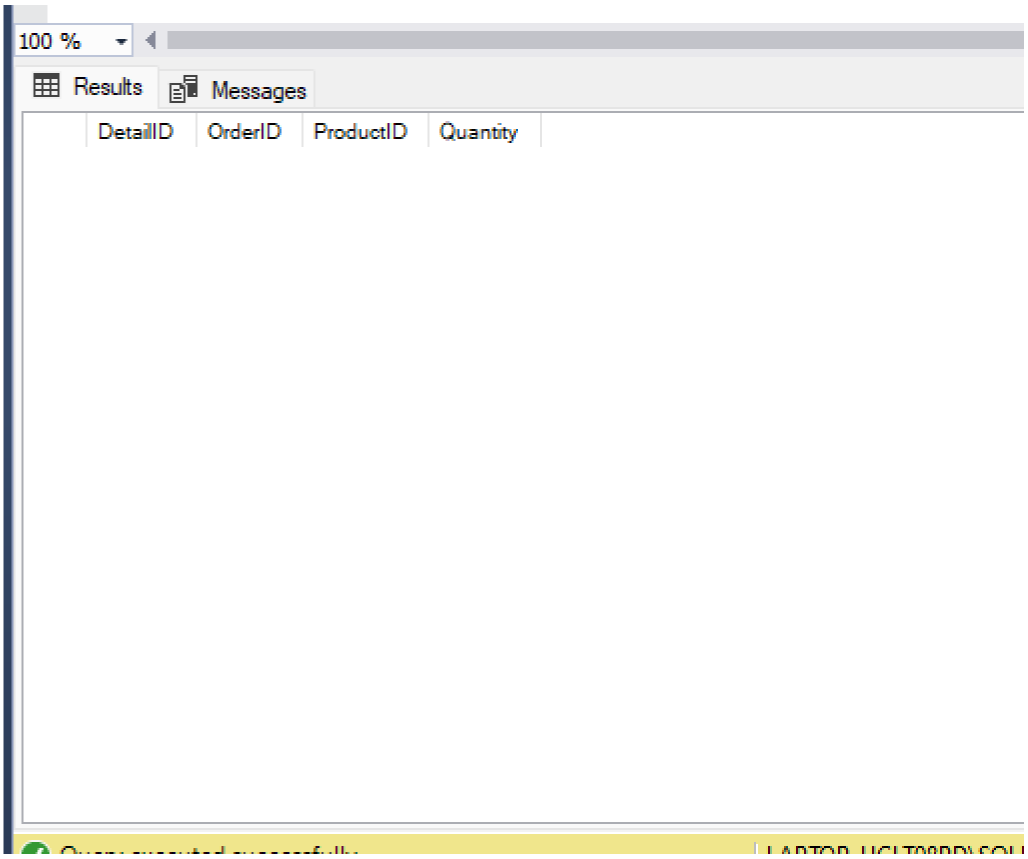
Results | Messages

DiscountID	ProductID	DiscountAmount	DiscountType	StartDate	EndDate

31. Detect foreign key orphan in OrderDetails (Product doesn’t exist).

SELECT * FROM OrderDetails

WHERE ProductID NOT IN (SELECT ProductID FROM Products);

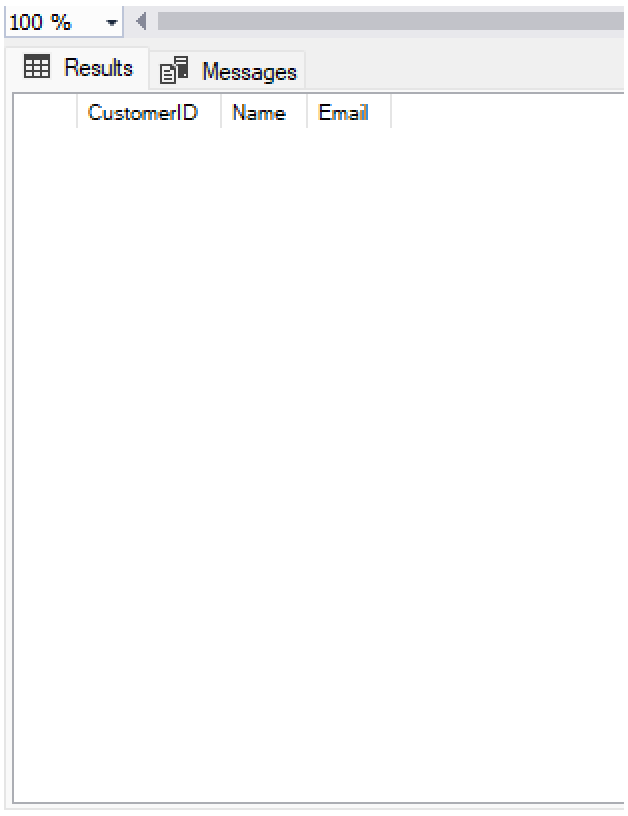


```
DetailID  OrderID  ProductID  Quantity
```

Query executed successfully.

32. Check for NULLs in NOT NULL columns in Customers.

```
SELECT * FROM Customers

WHERE Name IS NULL OR Email IS NULL;
```



```
CustomerID   Name   Email
```

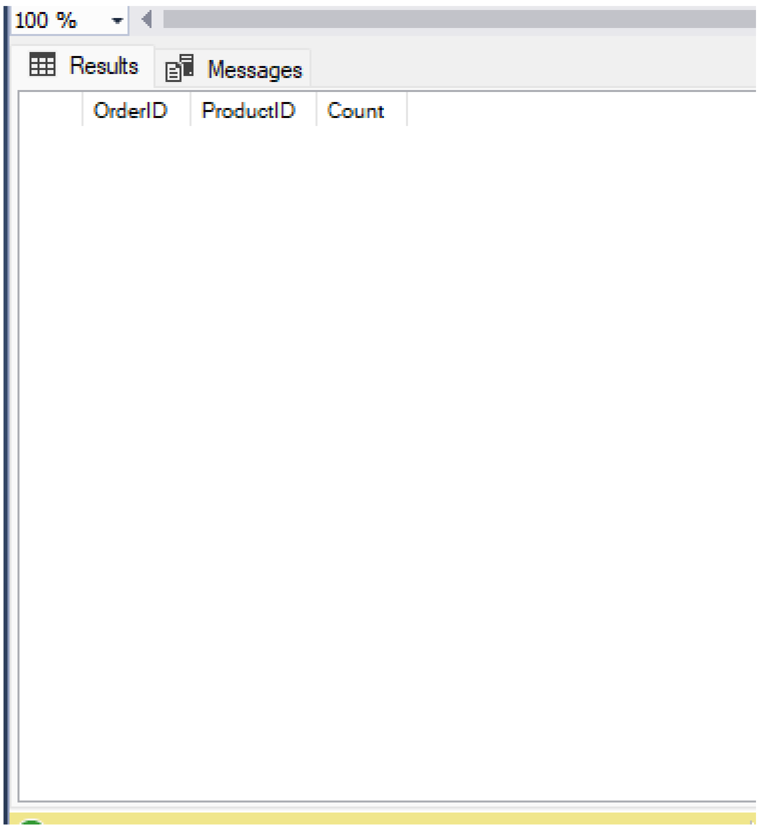
33. **Ensure no duplicate OrderDetails per order-product.**

SELECT OrderID, ProductID, COUNT(*) AS Count

FROM OrderDetails

GROUP BY OrderID, ProductID

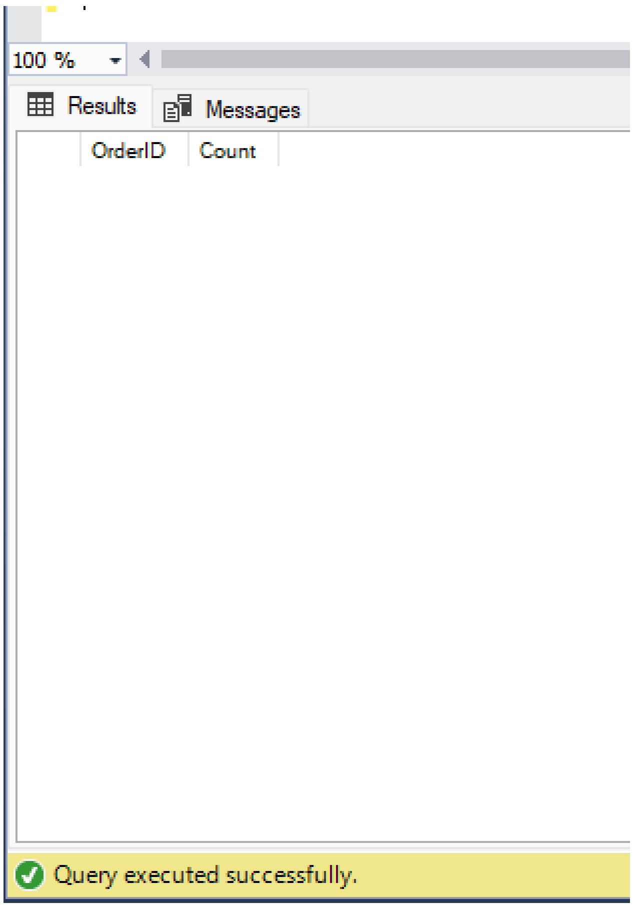
HAVING COUNT(*) > 1;



```
OrderID   ProductID   Count
```

34. Validate unique Shipping per OrderID.

```
SELECT OrderID, COUNT(*) AS Count

FROM Shipping

GROUP BY OrderID

HAVING COUNT(*) > 1;
```

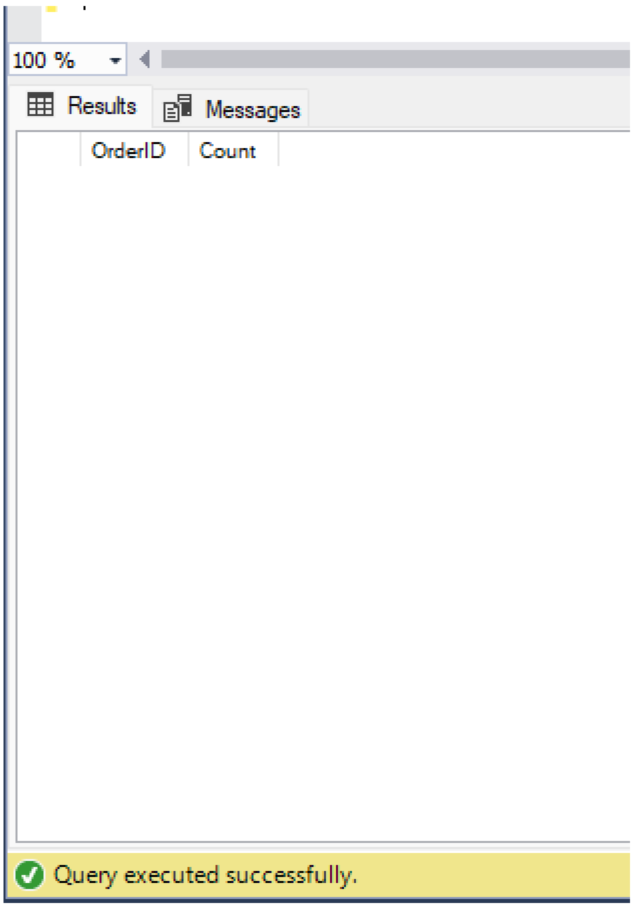


OrderID	Count

Query executed successfully.

35. Check if any Review has NULL for Rating.

SELECT * FROM Reviews WHERE Rating IS NULL;



```
100 %
Results   Messages
OrderID   Count
Query executed successfully.
```

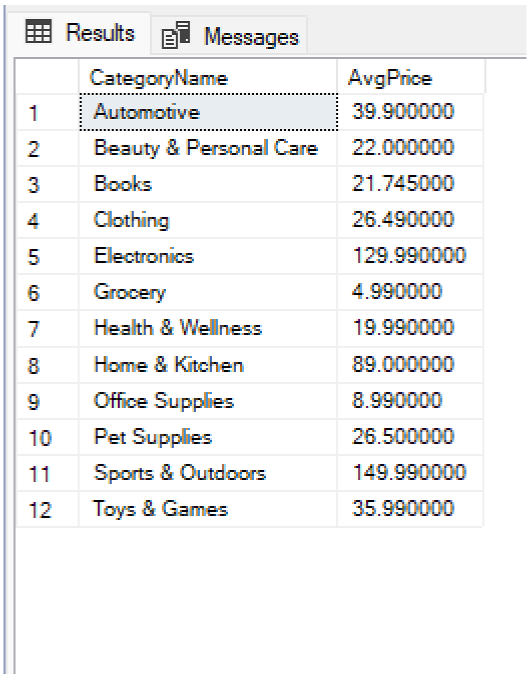
36. Find average price of products in each category.

SELECT c.CategoryName, AVG(p.Price) AS AvgPrice

FROM Categories c

JOIN Products p ON c.CategoryID = p.CategoryID

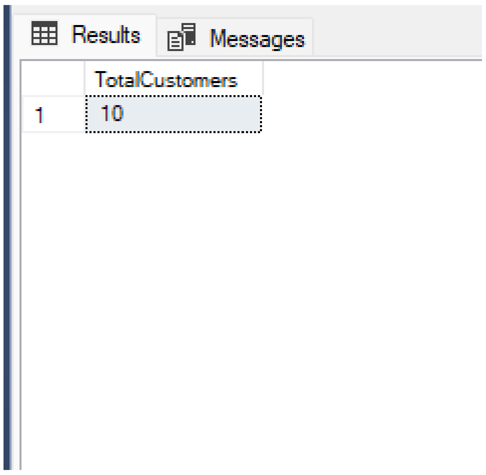
GROUP BY c.CategoryName;



	CategoryName	AvgPrice
1	Automotive	39.900000
2	Beauty & Personal Care	22.000000
3	Books	21.745000
4	Clothing	26.490000
5	Electronics	129.990000
6	Grocery	4.990000
7	Health & Wellness	19.990000
8	Home & Kitchen	89.000000
9	Office Supplies	8.990000
10	Pet Supplies	26.500000
11	Sports & Outdoors	149.990000
12	Toys & Games	35.990000

37. Count total number of customers.

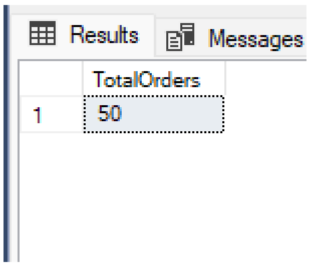
SELECT COUNT(*) AS TotalCustomers FROM Customers;



	TotalCustomers
1	10

38. Calculate total number of orders.

SELECT COUNT(*) AS TotalOrders FROM Orders;

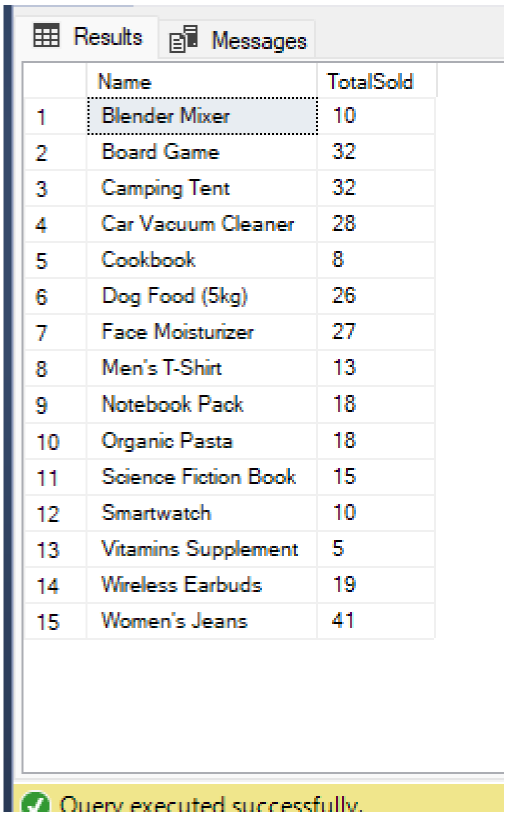


	TotalOrders
1	50

39. Total quantity of each product sold.

SELECT p.Name, SUM(od.Quantity) AS TotalSold

```
FROM Products p

JOIN OrderDetails od ON p.ProductID = od.ProductID

GROUP BY p.Name;
```



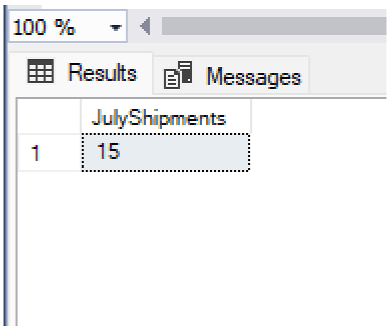
Results Messages

	Name	TotalSold
1	Blender Mixer	10
2	Board Game	32
3	Camping Tent	32
4	Car Vacuum Cleaner	28
5	Cookbook	8
6	Dog Food (5kg)	26
7	Face Moisturizer	27
8	Men's T-Shirt	13
9	Notebook Pack	18
10	Organic Pasta	18
11	Science Fiction Book	15
12	Smartwatch	10
13	Vitamins Supplement	5
14	Wireless Earbuds	19
15	Women's Jeans	41

Query executed successfully.

40. Number of orders shipped in July 2025.

```
SELECT COUNT(*) AS JulyShipments

FROM Shipping

WHERE MONTH(ShipDate) = 7 AND YEAR(ShipDate) = 2025;
```



100 % Results Messages

	JulyShipments
1	15

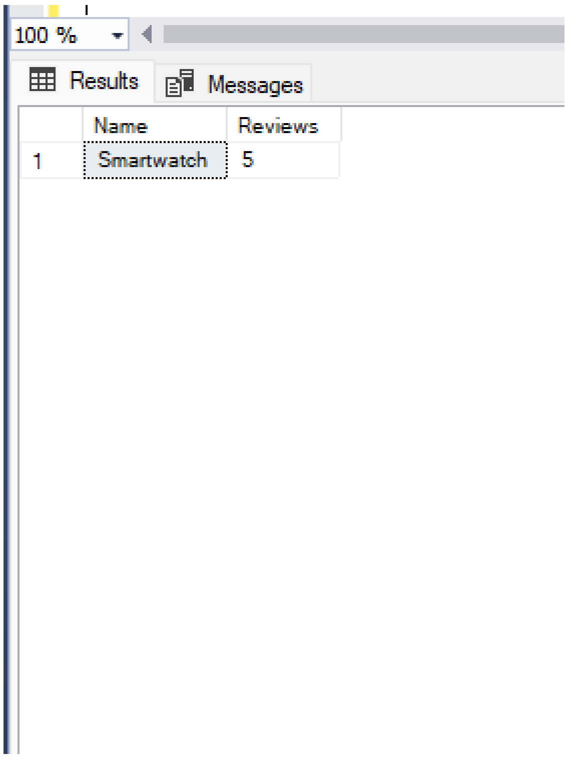
41. Find most common product in reviews.

```
SELECT TOP 1 p.Name, COUNT(*) AS Reviews

FROM Reviews r
```

JOIN Products p ON r.ProductID = p.ProductID

GROUP BY p.Name

ORDER BY Reviews DESC;



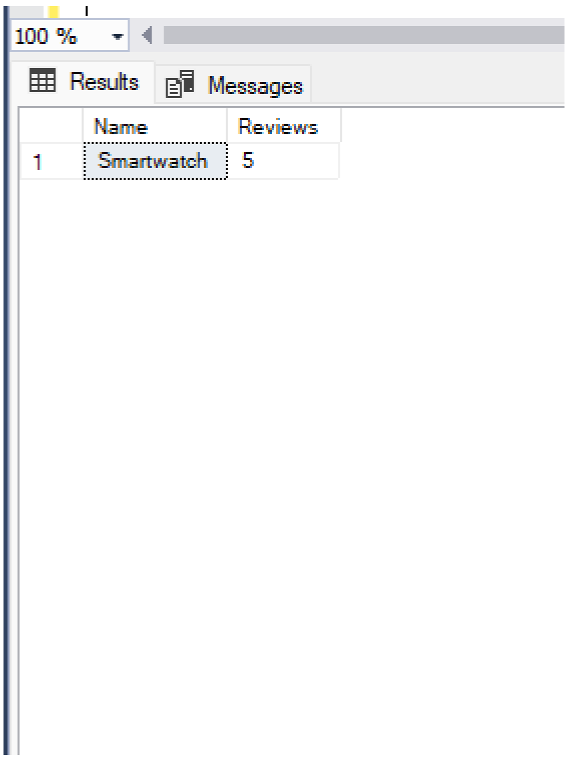
```
   Name        Reviews
1  Smartwatch  5
```

42. Display product with max stock available.

SELECT TOP 1 Name, StockQuantity

FROM Products

ORDER BY StockQuantity DESC;



100 %

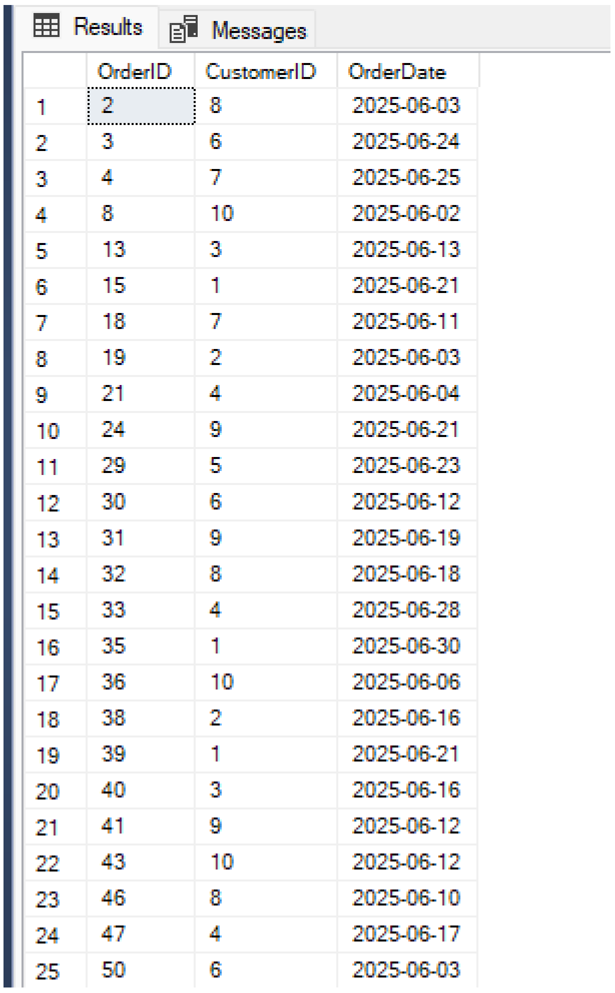
Results | Messages

```
   Name        Reviews
1  Smartwatch  5
```

43. Find orders placed in June 2025.

SELECT * FROM Orders

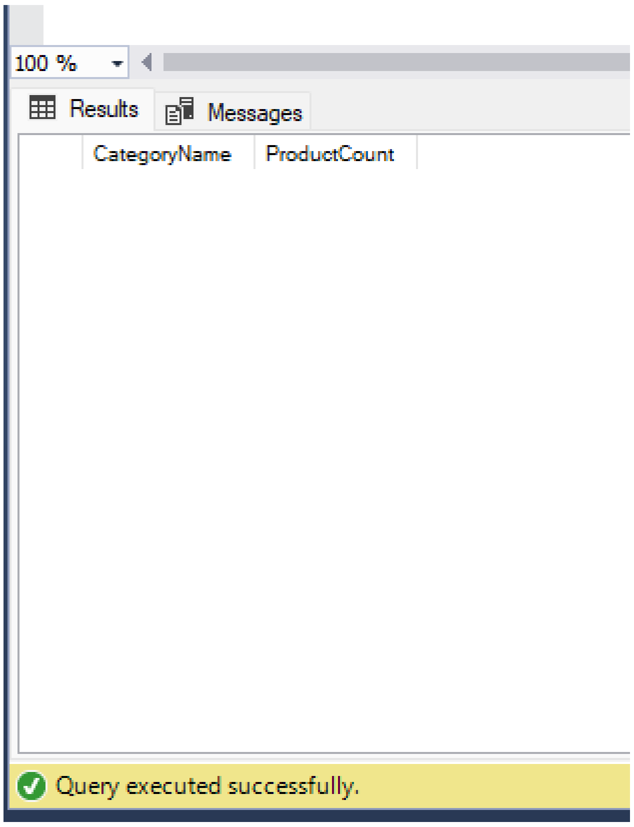
WHERE MONTH(OrderDate) = 6 AND YEAR(OrderDate) = 2025;


Results Messages

	OrderID	CustomerID	OrderDate
1	2	8	2025-06-03
2	3	6	2025-06-24
3	4	7	2025-06-25
4	8	10	2025-06-02
5	13	3	2025-06-13
6	15	1	2025-06-21
7	18	7	2025-06-11
8	19	2	2025-06-03
9	21	4	2025-06-04
10	24	9	2025-06-21
11	29	5	2025-06-23
12	30	6	2025-06-12
13	31	9	2025-06-19
14	32	8	2025-06-18
15	33	4	2025-06-28
16	35	1	2025-06-30
17	36	10	2025-06-06
18	38	2	2025-06-16
19	39	1	2025-06-21
20	40	3	2025-06-16
21	41	9	2025-06-12
22	43	10	2025-06-12
23	46	8	2025-06-10
24	47	4	2025-06-17
25	50	6	2025-06-03

44. List categories that have more than 2 products.

```
SELECT c.CategoryName, COUNT(p.ProductID) AS ProductCount

FROM Categories c

JOIN Products p ON c.CategoryID = p.CategoryID

GROUP BY c.CategoryName

HAVING COUNT(p.ProductID) > 2;
```

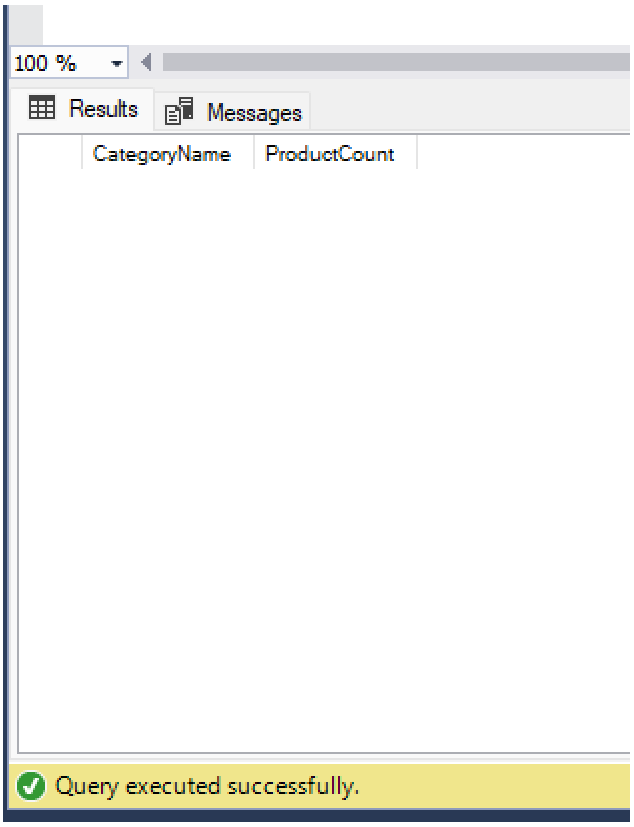


```
CategoryName    ProductCount
```

Query executed successfully.

45. Get products with discounts greater than 20.

```
SELECT p.Name, d.DiscountAmount

FROM Discounts d

JOIN Products p ON d.ProductID = p.ProductID

WHERE d.DiscountAmount > 20;
```



CategoryName	ProductCount

Query executed successfully.