Coding Challenge 6: Ecommerce – SQL

-Sarthak Niranjan Kulkarni (Maverick)

- sarthakkul2311@gmail.com - (+91) 93256 02791

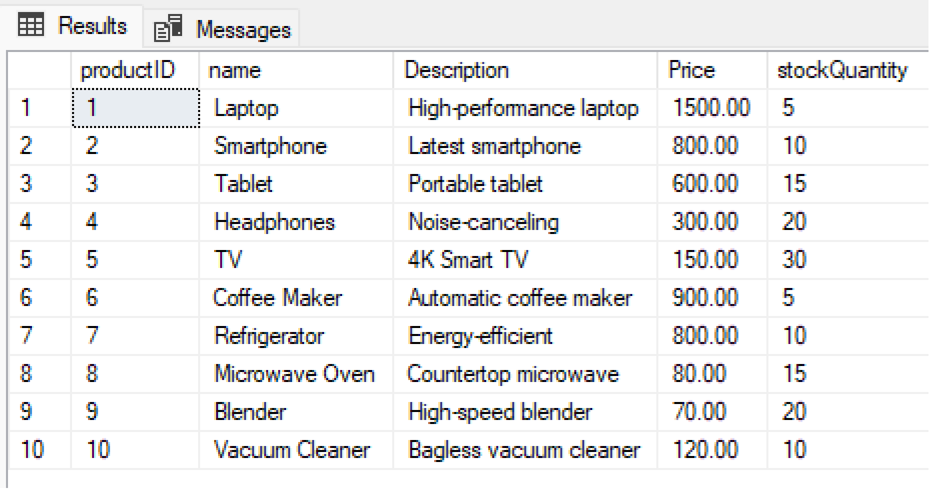
1. Update refrigerator product price to 800.

→ update products

set Price=800.00

where name='Refrigerator';

select * from products;

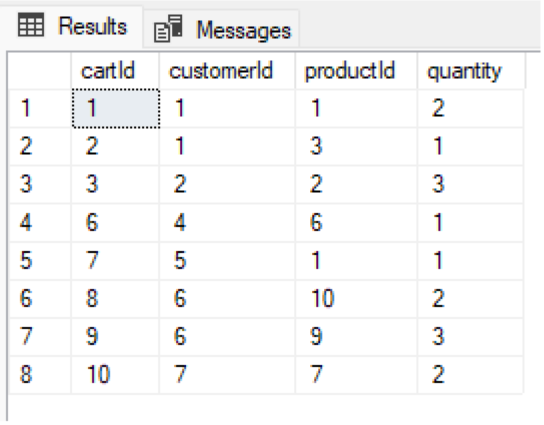


	productID	name	Description	Price	stockQuantity
1	1	Laptop	High-performance laptop	1500.00	5
2	2	Smartphone	Latest smartphone	800.00	10
3	3	Tablet	Portable tablet	600.00	15
4	4	Headphones	Noise-canceling	300.00	20
5	5	TV	4K Smart TV	150.00	30
6	6	Coffee Maker	Automatic coffee maker	900.00	5
7	7	Refrigerator	Energy-efficient	800.00	10
8	8	Microwave Oven	Countertop microwave	80.00	15
9	9	Blender	High-speed blender	70.00	20
10	10	Vacuum Cleaner	Bagless vacuum cleaner	120.00	10

2. Remove all cart items for a specific customer.

→ delete from cart where customerId = 3;

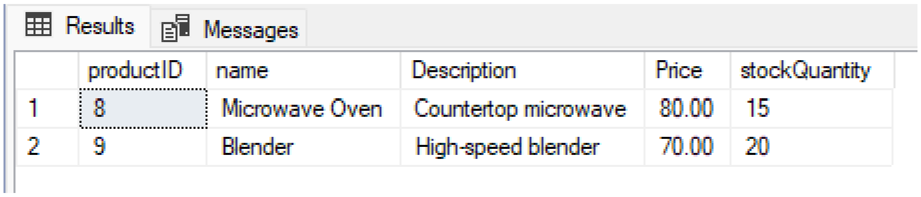
select * from cart;



	cartId	customerId	productId	quantity
1	1	1	1	2
2	2	1	3	1
3	3	2	2	3
4	6	4	6	1
5	7	5	1	1
6	8	6	10	2
7	9	6	9	3
8	10	7	7	2

3. Retrieve Products Priced Below $100.

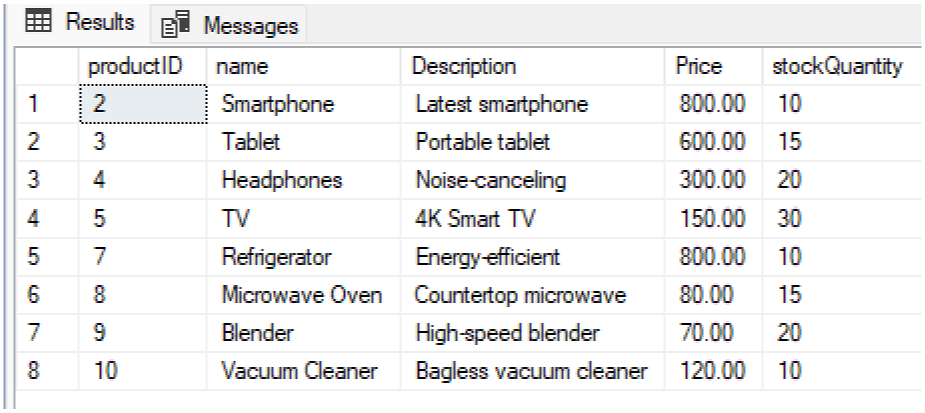
→ Select * from products where Price<100;



	productID	name	Description	Price	stockQuantity
1	8	Microwave Oven	Countertop microwave	80.00	15
2	9	Blender	High-speed blender	70.00	20

4. Find Products with Stock Quantity Greater Than 5.

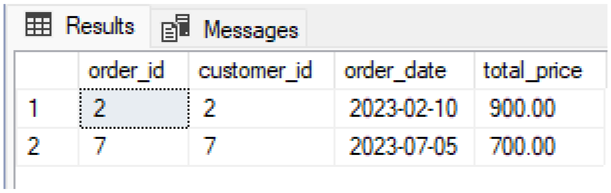
→ Select * from products where stockQuantity>5;



	productID	name	Description	Price	stockQuantity
1	2	Smartphone	Latest smartphone	800.00	10
2	3	Tablet	Portable tablet	600.00	15
3	4	Headphones	Noise-canceling	300.00	20
4	5	TV	4K Smart TV	150.00	30
5	7	Refrigerator	Energy-efficient	800.00	10
6	8	Microwave Oven	Countertop microwave	80.00	15
7	9	Blender	High-speed blender	70.00	20
8	10	Vacuum Cleaner	Bagless vacuum cleaner	120.00	10

5. Retrieve Orders with Total Amount Between $500 and $1000.

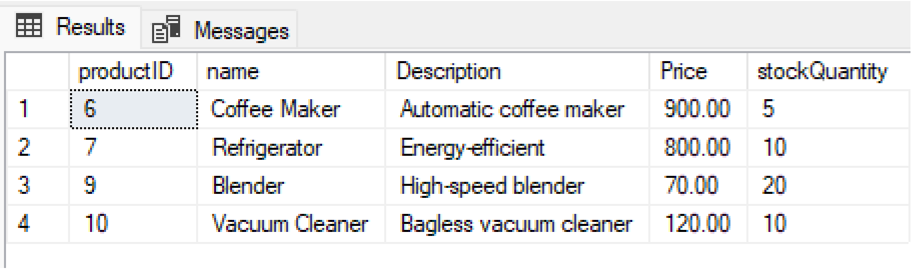
→ select * from orders where total_price between 500 and 1000;



	order_id	customer_id	order_date	total_price
1	2	2	2023-02-10	900.00
2	7	7	2023-07-05	700.00

6. Find Products which name end with letter ‘r’.

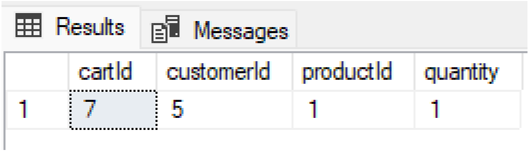
→ select * from products where name like '%r';



	productID	name	Description	Price	stockQuantity
1	6	Coffee Maker	Automatic coffee maker	900.00	5
2	7	Refrigerator	Energy-efficient	800.00	10
3	9	Blender	High-speed blender	70.00	20
4	10	Vacuum Cleaner	Bagless vacuum cleaner	120.00	10

7. Retrieve Cart Items for Customer 5.

→ select * from cart where customerId = 5;



	cartId	customerId	productId	quantity
1	7	5	1	1

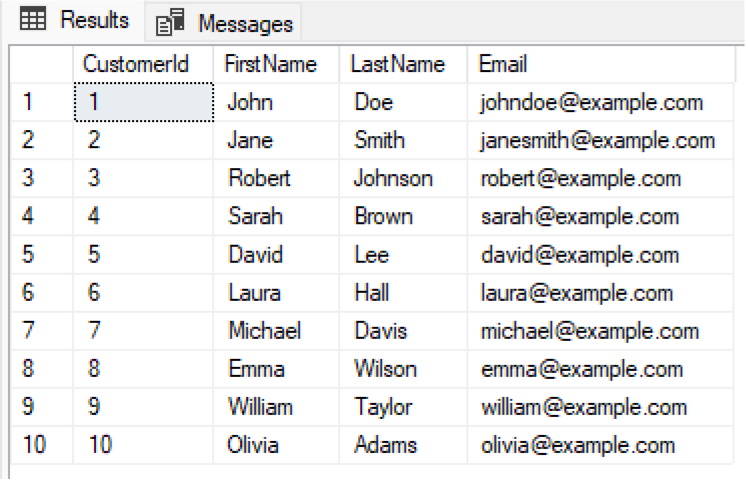
8. Find Customers Who Placed Orders in 2023.

→ SELECT DISTINCT c.CustomerId, c.FirstName, c.LastName, c.Email

FROM customers c

JOIN orders o ON c.CustomerId = o.customer_id

WHERE YEAR(o.order_date) = 2023;



	CustomerId	FirstName	LastName	Email
1	1	John	Doe	johndoe@example.com
2	2	Jane	Smith	janesmith@example.com
3	3	Robert	Johnson	robert@example.com
4	4	Sarah	Brown	sarah@example.com
5	5	David	Lee	david@example.com
6	6	Laura	Hall	laura@example.com
7	7	Michael	Davis	michael@example.com
8	8	Emma	Wilson	emma@example.com
9	9	William	Taylor	william@example.com
10	10	Olivia	Adams	olivia@example.com

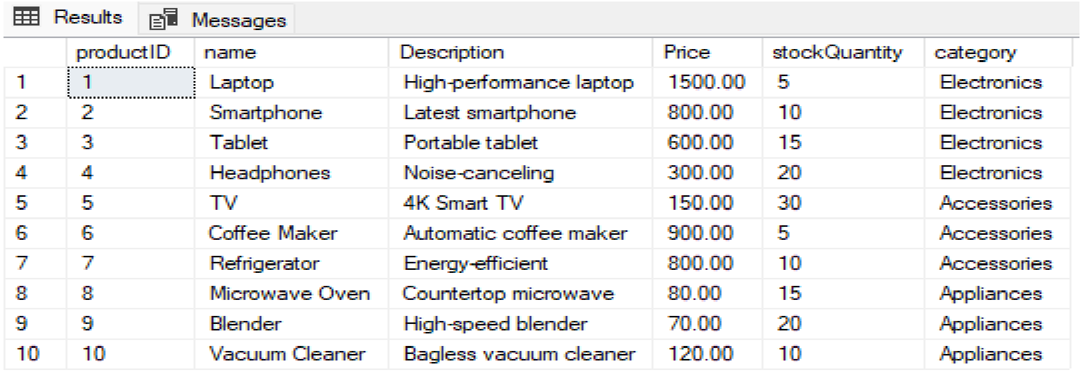
9. Determine the Minimum Stock Quantity for Each Product Category.

→ UPDATE products

SET category = CASE

```
  WHEN productID IN (1, 2, 3,4) THEN 'Electronics'
  WHEN productID IN (5, 6, 7) THEN 'Accessories'
  WHEN productID IN (8, 9, 10) THEN 'Appliances'
    END;
```

select * from products;



	productID	name	Description	Price	stockQuantity	category
1	1	Laptop	High-performance laptop	1500.00	5	Electronics
2	2	Smartphone	Latest smartphone	800.00	10	Electronics
3	3	Tablet	Portable tablet	600.00	15	Electronics
4	4	Headphones	Noise-canceling	300.00	20	Electronics
5	5	TV	4K Smart TV	150.00	30	Accessories
6	6	Coffee Maker	Automatic coffee maker	900.00	5	Accessories
7	7	Refrigerator	Energy-efficient	800.00	10	Accessories
8	8	Microwave Oven	Countertop microwave	80.00	15	Appliances
9	9	Blender	High-speed blender	70.00	20	Appliances
10	10	Vacuum Cleaner	Bagless vacuum cleaner	120.00	10	Appliances

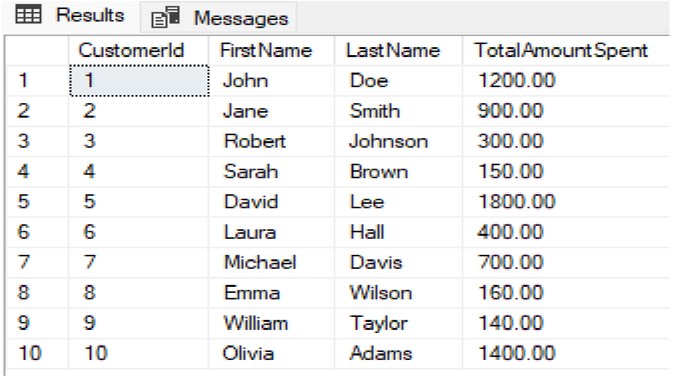
10. Calculate the Total Amount Spent by Each Customer.

→ SELECT c.CustomerId, c.FirstName, c.LastName, SUM(o.total_price) AS TotalAmountSpent

FROM customers c

JOIN orders o ON c.CustomerId = o.customer_id

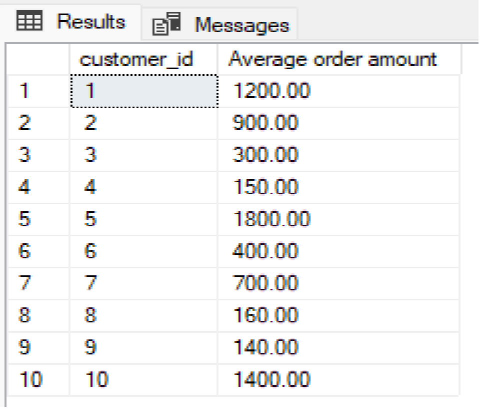
GROUP BY c.CustomerId, c.FirstName, c.LastName;



	CustomerId	FirstName	LastName	TotalAmountSpent
1	1	John	Doe	1200.00
2	2	Jane	Smith	900.00
3	3	Robert	Johnson	300.00
4	4	Sarah	Brown	150.00
5	5	David	Lee	1800.00
6	6	Laura	Hall	400.00
7	7	Michael	Davis	700.00
8	8	Emma	Wilson	160.00
9	9	William	Taylor	140.00
10	10	Olivia	Adams	1400.00

11. Find the Average Order Amount for Each Customer.

→ select customer_id, avg(total_price) as 'Average order amount' from orders group by customer_id;



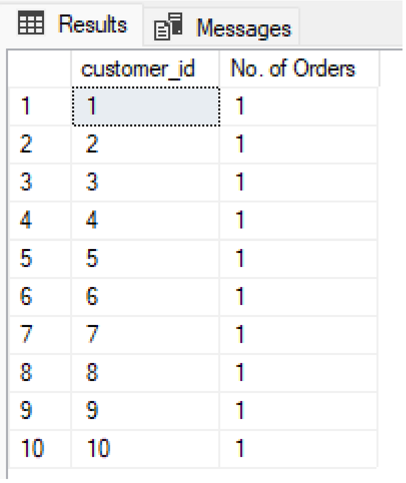
	customer_id	Average order amount
1	1	1200.00
2	2	900.00
3	3	300.00
4	4	150.00
5	5	1800.00
6	6	400.00
7	7	700.00
8	8	160.00
9	9	140.00
10	10	1400.00

12. Count the Number of Orders Placed by Each Customer.

```
→ SELECT customer_id, COUNT(*) as 'No. of Orders'

FROM orders

GROUP BY customer_id;
```



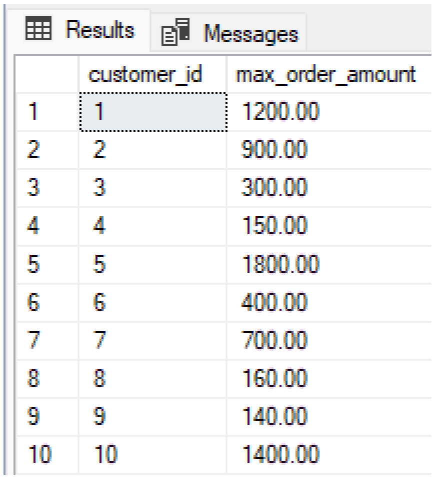
	customer_id	No. of Orders
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	6	1
7	7	1
8	8	1
9	9	1
10	10	1

13. Find the Maximum Order Amount for Each Customer.

```
→ SELECT customer_id, MAX(total_price) AS max_order_amount

FROM orders

GROUP BY customer_id;
```



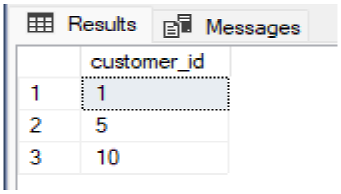
	customer_id	max_order_amount
1	1	1200.00
2	2	900.00
3	3	300.00
4	4	150.00
5	5	1800.00
6	6	400.00
7	7	700.00
8	8	160.00
9	9	140.00
10	10	1400.00

14. Get Customers Who Placed Orders Totaling Over $1000.

```
→ SELECT customer_id FROM orders

GROUP BY customer_id

HAVING SUM(total_price) > 1000;
```

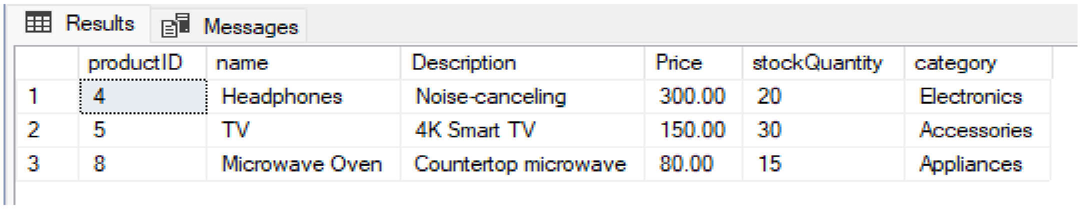


	customer_id
1	1
2	5
3	10

15. Subquery to Find Products Not in the Cart.

→ SELECT * FROM products

WHERE productID NOT IN (SELECT productID FROM cart);



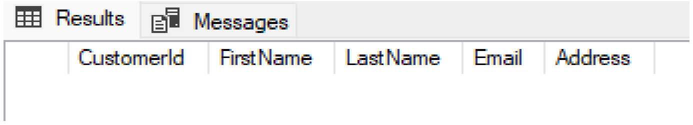
	productID	name	Description	Price	stockQuantity	category
1	4	Headphones	Noise-canceling	300.00	20	Electronics
2	5	TV	4K Smart TV	150.00	30	Accessories
3	8	Microwave Oven	Countertop microwave	80.00	15	Appliances

16. Subquery to Find Customers Who Haven't Placed Orders.

→ SELECT * FROM customers

WHERE CustomerId NOT IN (SELECT customer_id FROM orders);

---(Everyone has place order so output should not display anything)



	CustomerId	FirstName	LastName	Email	Address

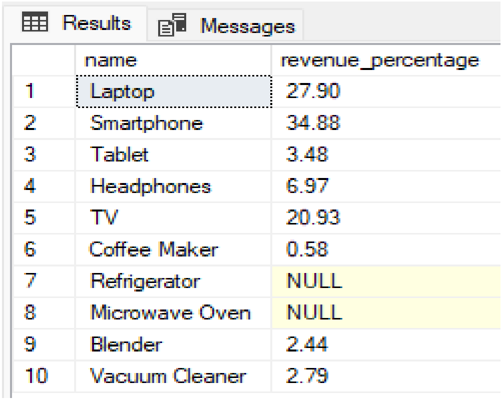
17. Subquery to Calculate the Percentage of Total Revenue for a Product.

→ SELECT name,

(SELECT SUM(itemamount) FROM order_items WHERE product_id = p.productID) /

(SELECT SUM(itemamount) FROM order_items) * 100 AS revenue_percentage

FROM products p;

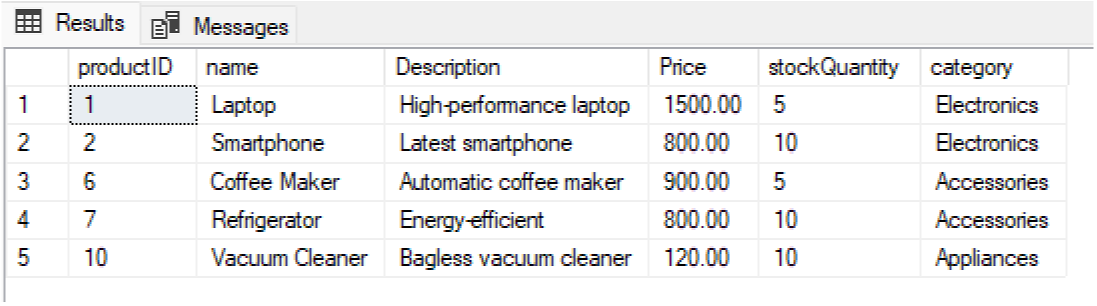


	name	revenue_percentage
1	Laptop	27.90
2	Smartphone	34.88
3	Tablet	3.48
4	Headphones	6.97
5	TV	20.93
6	Coffee Maker	0.58
7	Refrigerator	NULL
8	Microwave Oven	NULL
9	Blender	2.44
10	Vacuum Cleaner	2.79

18. Subquery to Find Products with Low Stock.

→ SELECT * FROM products

WHERE stockQuantity < (SELECT AVG(stockQuantity) FROM products);



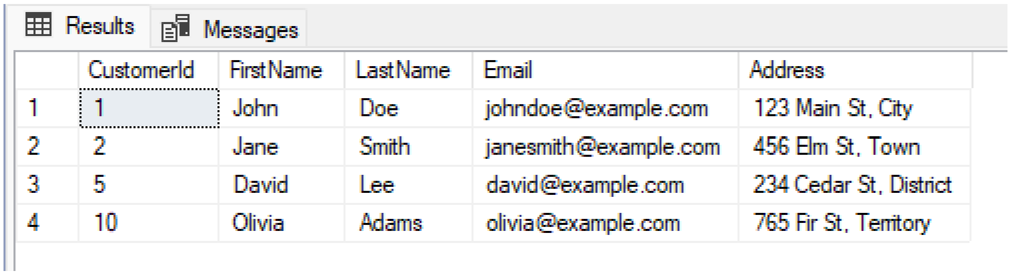
	productID	name	Description	Price	stockQuantity	category
1	1	Laptop	High-performance laptop	1500.00	5	Electronics
2	2	Smartphone	Latest smartphone	800.00	10	Electronics
3	6	Coffee Maker	Automatic coffee maker	900.00	5	Accessories
4	7	Refrigerator	Energy-efficient	800.00	10	Accessories
5	10	Vacuum Cleaner	Bagless vacuum cleaner	120.00	10	Appliances

19. Subquery to Find Customers Who Placed High-Value Orders.

→ select c.* from customers c

join orders o on c.CustomerId = o.customer_id

where o.total_price > (select avg (total_price) from orders);



	CustomerId	FirstName	LastName	Email	Address
1	1	John	Doe	johndoe@example.com	123 Main St, City
2	2	Jane	Smith	janesmith@example.com	456 Elm St, Town
3	5	David	Lee	david@example.com	234 Cedar St, District
4	10	Olivia	Adams	olivia@example.com	765 Fir St, Territory