CODING CHALLENGE-ECOMM

CREATION OF TABLES:

CUSTOMER TABLE:

```
CREATE TABLE customer (
   customer_id INT PRIMARY KEY,
   name VARCHAR(100),
   email VARCHAR(100),
   password VARCHAR(100)
);
```

PRODUCTS TABLE:

```
CREATE TABLE products (
   product_id INT PRIMARY KEY,
   name VARCHAR(100),
   price DECIMAL(10, 2),
   description VARCHAR(255),
   stockQuantity INT
);
```

CART TABLE:

```
CREATE TABLE cart (
   cart_id INT IDENTITY(1,1) PRIMARY KEY,
   customer_id INT NOT NULL,
   product_id INT NOT NULL,
   quantity INT NOT NULL CHECK (quantity > 0),
   FOREIGN KEY (customer_id) REFERENCES customer(customer_id) ON DELETE CASCADE,
   FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE CASCADE
);
```

ORDERS TABLE:

```
CREATE TABLE orders3 (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    total_price DECIMAL(10, 2),
    shipping_address VARCHAR(255),
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
);
```

ORDER_ITEMS TABLE:

```
CREATE TABLE order_items1 (
    order_item_id INT IDENTITY(1,1) PRIMARY KEY,
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL CHECK (quantity > 0),
    item_amount DECIMAL(10,2) NOT NULL CHECK (item_amount >= 0),
    FOREIGN KEY (order_id) REFERENCES orders3(order_id) ON DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE CASCADE
);
--Insertion
INSERT INTO customer (customer_id, name, email, password) VALUES
(1, 'John Doe', 'johndoe@example.com', 'password1'),
(2, 'Jane Smith', 'janesmith@example.com', 'password2'),
(3, 'Robert Johnson', 'robert@example.com', 'password3'),
(4, 'Sarah Brown', 'sarah@example.com', 'password4'),
(5, 'David Lee', 'david@example.com', 'password5'),
(6, 'Laura Hall', 'laura@example.com', 'password6'),
(7, 'Michael Davis', 'michael@example.com', 'password7'),
(8, 'Emma Wilson', 'emma@example.com', 'password8'),
(9, 'William Taylor', 'william@example.com', 'password9'),
(10, 'Olivia Adams', 'olivia@example.com', 'password10');

--

INSERT INTO products (product_id, name, price, description, stockQuantity) VALUES

(1, 'Laptop', 1200.00, 'High-performance laptop', 25),

(2, 'Smartphone', 800.00, 'Latest smartphone', 10),

(3, 'Tablet', 600.00, 'Portable tablet', 15),

(4, 'Headphones', 300.00, 'Noise-canceling', 20),

(5, 'TV', 1500.00, '4K Smart TV', 30),

(6, 'Coffee Maker', 900.00, 'Automatic coffee maker', 5),

(7, 'Refrigerator', 700.00, 'Energy-efficient', 10),

(8, 'Microwave Oven', 80.00, 'Countertop microwave', 15),

(9, 'Blender', 70.00, 'High-speed blender', 20),

(10, 'Vacuum Cleaner', 120.00, 'Bagless vacuum cleaner', 10);

--

INSERT INTO orders3 (order_id, customer_id, order_date, total_price, shipping_address)
VALUES

(1, 1, '2023-01-05', 2400.00, '123 Main St, City'),

(2, 2, '2023-02-10', 2400.00, '456 Elm St, Town'),

(3, 3, '2023-03-15', 1200.00, '789 Oak St, Village'),

(4, 4, '2023-04-20', 1200.00, '101 Pine St, Suburb'),

(5, 5, '2023-05-25', 2800.00, '234 Cedar St, District'),

(6, 6, '2023-06-30', 450.00, '567 Birch St, County'),

(7, 7, '2023-07-05', 700.00, '890 Maple St, State'),

(8, 8, '2023-08-10', 160.00, '321 Redwood St, Country'),

(9, 9, '2023-09-15', 140.00, '432 Spruce St, Province'),

(10, 10, '2023-10-20', 1400.00, '765 Fir St, Territory');

--

INSERT INTO order_items1 (order_id, product_id, quantity, item_amount) VALUES
```

(1, 1, 2, 2400.00),

(1, 3, 1, 300.00),

(2, 2, 3, 2400.00),

(3, 5, 2, 3000.00),

(4, 4, 4, 1200.00),

(4, 6, 1, 900.00),

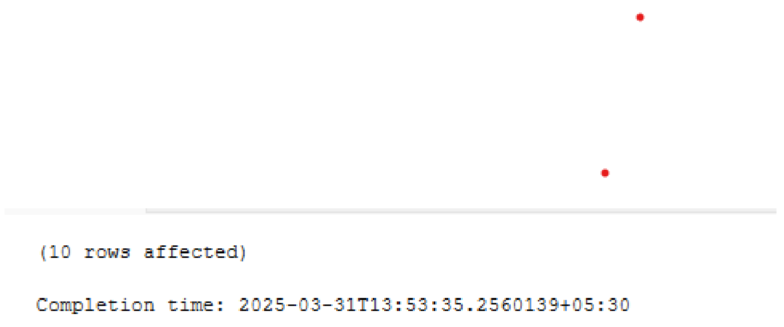
(5, 1, 1, 1200.00),

(5, 2, 2, 1600.00),

(6, 10, 2, 240.00),

(6, 9, 3, 210.00);

OUTPUT:



```
(10 rows affected)

Completion time: 2025-03-31T13:53:35.2560139+05:30
```

QUERIES:

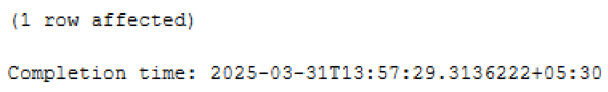
1. **Update refrigerator product price to 800.**
   CODE:

```
 UPDATE products

SET price = 800

WHERE name = 'Refrigerator';
```

OUTPUT:



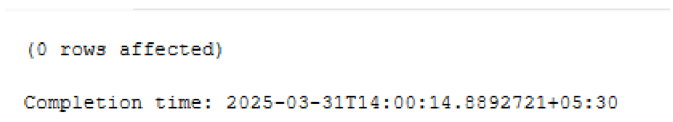
```
(1 row affected)

Completion time: 2025-03-31T13:57:29.3136222+05:30
```

2. **Remove all cart items for a specific customer.**
   CODE:

```
DELETE FROM cart
WHERE customer_id = 5;
```

   OUTPUT:



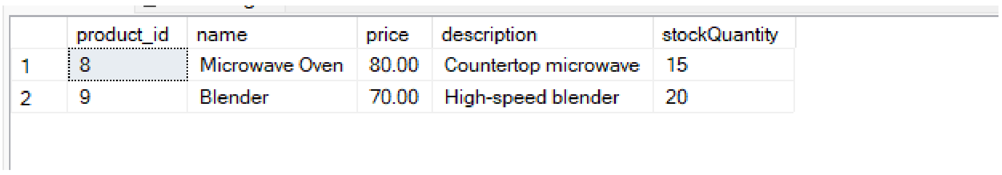
```
(0 rows affected)

Completion time: 2025-03-31T14:00:14.8892721+05:30
```

3. **Retrieve Products Priced Below $100.**
   CODE:

```
SELECT *
FROM products
WHERE price < 100;
```

   OUTPUT:

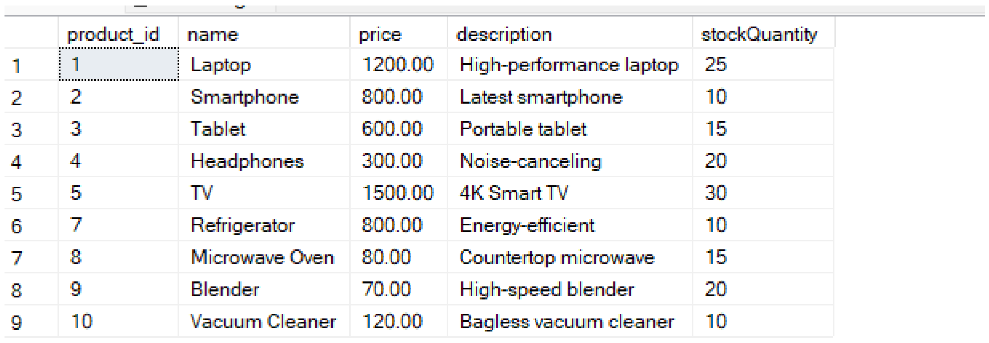


	product_id	name	price	description	stockQuantity
1	8	Microwave Oven	80.00	Countertop microwave	15
2	9	Blender	70.00	High-speed blender	20

4. **Find Products with Stock Quantity Greater Than 5.**
   CODE:

```
SELECT *
FROM products
WHERE stockQuantity > 5;
```

   OUTPUT:



	product_id	name	price	description	stockQuantity
1	1	Laptop	1200.00	High-performance laptop	25
2	2	Smartphone	800.00	Latest smartphone	10
3	3	Tablet	600.00	Portable tablet	15
4	4	Headphones	300.00	Noise-canceling	20
5	5	TV	1500.00	4K Smart TV	30
6	7	Refrigerator	800.00	Energy-efficient	10
7	8	Microwave Oven	80.00	Countertop microwave	15
8	9	Blender	70.00	High-speed blender	20
9	10	Vacuum Cleaner	120.00	Bagless vacuum cleaner	10

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

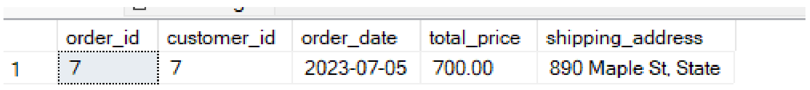
CODE:

SELECT *

FROM orders3

WHERE total_price BETWEEN 500 AND 1000;

OUTPUT:



	order_id	customer_id	order_date	total_price	shipping_address
1	7	7	2023-07-05	700.00	890 Maple St, State

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

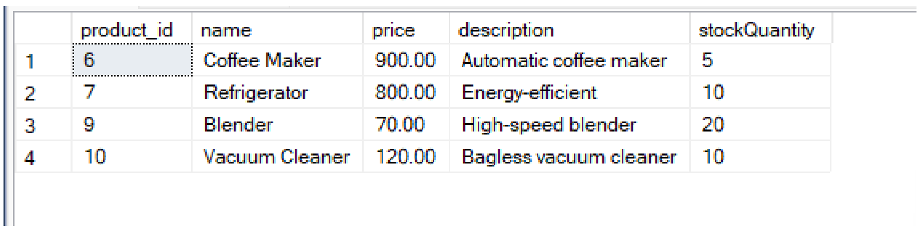
   CODE:

   SELECT *

   FROM products

   WHERE name LIKE '%r';

   OUTPUT:



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

7. **Retrieve Cart Items for Customer 5.**

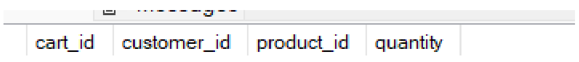
   CODE:

   SELECT *

   FROM cart

   WHERE customer_id = 5;

   OUTPUT:



cart_id	customer_id	product_id	quantity

8. **Find Customers Who Placed Orders in 2023.**

   CODE:

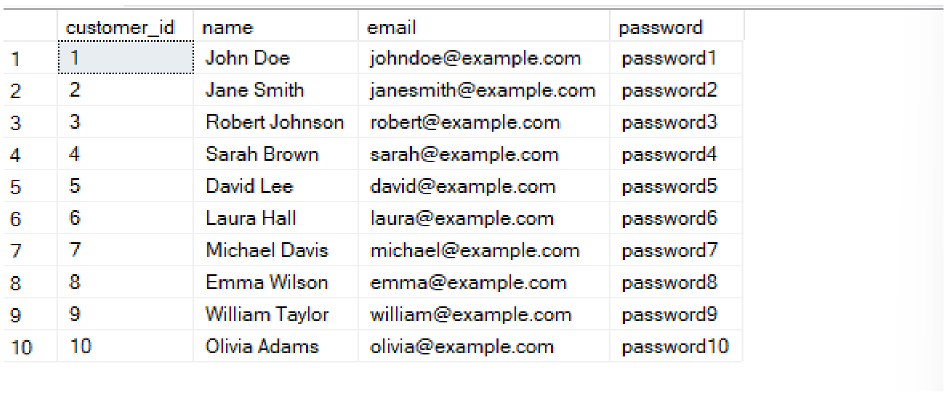
   SELECT DISTINCT c.*

   FROM customers c

   JOIN orders3 o ON c.customer_id = o.customer_id

   WHERE YEAR(o.order_date) = 2023;

   OUPUT:



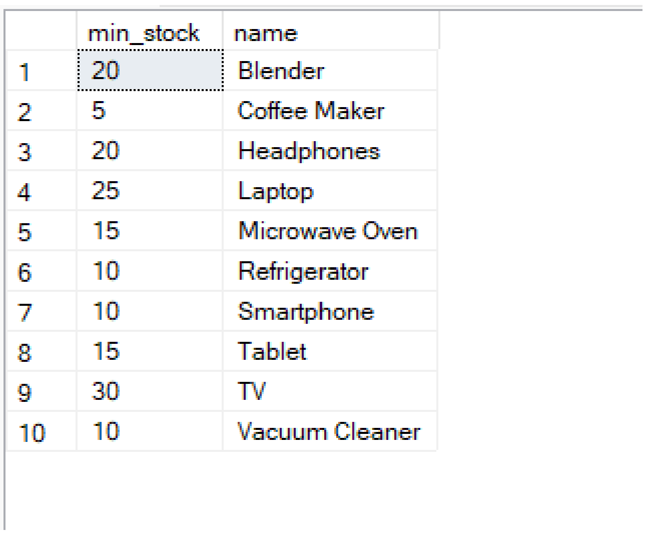
	customer_id	name	email	password
1	1	John Doe	johndoe@example.com	password1
2	2	Jane Smith	janesmith@example.com	password2
3	3	Robert Johnson	robert@example.com	password3
4	4	Sarah Brown	sarah@example.com	password4
5	5	David Lee	david@example.com	password5
6	6	Laura Hall	laura@example.com	password6
7	7	Michael Davis	michael@example.com	password7
8	8	Emma Wilson	emma@example.com	password8
9	9	William Taylor	william@example.com	password9
10	10	Olivia Adams	olivia@example.com	password10

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

CODE:

```
SELECT MIN(stockQuantity) AS min_stock, name
FROM products
GROUP BY name;
```

OUTPUT:



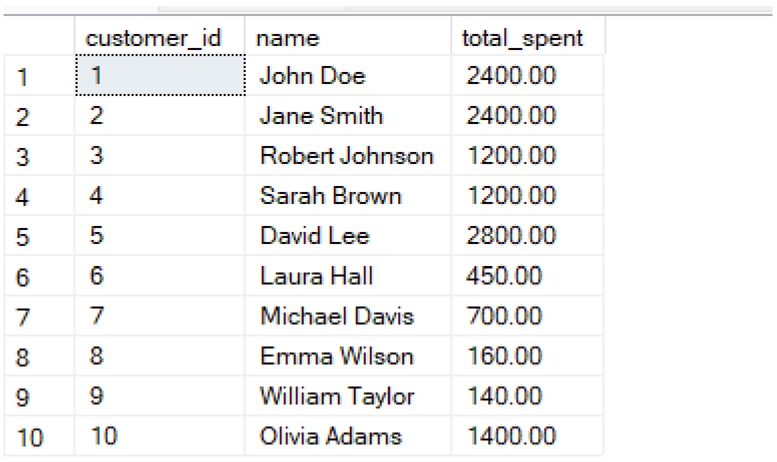
	min_stock	name
1	20	Blender
2	5	Coffee Maker
3	20	Headphones
4	25	Laptop
5	15	Microwave Oven
6	10	Refrigerator
7	10	Smartphone
8	15	Tablet
9	30	TV
10	10	Vacuum Cleaner

10. **Calculate the Total Amount Spent by Each Customer.**

CODE:

```
SELECT c.customer_id, c.name, SUM(o.total_price) AS total_spent
FROM customers c
JOIN orders3 o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.name;
```

OUTPUT:



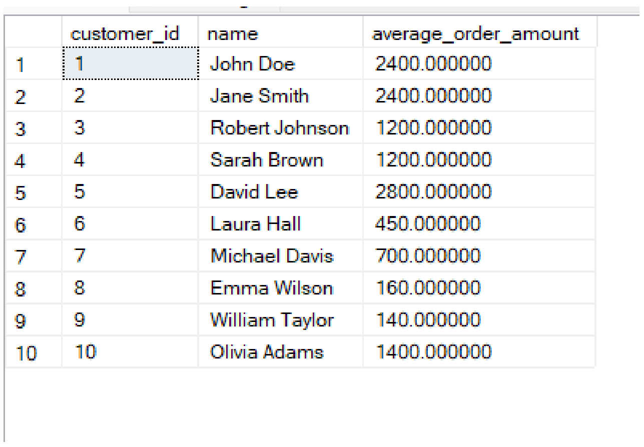
	customer_id	name	total_spent
1	1	John Doe	2400.00
2	2	Jane Smith	2400.00
3	3	Robert Johnson	1200.00
4	4	Sarah Brown	1200.00
5	5	David Lee	2800.00
6	6	Laura Hall	450.00
7	7	Michael Davis	700.00
8	8	Emma Wilson	160.00
9	9	William Taylor	140.00
10	10	Olivia Adams	1400.00

11. Find the Average Order Amount for Each Customer.

CODE:

```
SELECT c.customer_id, c.name, AVG(o.total_price) AS average_order_amount
FROM customers c
JOIN orders3 o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.name;
```

OUTPUT:



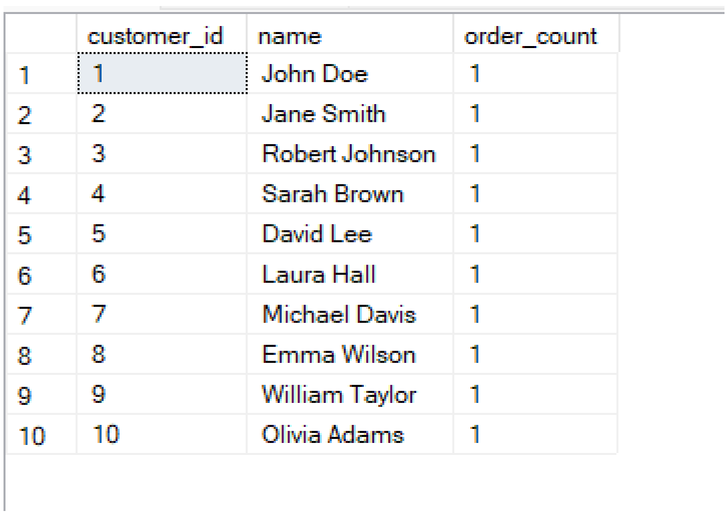
	customer_id	name	average_order_amount
1	1	John Doe	2400.000000
2	2	Jane Smith	2400.000000
3	3	Robert Johnson	1200.000000
4	4	Sarah Brown	1200.000000
5	5	David Lee	2800.000000
6	6	Laura Hall	450.000000
7	7	Michael Davis	700.000000
8	8	Emma Wilson	160.000000
9	9	William Taylor	140.000000
10	10	Olivia Adams	1400.000000

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

CODE:

```
SELECT c.customer_id, c.name, COUNT(o.order_id) AS order_count
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.name;
```

OUTPUT:



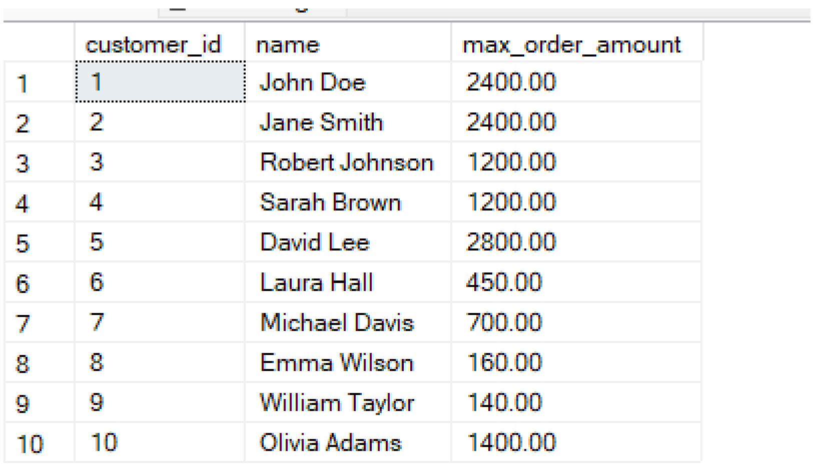
	customer_id	name	order_count
1	1	John Doe	1
2	2	Jane Smith	1
3	3	Robert Johnson	1
4	4	Sarah Brown	1
5	5	David Lee	1
6	6	Laura Hall	1
7	7	Michael Davis	1
8	8	Emma Wilson	1
9	9	William Taylor	1
10	10	Olivia Adams	1

13. Find the Maximum Order Amount for Each Customer.

CODE:

```
SELECT c.customer_id, c.name, MAX(o.total_price) AS max_order_amount
FROM customer c
JOIN orders3 o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.name;
```

OUTPUT:



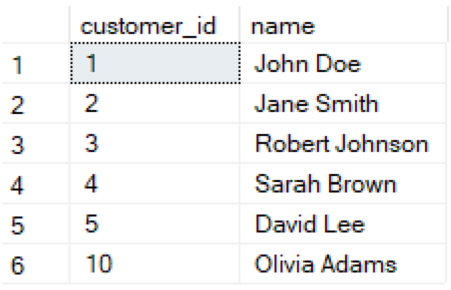
	customer_id	name	max_order_amount
1	1	John Doe	2400.00
2	2	Jane Smith	2400.00
3	3	Robert Johnson	1200.00
4	4	Sarah Brown	1200.00
5	5	David Lee	2800.00
6	6	Laura Hall	450.00
7	7	Michael Davis	700.00
8	8	Emma Wilson	160.00
9	9	William Taylor	140.00
10	10	Olivia Adams	1400.00

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

CODE:

```
SELECT c.customer_id, c.name
FROM customers c
JOIN orders3 o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.name
HAVING SUM(o.total_price) > 1000;
```

OUTPUT:



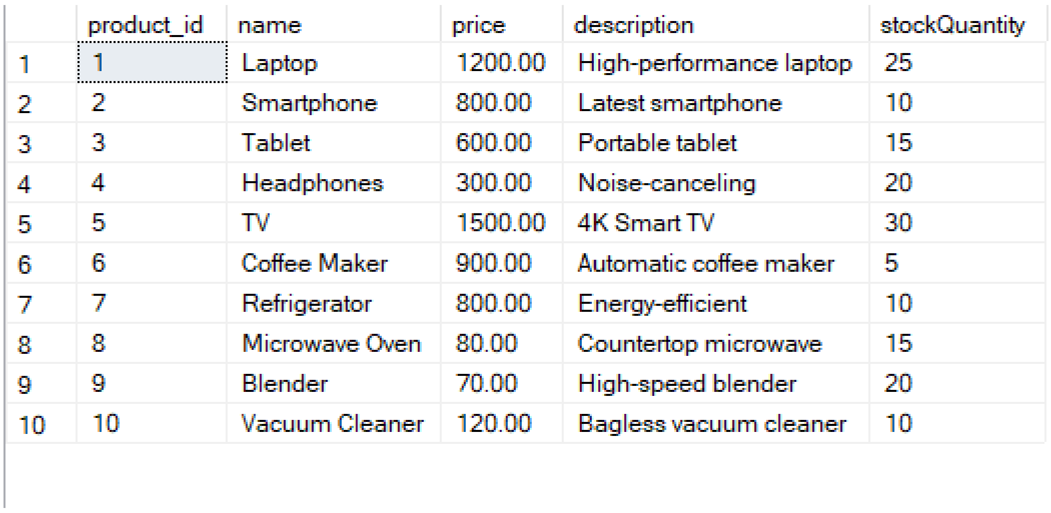
	customer_id	name
1	1	John Doe
2	2	Jane Smith
3	3	Robert Johnson
4	4	Sarah Brown
5	5	David Lee
6	10	Olivia Adams

15. Subquery to Find Products Not in the Cart.

CODE:

```
SELECT *
FROM products
WHERE product_id NOT IN (SELECT product_id FROM cart);
```

OUTPUT:



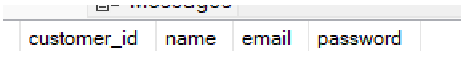
	product_id	name	price	description	stockQuantity
1	1	Laptop	1200.00	High-performance laptop	25
2	2	Smartphone	800.00	Latest smartphone	10
3	3	Tablet	600.00	Portable tablet	15
4	4	Headphones	300.00	Noise-canceling	20
5	5	TV	1500.00	4K Smart TV	30
6	6	Coffee Maker	900.00	Automatic coffee maker	5
7	7	Refrigerator	800.00	Energy-efficient	10
8	8	Microwave Oven	80.00	Countertop microwave	15
9	9	Blender	70.00	High-speed blender	20
10	10	Vacuum Cleaner	120.00	Bagless vacuum cleaner	10

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

CODE:

```
SELECT *
FROM customers
WHERE customer_id NOT IN (SELECT customer_id FROM orders3);
```

OUTPUT:



customer_id	name	email	password

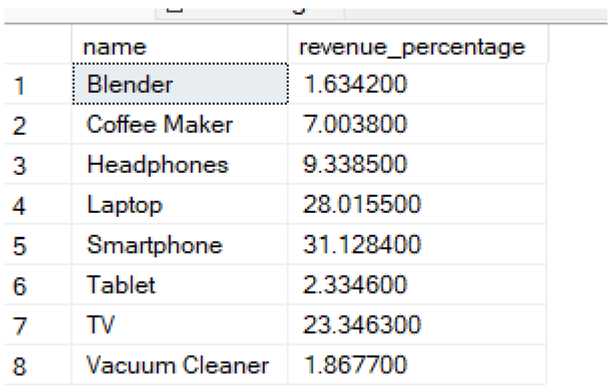
17. Subquery to Calculate the Percentage of Total Revenue for a Product.

CODE:

```
SELECT p.name,
    (SUM(oi.itemAmount) / (SELECT SUM(total_price) FROM orders3) * 100) AS
revenue_percentage
FROM orders3_items oi
JOIN products p ON oi.product_id = p.product_id
```

```
GROUP BY p.name;
```

OUTPUT:



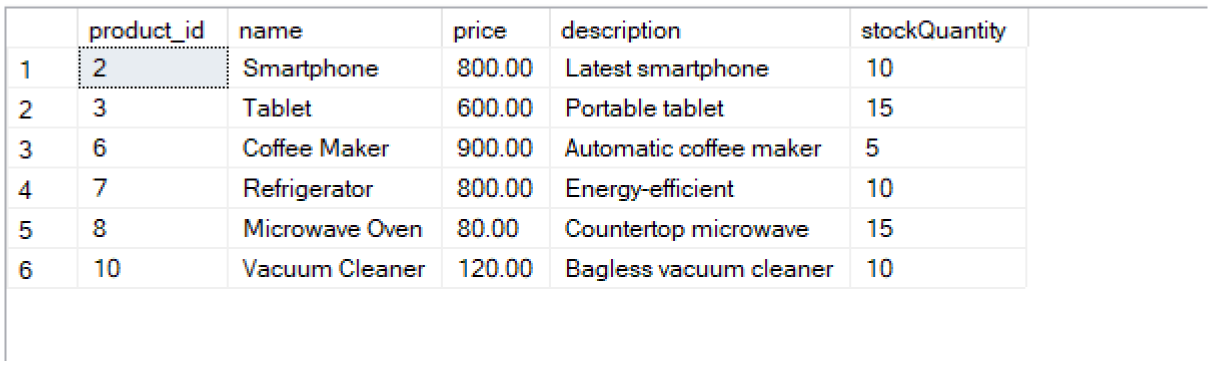
	name	revenue_percentage
1	Blender	1.634200
2	Coffee Maker	7.003800
3	Headphones	9.338500
4	Laptop	28.015500
5	Smartphone	31.128400
6	Tablet	2.334600
7	TV	23.346300
8	Vacuum Cleaner	1.867700

18. Subquery to Find Products with Low Stock.

CODE:

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

OUTPUT:



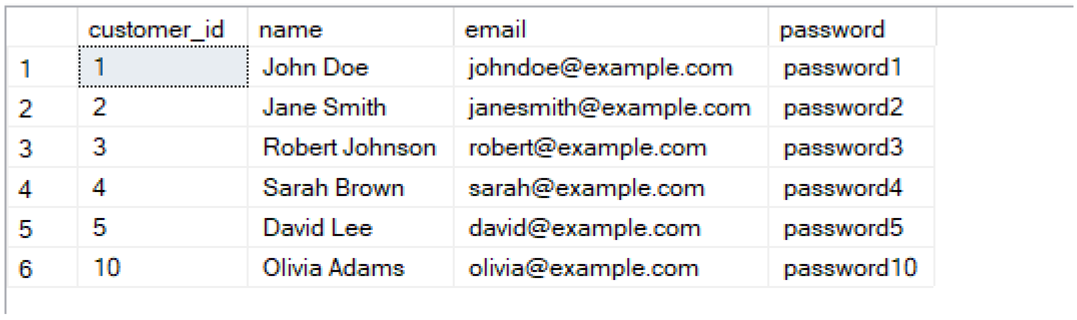
	product_id	name	price	description	stockQuantity
1	2	Smartphone	800.00	Latest smartphone	10
2	3	Tablet	600.00	Portable tablet	15
3	6	Coffee Maker	900.00	Automatic coffee maker	5
4	7	Refrigerator	800.00	Energy-efficient	10
5	8	Microwave Oven	80.00	Countertop microwave	15
6	10	Vacuum Cleaner	120.00	Bagless vacuum cleaner	10

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

CODE:

```
SELECT *
FROM customer
WHERE customer_id IN (SELECT customer_id FROM orders3 WHERE total_price >
1000);
```

OUTPUT:



	customer_id	name	email	password
1	1	John Doe	johndoe@example.com	password1
2	2	Jane Smith	janesmith@example.com	password2
3	3	Robert Johnson	robert@example.com	password3
4	4	Sarah Brown	sarah@example.com	password4
5	5	David Lee	david@example.com	password5
6	10	Olivia Adams	olivia@example.com	password10