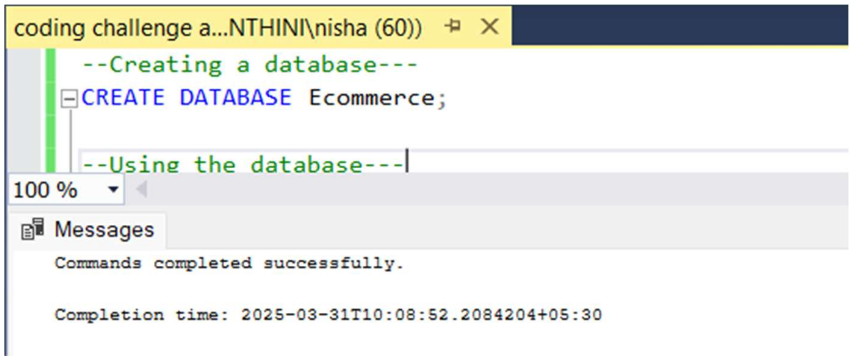
CODING CHALLENGE - 1 DOCUMENTATION (31-03-2025)

ECOMMERCE -SQL

CHECKING WITH THE PREREQUISITES BEFORE SOLVING THE QUESTION:

STEP 1: Create a database called Ecommerce.



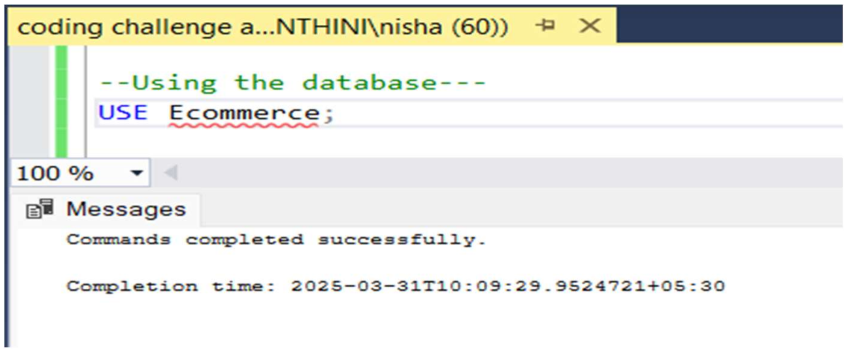
```
--Creating a database---
CREATE DATABASE Ecommerce;

--Using the database---
```

```
Commands completed successfully.

Completion time: 2025-03-31T10:08:52.2084204+05:30
```

STEP 2: Using the database that was created.

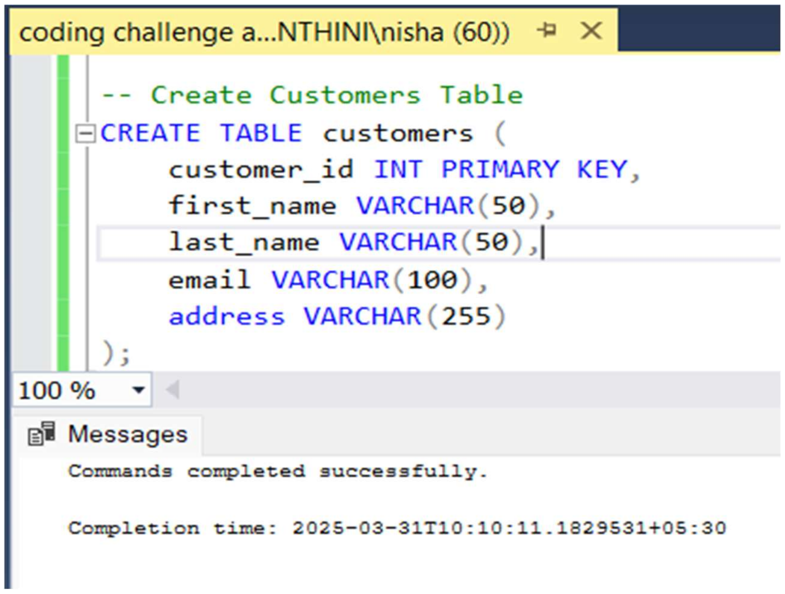


```
--Using the database---
USE Ecommerce;
```

```
Commands completed successfully.

Completion time: 2025-03-31T10:09:29.9524721+05:30
```

STEP 3: Creating the customers table with the given constraints (mentioning first name and last name following the examples given).

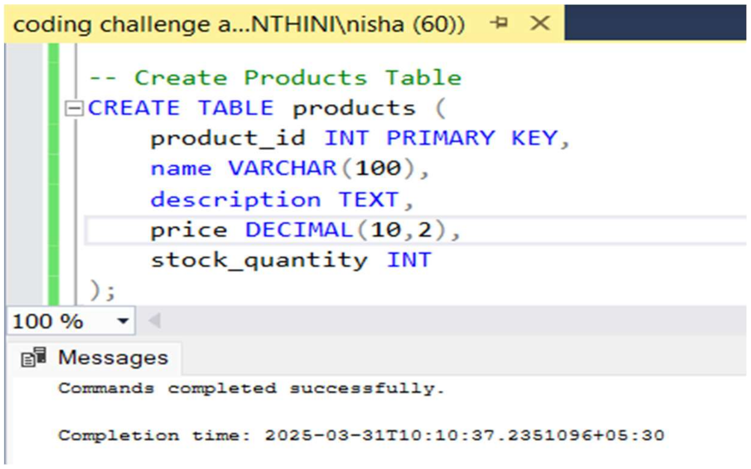


```
-- Create Customers Table
CREATE TABLE customers (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    address VARCHAR(255)
);
```

```
Commands completed successfully.

Completion time: 2025-03-31T10:10:11.1829531+05:30
```

STEP 4: Creating the products table with the given constraints.

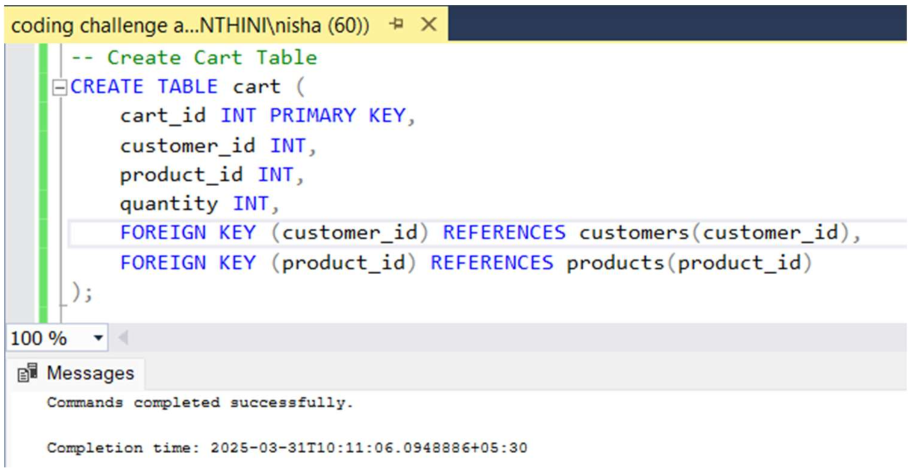


```
-- Create Products Table
CREATE TABLE products (
    product_id INT PRIMARY KEY,
    name VARCHAR(100),
    description TEXT,
    price DECIMAL(10,2),
    stock_quantity INT
);
```

```
Commands completed successfully.

Completion time: 2025-03-31T10:10:37.2351096+05:30
```

STEP 5: Creating the cart table with the given constraints.

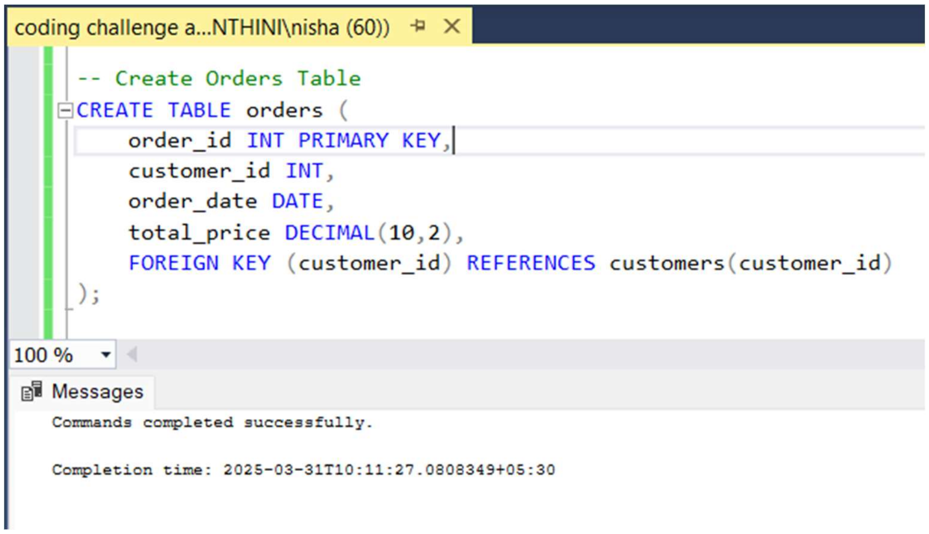


```
-- Create Cart Table
CREATE TABLE cart (
    cart_id INT PRIMARY KEY,
    customer_id INT,
    product_id INT,
    quantity INT,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

```
Commands completed successfully.

Completion time: 2025-03-31T10:11:06.0948886+05:30
```

STEP 6: Creating the orders table with the given constraints.

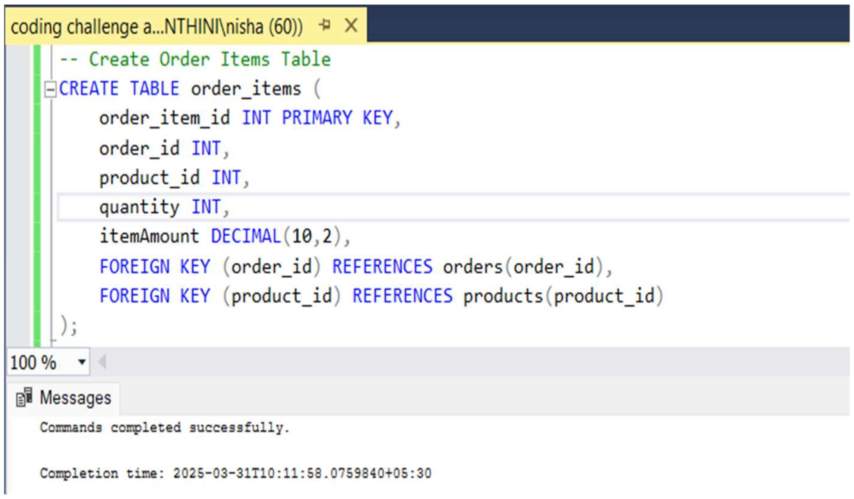


```
-- Create Orders Table
CREATE TABLE orders (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    total_price DECIMAL(10,2),
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
```

```
Commands completed successfully.

Completion time: 2025-03-31T10:11:27.0808349+05:30
```

STEP 7: Creating the order items table with the given constraints.



coding challenge a...NTHINI\nisha (60))

```
-- Create Order Items Table
CREATE TABLE order_items (
    order_item_id INT PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT,
    itemAmount DECIMAL(10,2),
    FOREIGN KEY (order_id) REFERENCES orders(order_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

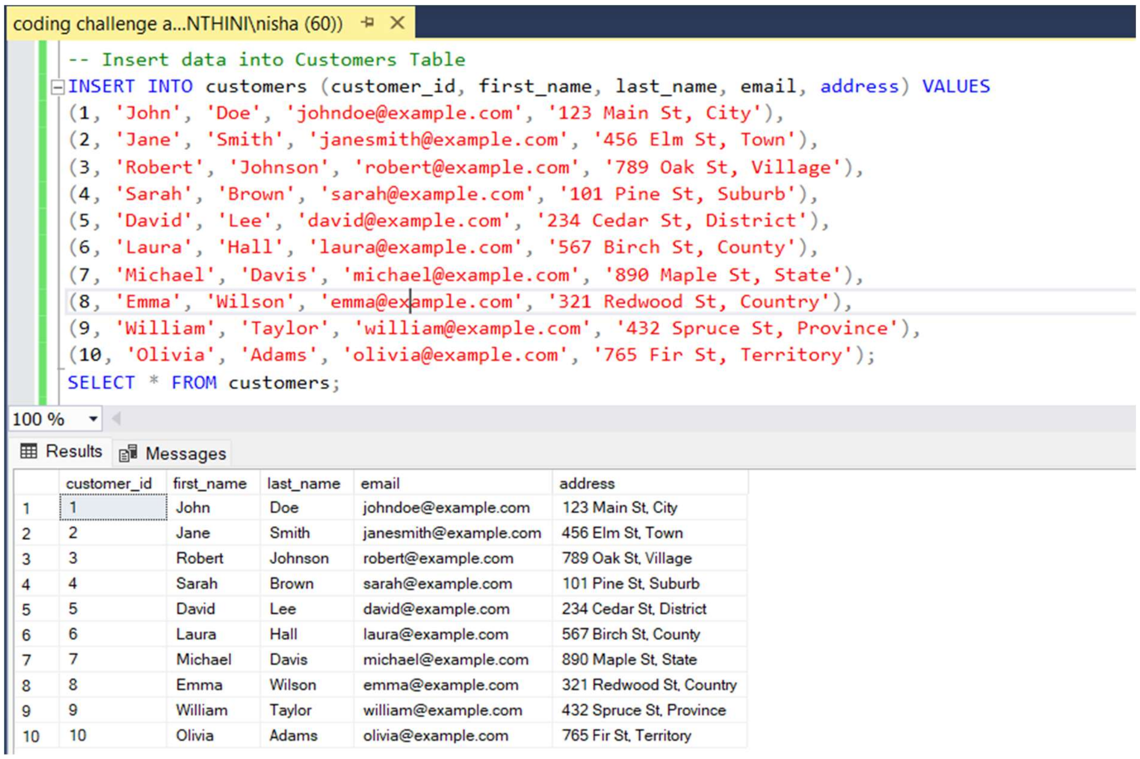
100 %

Messages

```
Commands completed successfully.

Completion time: 2025-03-31T10:11:58.0759840+05:30
```

STEP 8: Inserting values to the customer table as per the given example.



coding challenge a...NTHINI\nisha (60))

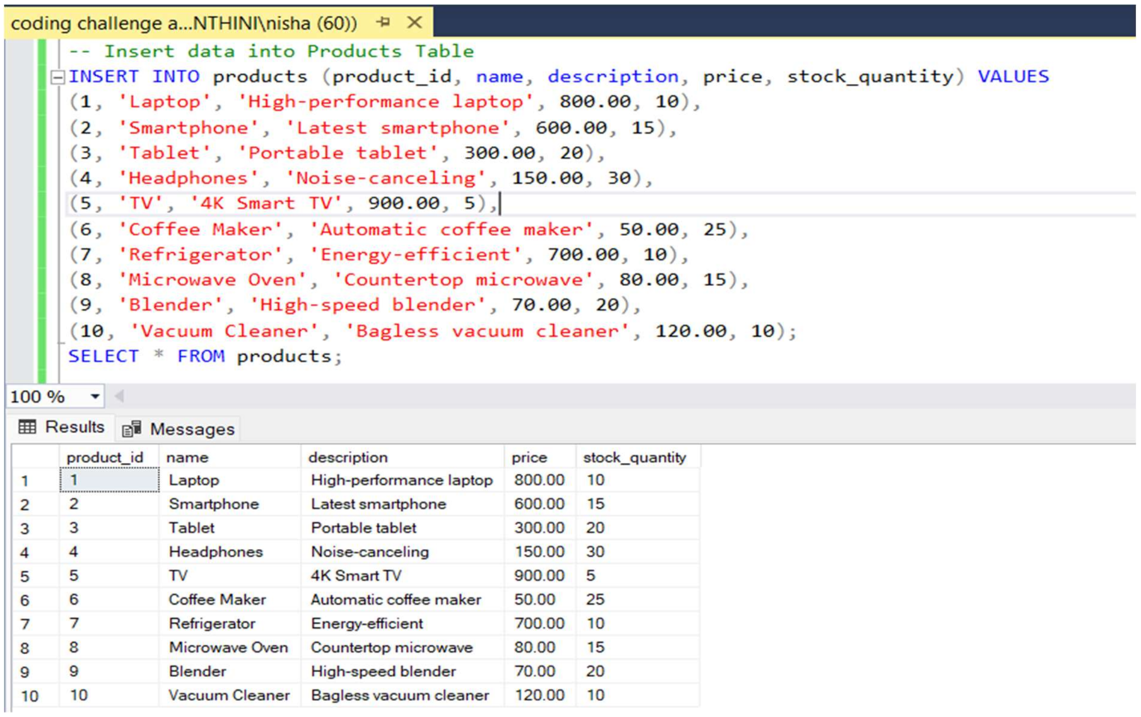
```
-- Insert data into Customers Table
INSERT INTO customers (customer_id, first_name, last_name, email, address) VALUES
(1, 'John', 'Doe', 'johndoe@example.com', '123 Main St, City'),
(2, 'Jane', 'Smith', 'janesmith@example.com', '456 Elm St, Town'),
(3, 'Robert', 'Johnson', 'robert@example.com', '789 Oak St, Village'),
(4, 'Sarah', 'Brown', 'sarah@example.com', '101 Pine St, Suburb'),
(5, 'David', 'Lee', 'david@example.com', '234 Cedar St, District'),
(6, 'Laura', 'Hall', 'laura@example.com', '567 Birch St, County'),
(7, 'Michael', 'Davis', 'michael@example.com', '890 Maple St, State'),
(8, 'Emma', 'Wilson', 'emma@example.com', '321 Redwood St, Country'),
(9, 'William', 'Taylor', 'william@example.com', '432 Spruce St, Province'),
(10, 'Olivia', 'Adams', 'olivia@example.com', '765 Fir St, Territory');
SELECT * FROM customers;
```

100 %

Results | Messages

	customer_id	first_name	last_name	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	3	Robert	Johnson	robert@example.com	789 Oak St, Village
4	4	Sarah	Brown	sarah@example.com	101 Pine St, Suburb
5	5	David	Lee	david@example.com	234 Cedar St, District
6	6	Laura	Hall	laura@example.com	567 Birch St, County
7	7	Michael	Davis	michael@example.com	890 Maple St, State
8	8	Emma	Wilson	emma@example.com	321 Redwood St, Country
9	9	William	Taylor	william@example.com	432 Spruce St, Province
10	10	Olivia	Adams	olivia@example.com	765 Fir St, Territory

STEP 9: Inserting values to the products table as per the given example.



coding challenge a...NTHINI\nisha (60))

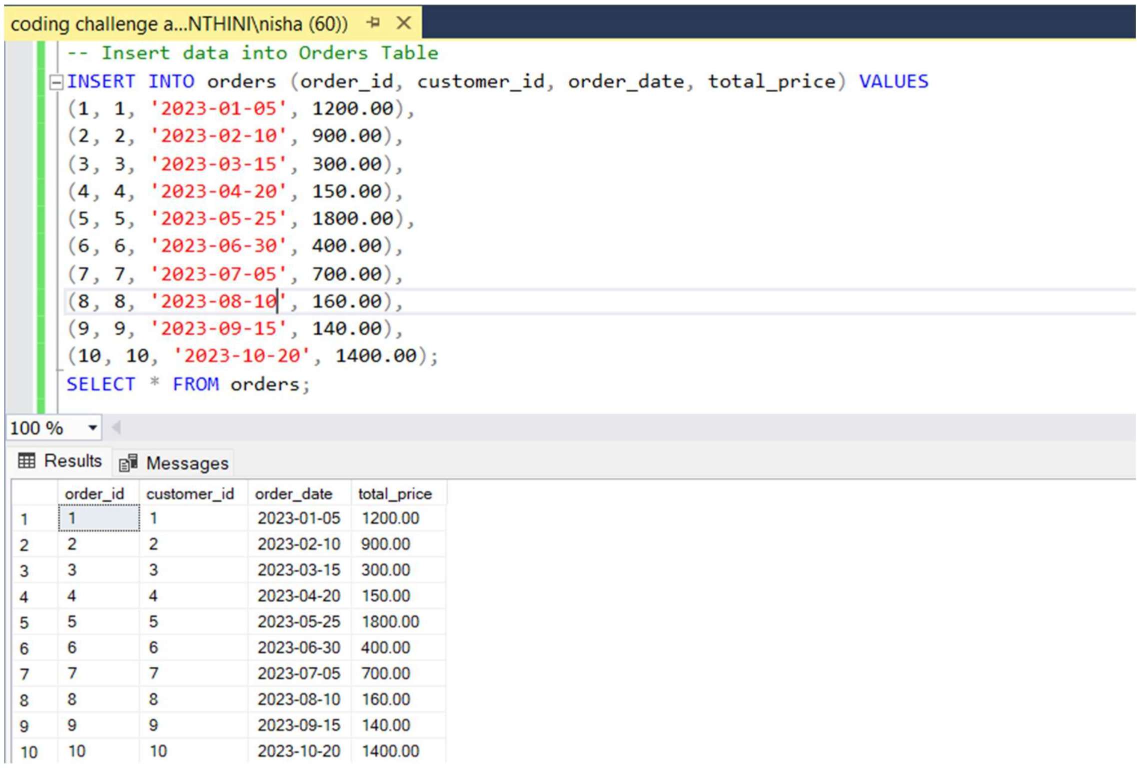
```sql
-- Insert data into Products Table
INSERT INTO products (product_id, name, description, price, stock_quantity) VALUES
(1, 'Laptop', 'High-performance laptop', 800.00, 10),
(2, 'Smartphone', 'Latest smartphone', 600.00, 15),
(3, 'Tablet', 'Portable tablet', 300.00, 20),
(4, 'Headphones', 'Noise-canceling', 150.00, 30),
(5, 'TV', '4K Smart TV', 900.00, 5),
(6, 'Coffee Maker', 'Automatic coffee maker', 50.00, 25),
(7, 'Refrigerator', 'Energy-efficient', 700.00, 10),
(8, 'Microwave Oven', 'Countertop microwave', 80.00, 15),
(9, 'Blender', 'High-speed blender', 70.00, 20),
(10, 'Vacuum Cleaner', 'Bagless vacuum cleaner', 120.00, 10);
SELECT * FROM products;
```

100 %

Results | Messages

	product_id	name	description	price	stock_quantity
1	1	Laptop	High-performance laptop	800.00	10
2	2	Smartphone	Latest smartphone	600.00	15
3	3	Tablet	Portable tablet	300.00	20
4	4	Headphones	Noise-canceling	150.00	30
5	5	TV	4K Smart TV	900.00	5
6	6	Coffee Maker	Automatic coffee maker	50.00	25
7	7	Refrigerator	Energy-efficient	700.00	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

STEP 10: Inserting values to the orders table as per the given example.



coding challenge a...NTHINI\nisha (60))

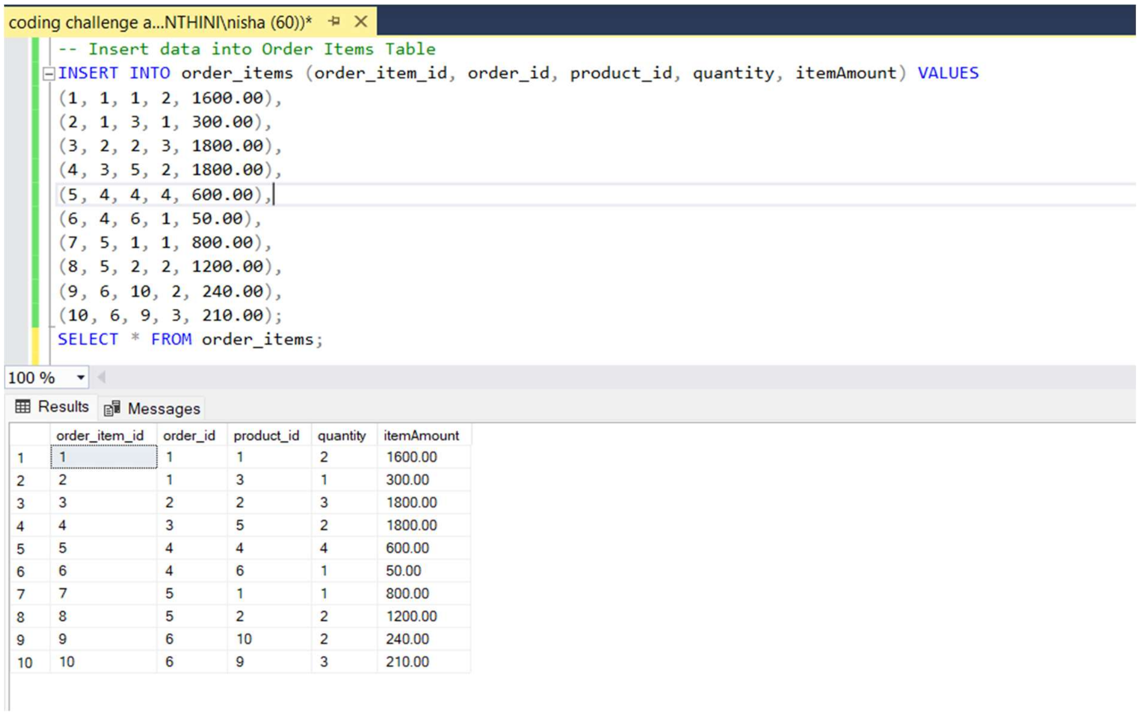
```sql
-- Insert data into Orders Table
INSERT INTO orders (order_id, customer_id, order_date, total_price) VALUES
(1, 1, '2023-01-05', 1200.00),
(2, 2, '2023-02-10', 900.00),
(3, 3, '2023-03-15', 300.00),
(4, 4, '2023-04-20', 150.00),
(5, 5, '2023-05-25', 1800.00),
(6, 6, '2023-06-30', 400.00),
(7, 7, '2023-07-05', 700.00),
(8, 8, '2023-08-10', 160.00),
(9, 9, '2023-09-15', 140.00),
(10, 10, '2023-10-20', 1400.00);
SELECT * FROM orders;
```

100 %

Results | Messages

	order_id	customer_id	order_date	total_price
1	1	1	2023-01-05	1200.00
2	2	2	2023-02-10	900.00
3	3	3	2023-03-15	300.00
4	4	4	2023-04-20	150.00
5	5	5	2023-05-25	1800.00
6	6	6	2023-06-30	400.00
7	7	7	2023-07-05	700.00
8	8	8	2023-08-10	160.00
9	9	9	2023-09-15	140.00
10	10	10	2023-10-20	1400.00

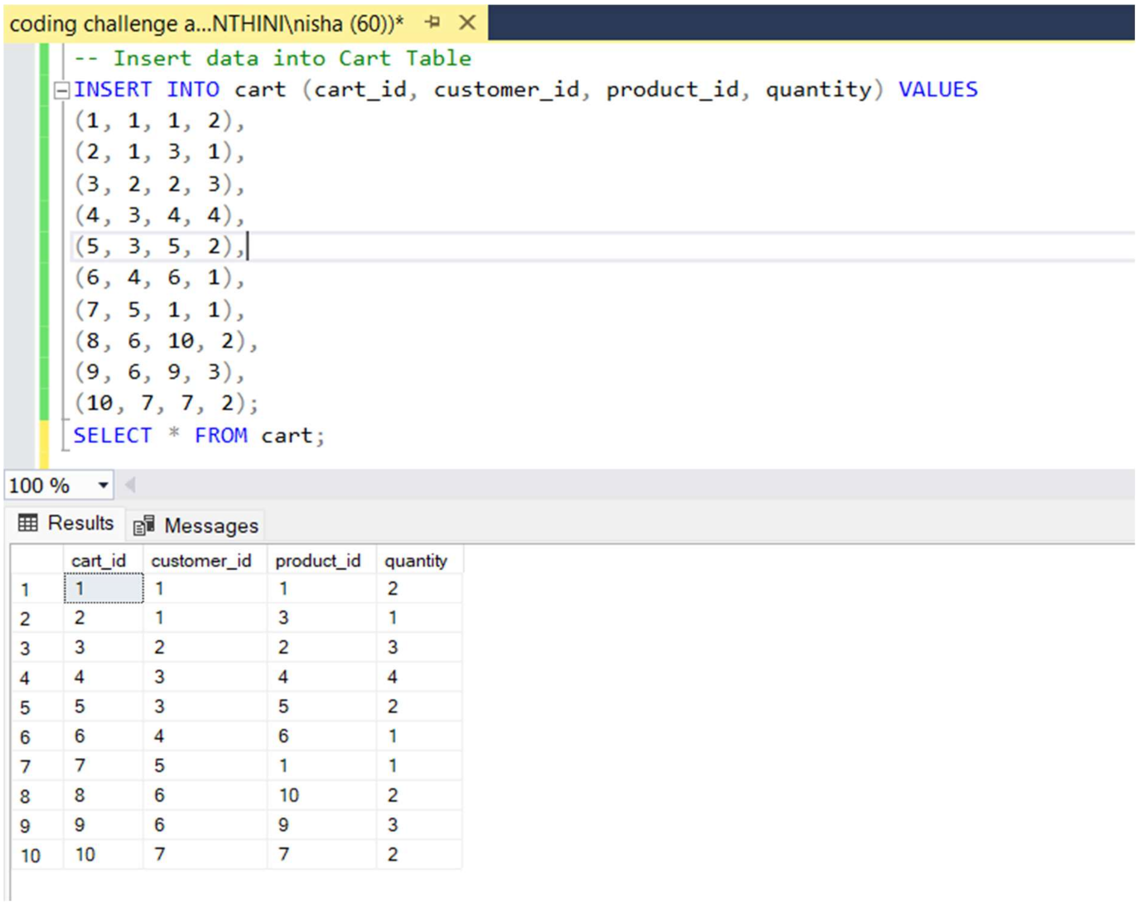
STEP 11: Inserting values to the order items table as per the given example.



```
-- Insert data into Order Items Table
INSERT INTO order_items (order_item_id, order_id, product_id, quantity, itemAmount) VALUES
(1, 1, 1, 2, 1600.00),
(2, 1, 3, 1, 300.00),
(3, 2, 2, 3, 1800.00),
(4, 3, 5, 2, 1800.00),
(5, 4, 4, 4, 600.00),
(6, 4, 6, 1, 50.00),
(7, 5, 1, 1, 800.00),
(8, 5, 2, 2, 1200.00),
(9, 6, 10, 2, 240.00),
(10, 6, 9, 3, 210.00);
SELECT * FROM order_items;
```

	order_item_id	order_id	product_id	quantity	itemAmount
1	1	1	1	2	1600.00
2	2	1	3	1	300.00
3	3	2	2	3	1800.00
4	4	3	5	2	1800.00
5	5	4	4	4	600.00
6	6	4	6	1	50.00
7	7	5	1	1	800.00
8	8	5	2	2	1200.00
9	9	6	10	2	240.00
10	10	6	9	3	210.00

STEP 12: Inserting values to the cart table as per the given example.

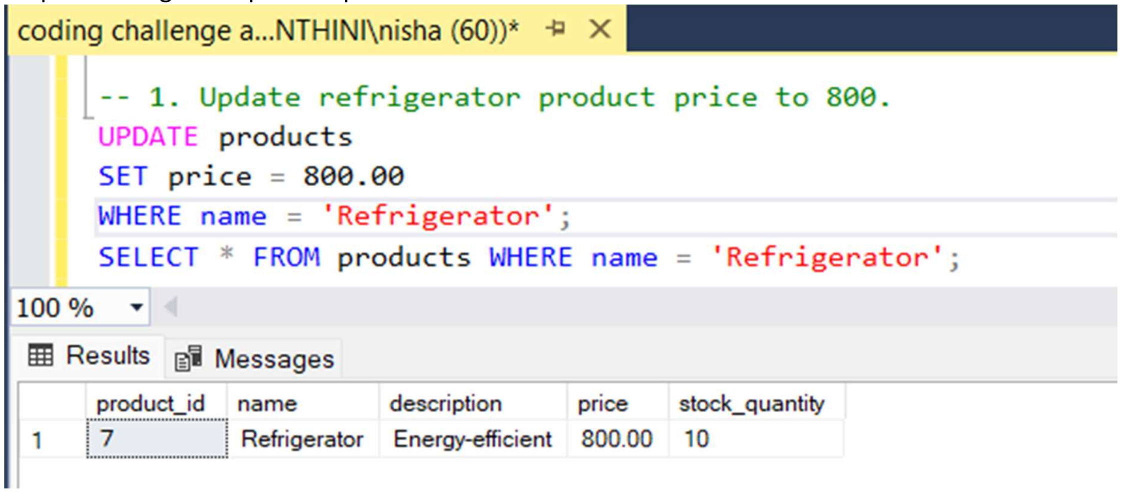


```
-- Insert data into Cart Table
INSERT INTO cart (cart_id, customer_id, product_id, quantity) VALUES
(1, 1, 1, 2),
(2, 1, 3, 1),
(3, 2, 2, 3),
(4, 3, 4, 4),
(5, 3, 5, 2),
(6, 4, 6, 1),
(7, 5, 1, 1),
(8, 6, 10, 2),
(9, 6, 9, 3),
(10, 7, 7, 2);
SELECT * FROM cart;
```

	cart_id	customer_id	product_id	quantity
1	1	1	1	2
2	2	1	3	1
3	3	2	2	3
4	4	3	4	4
5	5	3	5	2
6	6	4	6	1
7	7	5	1	1
8	8	6	10	2
9	9	6	9	3
10	10	7	7	2

TASKS AND QUERIES:

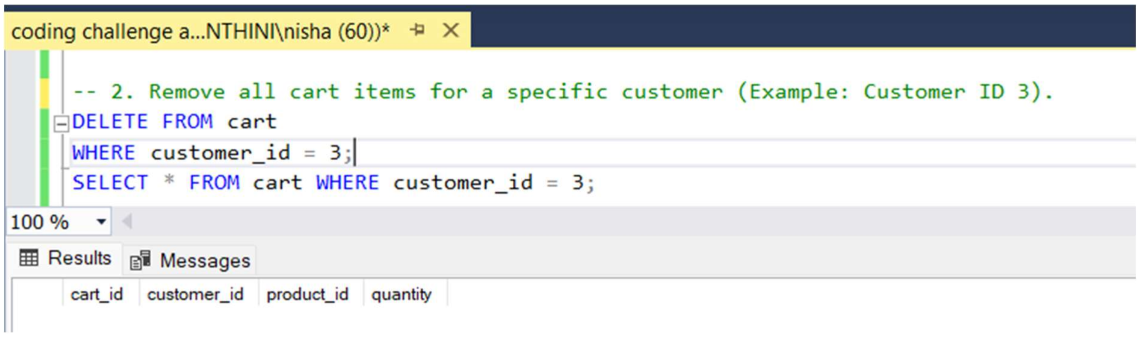
1. Update refrigerator product price to 800.



```
-- 1. Update refrigerator product price to 800.
UPDATE products
SET price = 800.00
WHERE name = 'Refrigerator';
SELECT * FROM products WHERE name = 'Refrigerator';
```

	product_id	name	description	price	stock_quantity
1	7	Refrigerator	Energy-efficient	800.00	10

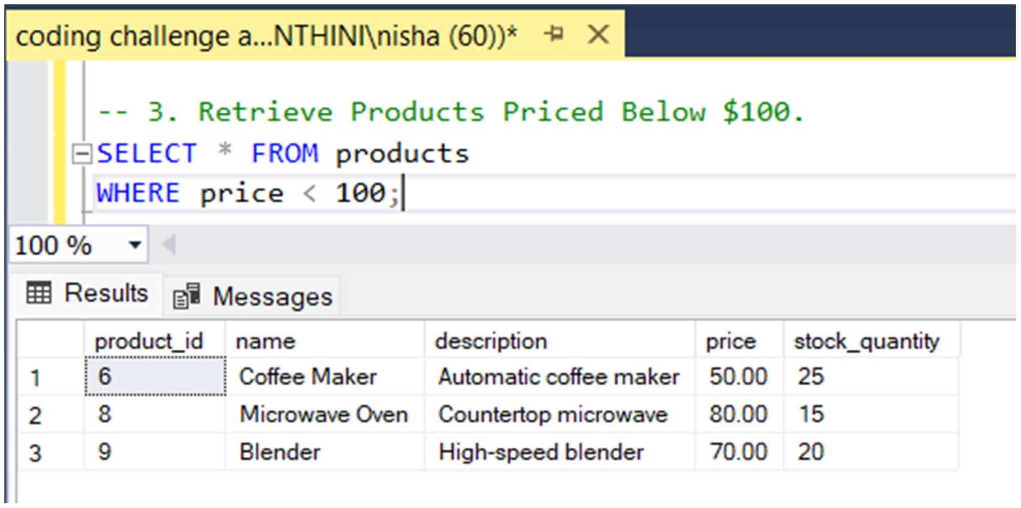
2. Remove all cart items for a specific customer.



```
-- 2. Remove all cart items for a specific customer (Example: Customer ID 3).
DELETE FROM cart
WHERE customer_id = 3;
SELECT * FROM cart WHERE customer_id = 3;
```

cart_id	customer_id	product_id	quantity

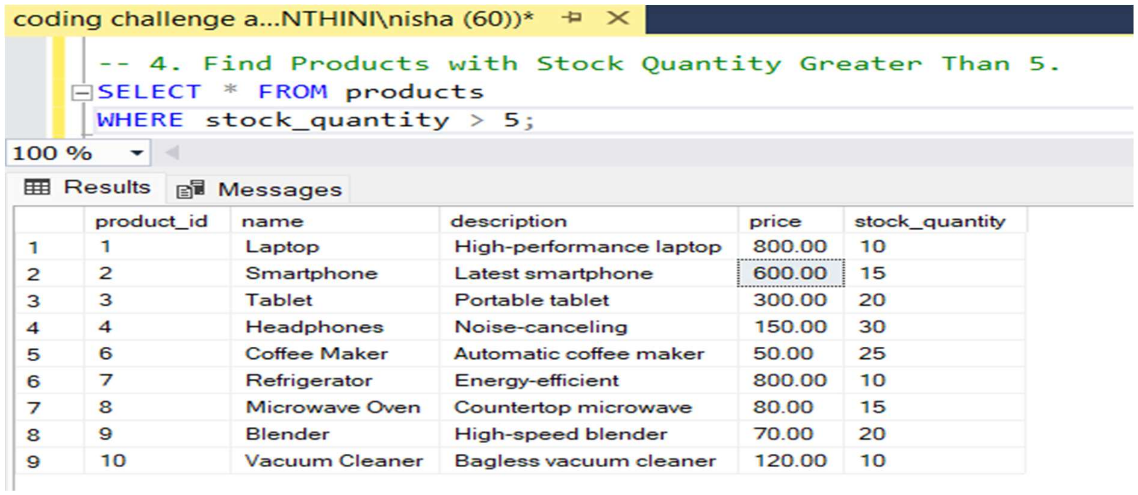
3. Retrieve Products Priced Below $100.



```
-- 3. Retrieve Products Priced Below $100.
SELECT * FROM products
WHERE price < 100;
```

	product_id	name	description	price	stock_quantity
1	6	Coffee Maker	Automatic coffee maker	50.00	25
2	8	Microwave Oven	Countertop microwave	80.00	15
3	9	Blender	High-speed blender	70.00	20

4. Find Products with Stock Quantity Greater Than 5.



coding challenge a...NTHINI\nisha (60))*

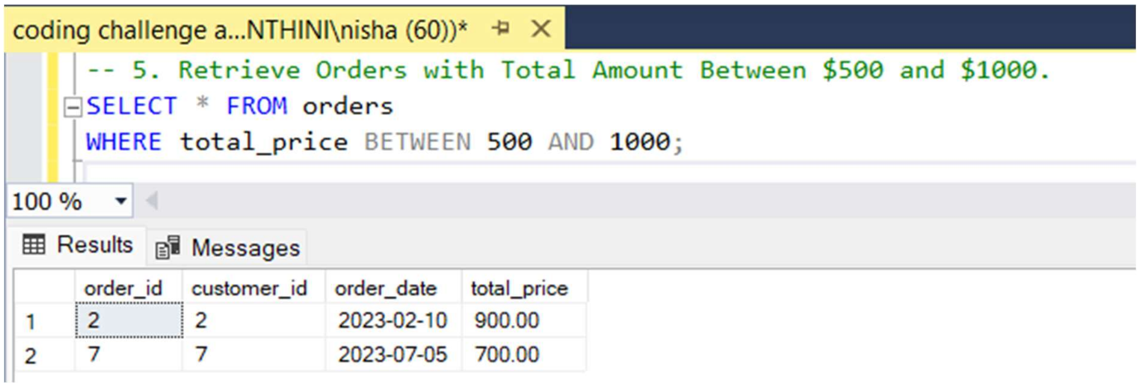
```
-- 4. Find Products with Stock Quantity Greater Than 5.
SELECT * FROM products
WHERE stock_quantity > 5;
```

100 %

Results Messages

	product_id	name	description	price	stock_quantity
1	1	Laptop	High-performance laptop	800.00	10
2	2	Smartphone	Latest smartphone	600.00	15
3	3	Tablet	Portable tablet	300.00	20
4	4	Headphones	Noise-canceling	150.00	30
5	6	Coffee Maker	Automatic coffee maker	50.00	25
6	7	Refrigerator	Energy-efficient	800.00	10
7	8	Microwave Oven	Countertop microwave	80.00	15
8	9	Blender	High-speed blender	70.00	20
9	10	Vacuum Cleaner	Bagless vacuum cleaner	120.00	10

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



coding challenge a...NTHINI\nisha (60))*

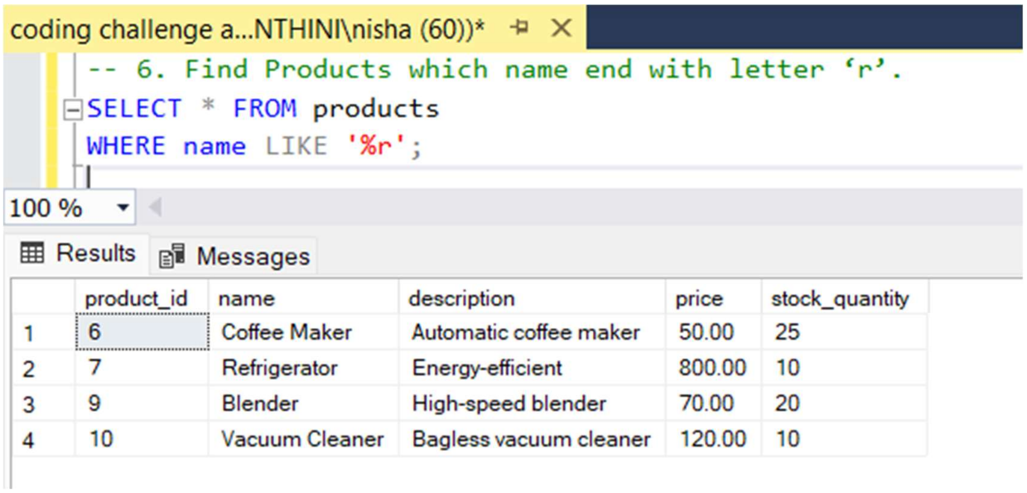
```
-- 5. Retrieve Orders with Total Amount Between $500 and $1000.
SELECT * FROM orders
WHERE total_price BETWEEN 500 AND 1000;
```

100 %

Results Messages

	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'.



coding challenge a...NTHINI\nisha (60))*

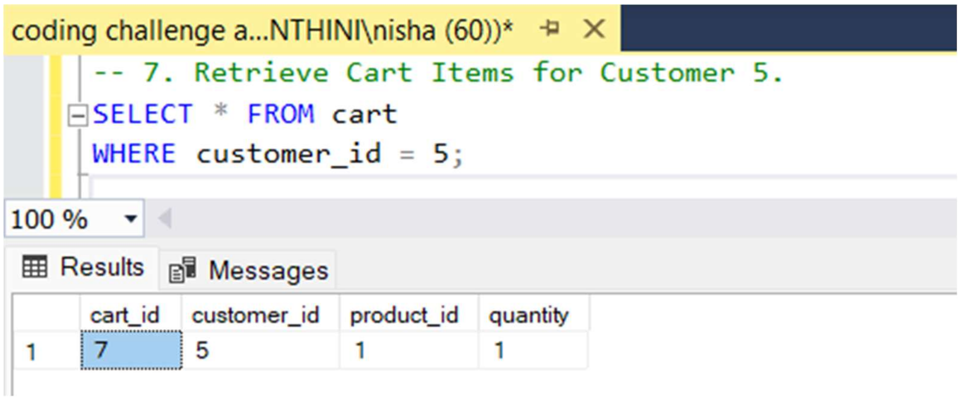
```
-- 6. Find Products which name end with letter 'r'.
SELECT * FROM products
WHERE name LIKE '%r';
```

100 %

Results Messages

	product_id	name	description	price	stock_quantity
1	6	Coffee Maker	Automatic coffee maker	50.00	25
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.


coding challenge a...NTHINI\nisha (60))*

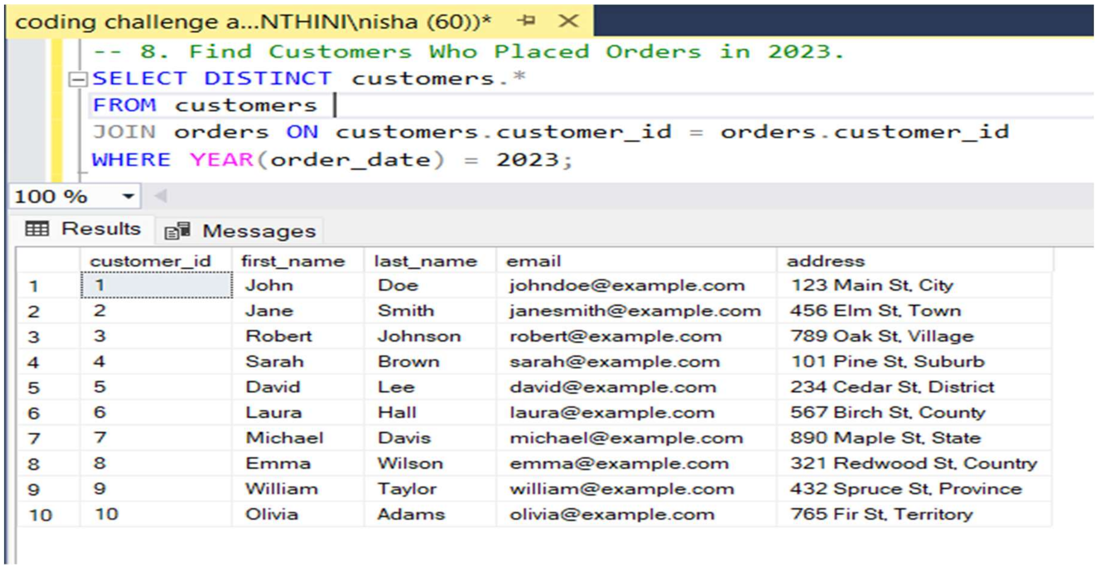
```
-- 7. Retrieve Cart Items for Customer 5.
SELECT * FROM cart
WHERE customer_id = 5;
```

100 %

Results Messages

	cart_id	customer_id	product_id	quantity
1	7	5	1	1

8. Find Customers Who Placed Orders in 2023.


coding challenge a...NTHINI\nisha (60))*

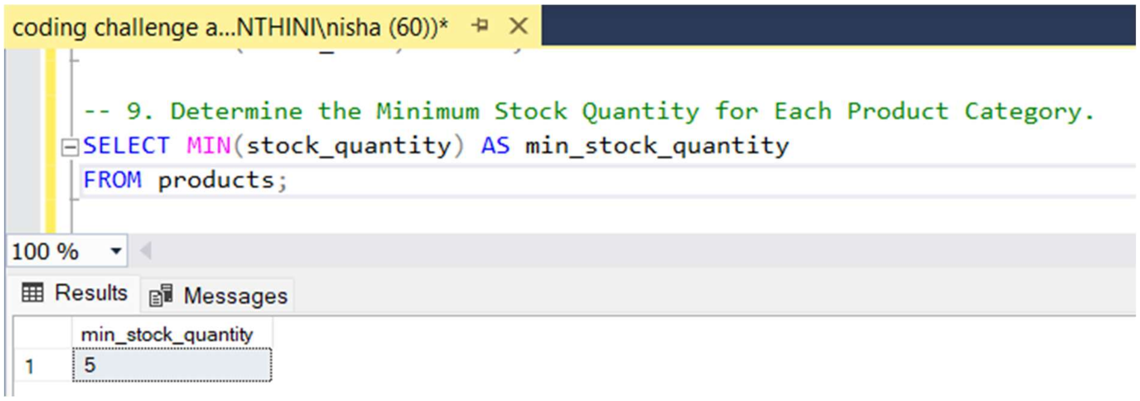
```
-- 8. Find Customers Who Placed Orders in 2023.
SELECT DISTINCT customers.*
FROM customers
JOIN orders ON customers.customer_id = orders.customer_id
WHERE YEAR(order_date) = 2023;
```

100 %

Results Messages

	customer_id	first_name	last_name	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	3	Robert	Johnson	robert@example.com	789 Oak St, Village
4	4	Sarah	Brown	sarah@example.com	101 Pine St, Suburb
5	5	David	Lee	david@example.com	234 Cedar St, District
6	6	Laura	Hall	laura@example.com	567 Birch St, County
7	7	Michael	Davis	michael@example.com	890 Maple St, State
8	8	Emma	Wilson	emma@example.com	321 Redwood St, Country
9	9	William	Taylor	william@example.com	432 Spruce St, Province
10	10	Olivia	Adams	olivia@example.com	765 Fir St, Territory

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


coding challenge a...NTHINI\nisha (60))*

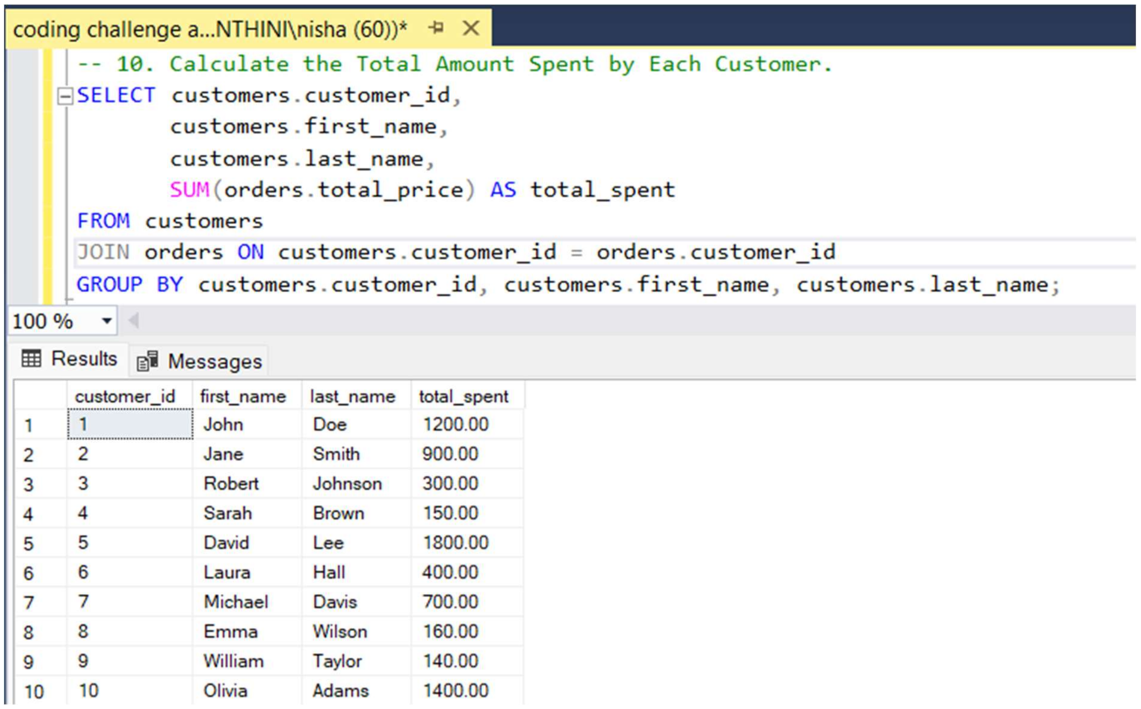
```
-- 9. Determine the Minimum Stock Quantity for Each Product Category.
SELECT MIN(stock_quantity) AS min_stock_quantity
FROM products;
```

100 %

Results Messages

	min_stock_quantity
1	5

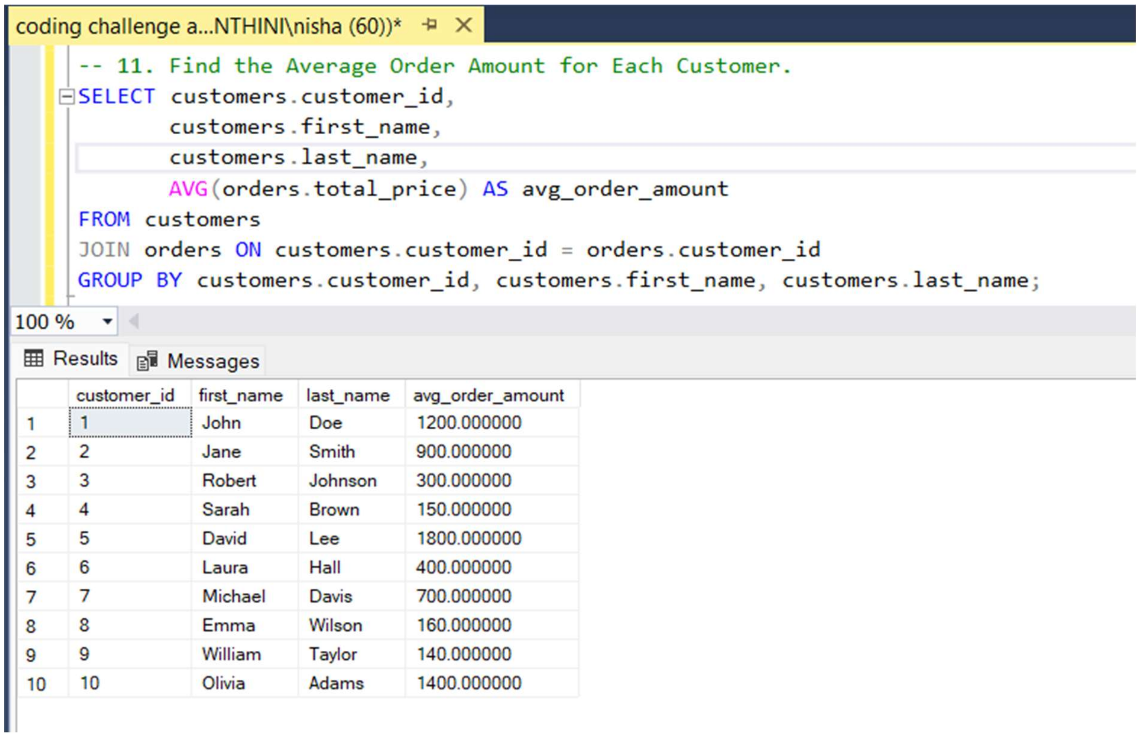
10. Calculate the Total Amount Spent by Each Customer.



```sql
-- 10. Calculate the Total Amount Spent by Each Customer.
SELECT customers.customer_id,
       customers.first_name,
       customers.last_name,
       SUM(orders.total_price) AS total_spent
FROM customers
JOIN orders ON customers.customer_id = orders.customer_id
GROUP BY customers.customer_id, customers.first_name, customers.last_name;
```

	customer_id	first_name	last_name	total_spent
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

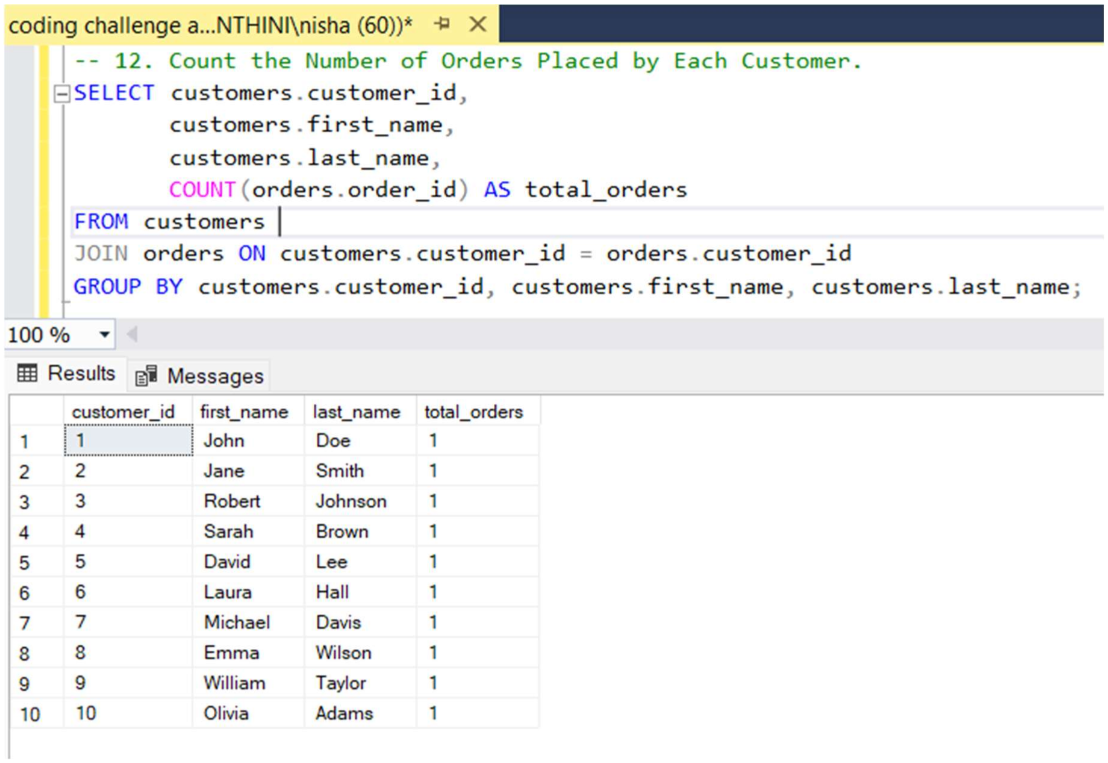
11. Find the Average Order Amount for Each Customer.



```sql
-- 11. Find the Average Order Amount for Each Customer.
SELECT customers.customer_id,
       customers.first_name,
       customers.last_name,
       AVG(orders.total_price) AS avg_order_amount
FROM customers
JOIN orders ON customers.customer_id = orders.customer_id
GROUP BY customers.customer_id, customers.first_name, customers.last_name;
```

	customer_id	first_name	last_name	avg_order_amount
1	1	John	Doe	1200.000000
2	2	Jane	Smith	900.000000
3	3	Robert	Johnson	300.000000
4	4	Sarah	Brown	150.000000
5	5	David	Lee	1800.000000
6	6	Laura	Hall	400.000000
7	7	Michael	Davis	700.000000
8	8	Emma	Wilson	160.000000
9	9	William	Taylor	140.000000
10	10	Olivia	Adams	1400.000000

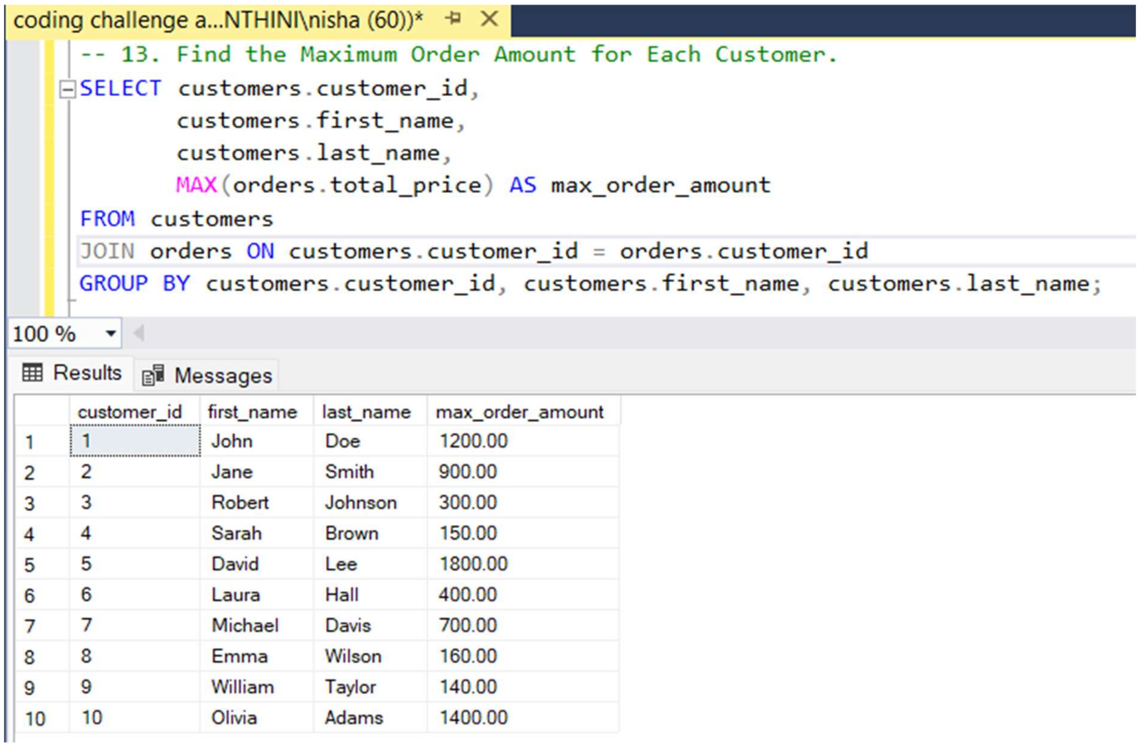
12. Count the Number of Orders Placed by Each Customer.



```sql
-- 12. Count the Number of Orders Placed by Each Customer.
SELECT customers.customer_id,
       customers.first_name,
       customers.last_name,
       COUNT(orders.order_id) AS total_orders
FROM customers
JOIN orders ON customers.customer_id = orders.customer_id
GROUP BY customers.customer_id, customers.first_name, customers.last_name;
```

	customer_id	first_name	last_name	total_orders
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

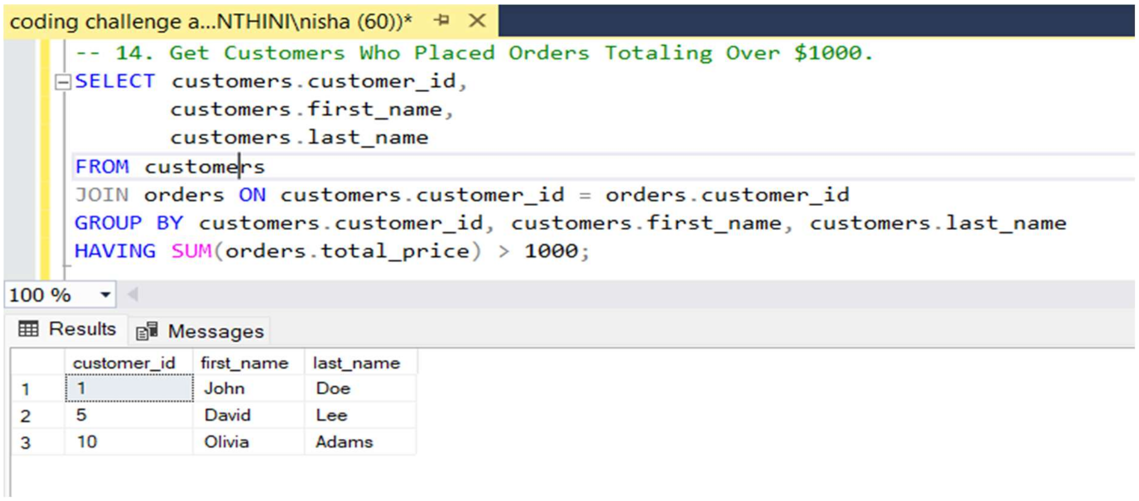
13. Find the Maximum Order Amount for Each Customer.



```sql
-- 13. Find the Maximum Order Amount for Each Customer.
SELECT customers.customer_id,
       customers.first_name,
       customers.last_name,
       MAX(orders.total_price) AS max_order_amount
FROM customers
JOIN orders ON customers.customer_id = orders.customer_id
GROUP BY customers.customer_id, customers.first_name, customers.last_name;
```

	customer_id	first_name	last_name	max_order_amount
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

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



coding challenge a...NTHINI\nisha (60))*

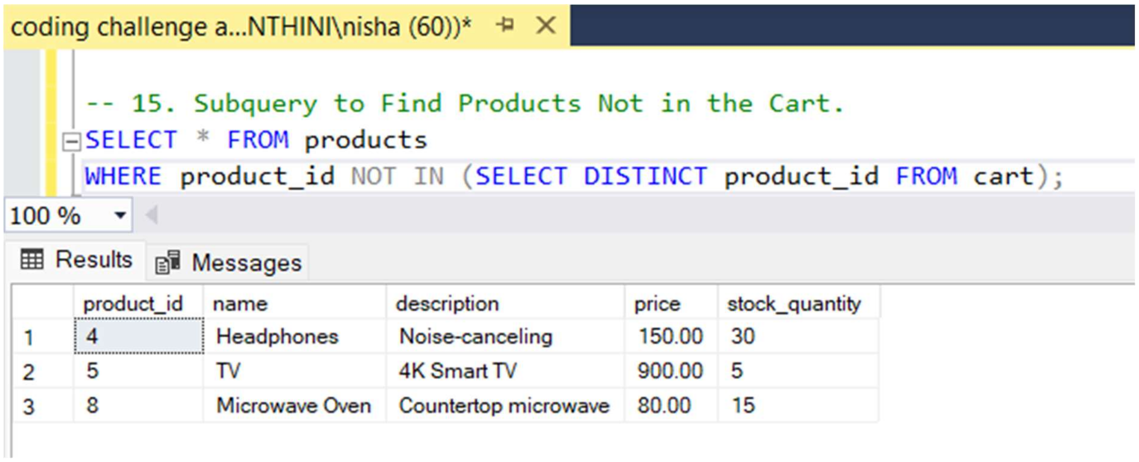
```
-- 14. Get Customers Who Placed Orders Totaling Over $1000.
SELECT customers.customer_id,
       customers.first_name,
       customers.last_name
FROM customers
JOIN orders ON customers.customer_id = orders.customer_id
GROUP BY customers.customer_id, customers.first_name, customers.last_name
HAVING SUM(orders.total_price) > 1000;
```

100 %

Results Messages

	customer_id	first_name	last_name
1	1	John	Doe
2	5	David	Lee
3	10	Olivia	Adams

15. Subquery to Find Products Not in the Cart.



coding challenge a...NTHINI\nisha (60))*

```
-- 15. Subquery to Find Products Not in the Cart.
SELECT * FROM products
WHERE product_id NOT IN (SELECT DISTINCT product_id FROM cart);
```

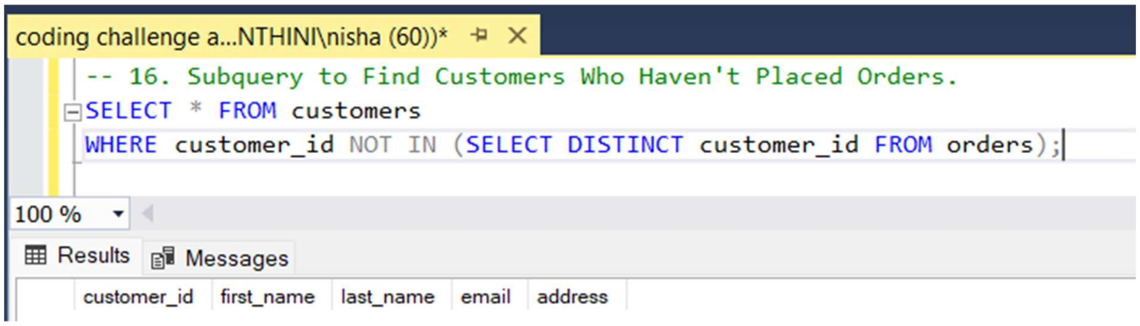
100 %

Results Messages

	product_id	name	description	price	stock_quantity
1	4	Headphones	Noise-canceling	150.00	30
2	5	TV	4K Smart TV	900.00	5
3	8	Microwave Oven	Countertop microwave	80.00	15

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

Case a: Since all the customers have placed order with respect to the example, no rows are returned as a result.



coding challenge a...NTHINI\nisha (60))*

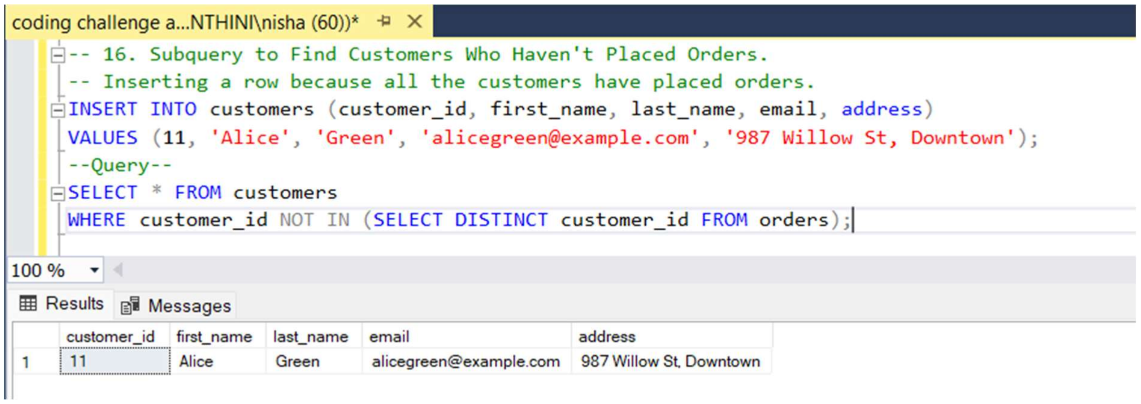
```
-- 16. Subquery to Find Customers Who Haven't Placed Orders.
SELECT * FROM customers
WHERE customer_id NOT IN (SELECT DISTINCT customer_id FROM orders);
```

100 %

Results Messages

customer_id	first_name	last_name	email	address

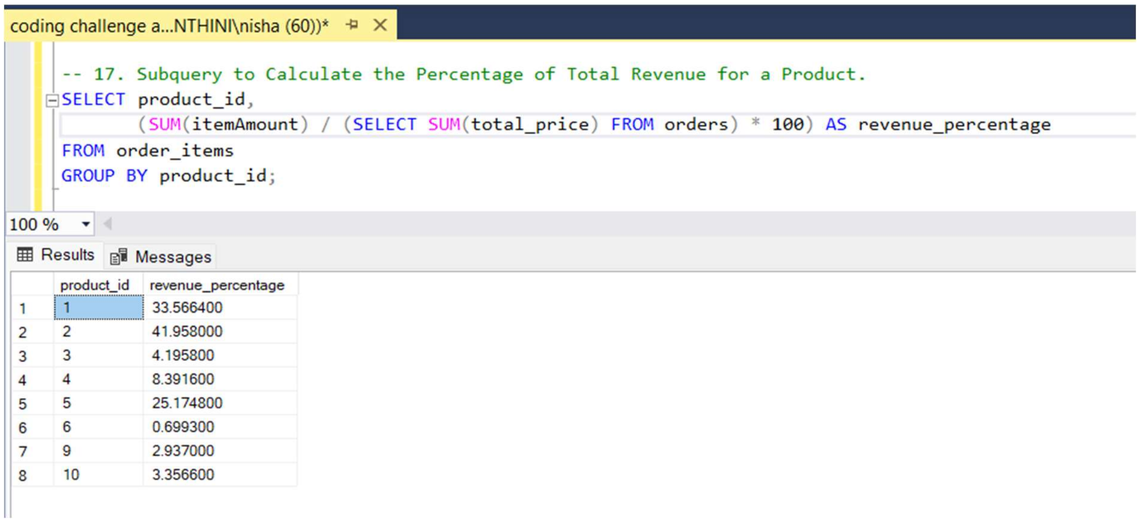
Case b: To avoid the occurence of 'case a', a new row is inserted to the table customers.



```sql
-- 16. Subquery to Find Customers Who Haven't Placed Orders.
-- Inserting a row because all the customers have placed orders.
INSERT INTO customers (customer_id, first_name, last_name, email, address)
VALUES (11, 'Alice', 'Green', 'alicegreen@example.com', '987 Willow St, Downtown');
--Query--
SELECT * FROM customers
WHERE customer_id NOT IN (SELECT DISTINCT customer_id FROM orders);
```

	customer_id	first_name	last_name	email	address
1	11	Alice	Green	alicegreen@example.com	987 Willow St, Downtown

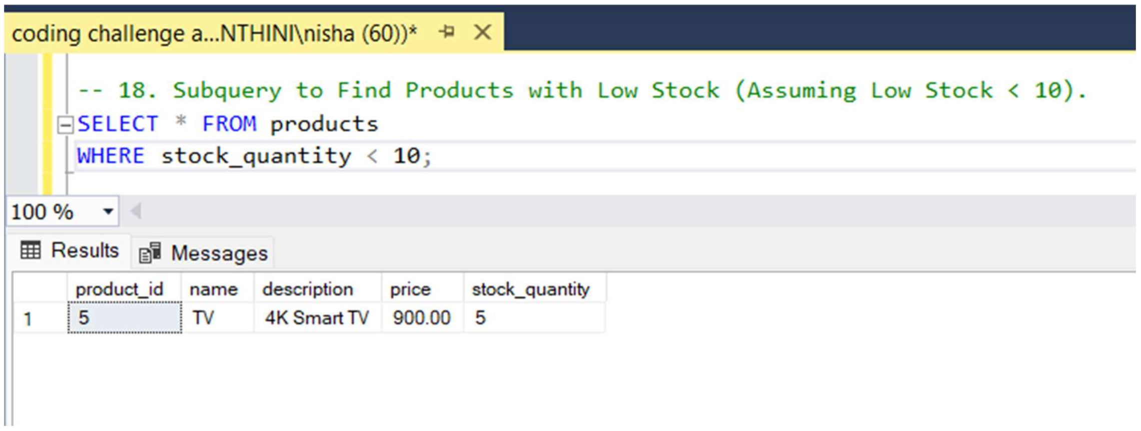
17. Subquery to Calculate the Percentage of Total Revenue for a Product.



```sql
-- 17. Subquery to Calculate the Percentage of Total Revenue for a Product.
SELECT product_id,
       (SUM(itemAmount) / (SELECT SUM(total_price) FROM orders) * 100) AS revenue_percentage
FROM order_items
GROUP BY product_id;
```

	product_id	revenue_percentage
1	1	33.566400
2	2	41.958000
3	3	4.195800
4	4	8.391600
5	5	25.174800
6	6	0.699300
7	9	2.937000
8	10	3.356600

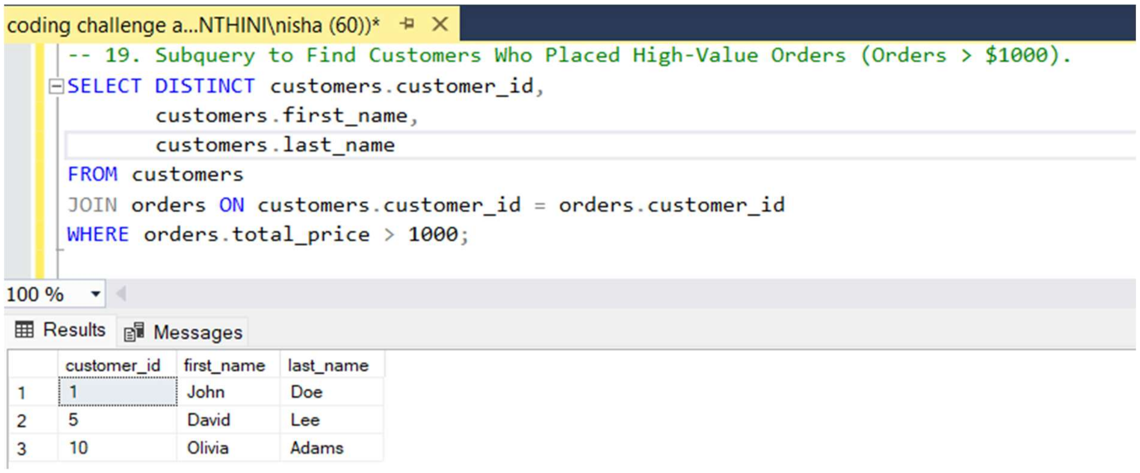
18. Subquery to Find Products with Low Stock.



```sql
-- 18. Subquery to Find Products with Low Stock (Assuming Low Stock < 10).
SELECT * FROM products
WHERE stock_quantity < 10;
```

	product_id	name	description	price	stock_quantity
1	5	TV	4K Smart TV	900.00	5

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



```
-- 19. Subquery to Find Customers Who Placed High-Value Orders (Orders > $1000).
SELECT DISTINCT customers.customer_id,
       customers.first_name,
       customers.last_name
FROM customers
JOIN orders ON customers.customer_id = orders.customer_id
WHERE orders.total_price > 1000;
```

	customer_id	first_name	last_name
1	1	John	Doe
2	5	David	Lee
3	10	Olivia	Adams