

## SQL CODING CHALLENGE – ECOMMERCE

### a) Creating a Database to store the Ecom tables:

#### Query:

```
create database ecom;
```

```
use ecom;
```

#### Output:

✓	2	11:48:57	create database ecom	1 row(s) affected
✓	3	11:49:16	use ecom	0 row(s) affected

### b) Creation of SQL Tables:

1. **customers table:** • customer\_id (Primary Key) • name • email • Address.

#### Query:

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

2. **products table:** • product\_id (Primary Key) • name • price • description • stockQuantity.

#### Query:

```
CREATE TABLE products (  
    product_id INT PRIMARY KEY,  
    name VARCHAR(100),
```

```

    description TEXT,
    price DECIMAL(10,2),
    stockQuantity INT
);

```

3. **cart table:** • cart\_id (Primary Key) • customer\_id (Foreign Key) • product\_id (Foreign Key) • quantity.

**Query:**

```

CREATE TABLE cart (
    cart_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT,
    product_id INT,
    quantity INT,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON
DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES products(product_id) ON
DELETE CASCADE
);

```

4. **orders table:** • order\_id (Primary Key) • customer\_id (Foreign Key) • order\_date • total\_price • shipping\_address.

**Query:**

```

CREATE TABLE orders (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT,
    order_date DATE,
    total_price DECIMAL(10,2),
    shipping_address VARCHAR(255),
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON
DELETE CASCADE);

```

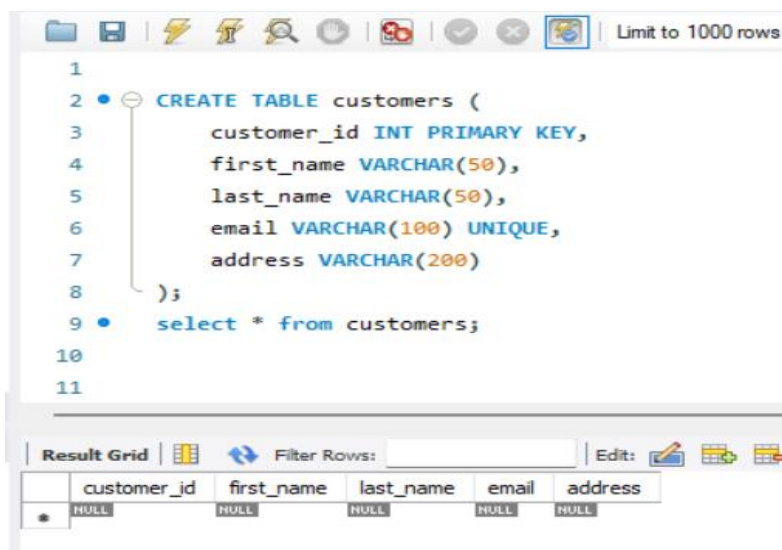
**5. order\_items table** (to store order details): • order\_item\_id (Primary Key) • order\_id (Foreign Key) • product\_id (Foreign Key) • quantity • item\_amount.

### Query:

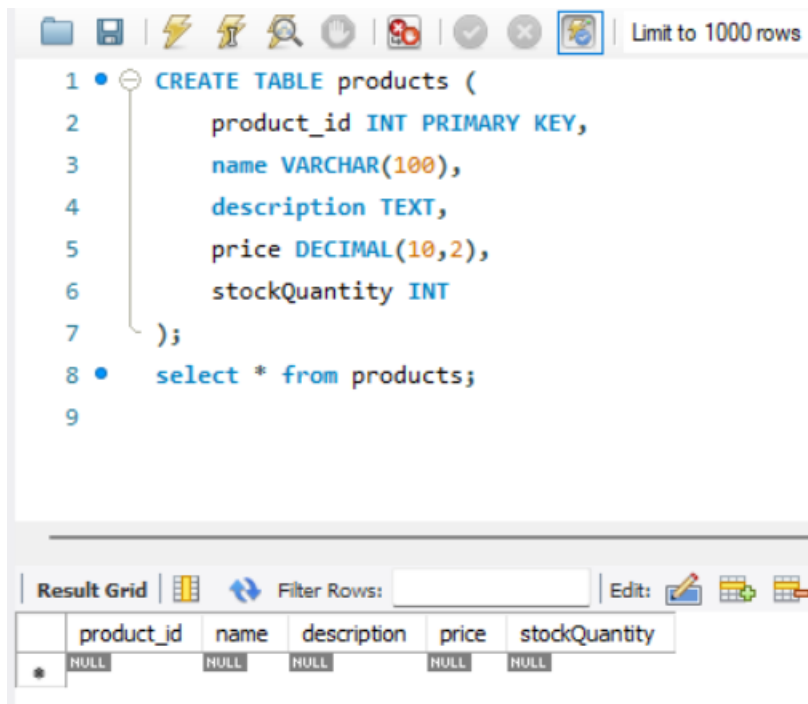
```
CREATE TABLE order_items (
    order_item_id INT PRIMARY KEY AUTO_INCREMENT,
    order_id INT,
    product_id INT,
    quantity INT,
    item_amount DECIMAL(10,2),
    FOREIGN KEY (order_id) REFERENCES orders(order_id) ON DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE CASCADE
);
```

### Outputs:

1. *Select \* from customers;*



## 2. *Select \* from products;*



The screenshot shows a database IDE with a toolbar at the top. The SQL editor contains the following code:

```

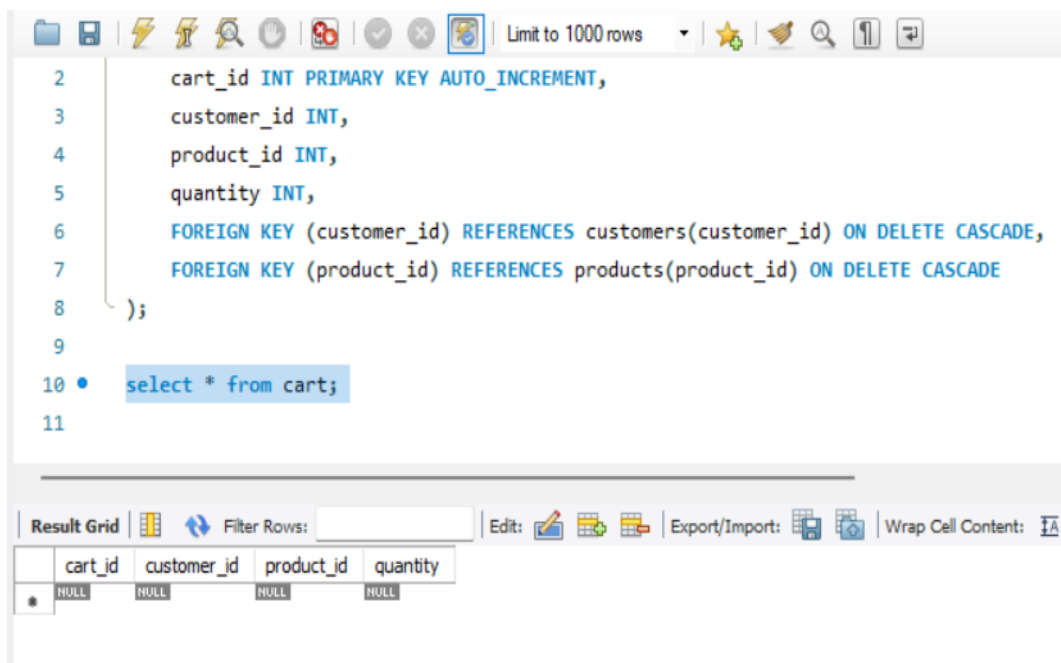
1 • CREATE TABLE products (
2     product_id INT PRIMARY KEY,
3     name VARCHAR(100),
4     description TEXT,
5     price DECIMAL(10,2),
6     stockQuantity INT
7 );
8 • select * from products;
9

```

Below the editor is a 'Result Grid' section. It includes a 'Filter Rows' input field and an 'Edit' button. The grid shows the following columns: product\_id, name, description, price, and stockQuantity. The first row contains NULL values for all columns.

	product_id	name	description	price	stockQuantity
*	NULL	NULL	NULL	NULL	NULL

## 3. *Select \* from cart;*



The screenshot shows a database IDE with a toolbar at the top. The SQL editor contains the following code:

```

2     cart_id INT PRIMARY KEY AUTO_INCREMENT,
3     customer_id INT,
4     product_id INT,
5     quantity INT,
6     FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE,
7     FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE CASCADE
8 );
9
10 • select * from cart;
11

```

Below the editor is a 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Edit' button, an 'Export/Import' button, and a 'Wrap Cell Content' checkbox. The grid shows the following columns: cart\_id, customer\_id, product\_id, and quantity. The first row contains NULL values for all columns.

	cart_id	customer_id	product_id	quantity
*	NULL	NULL	NULL	NULL

#### 4. *Select \* from orders;*

```

1 • CREATE TABLE orders (
2     order_id INT PRIMARY KEY AUTO_INCREMENT,
3     customer_id INT,
4     order_date DATE,
5     total_price DECIMAL(10,2),
6     shipping_address VARCHAR(255),
7     FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE
8 );
9
10
11 • select * from orders;
12

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

	order_id	customer_id	order_date	total_price	shipping_address
*	NULL	NULL	NULL	NULL	NULL

#### 5. *Select \* from order\_items;*

Limit to 1000 rows

```

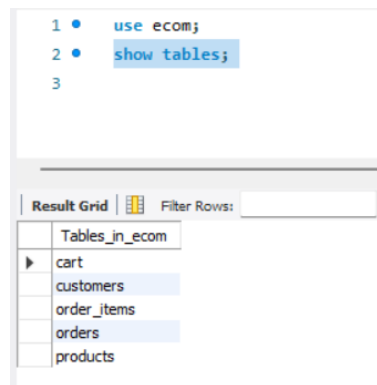
1 • CREATE TABLE order_items (
2     order_item_id INT PRIMARY KEY AUTO_INCREMENT,
3     order_id INT,
4     product_id INT,
5     quantity INT,
6     item_amount DECIMAL(10,2),
7     FOREIGN KEY (order_id) REFERENCES orders(order_id) ON DELETE CASCADE,
8     FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE CASCADE
9 );
10
11 • select * from order_items;
12

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

	order_item_id	order_id	product_id	quantity	item_amount
*	NULL	NULL	NULL	NULL	NULL

## Tables in Ecom Database:



## c) Inserting Records into the Tables:

### 1. Insert data into Customers Table -

#### Query:

```
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');
```

#### Output:

```

1 • INSERT INTO customers (customer_id, first_name, last_name, email, address) VALUES
2   (1, 'John', 'Doe', 'johndoe@example.com', '123 Main St, City'),
3   (2, 'Jane', 'Smith', 'janesmith@example.com', '456 Elm St, Town'),
4   (3, 'Robert', 'Johnson', 'robert@example.com', '789 Oak St, Village'),
5   (4, 'Sarah', 'Brown', 'sarah@example.com', '101 Pine St, Suburb'),
6   (5, 'David', 'Lee', 'david@example.com', '234 Cedar St, District'),
7   (6, 'Laura', 'Hall', 'laura@example.com', '567 Birch St, County'),
8   (7, 'Michael', 'Davis', 'michael@example.com', '890 Maple St, State'),
9   (8, 'Emma', 'Wilson', 'emma@example.com', '321 Redwood St, Country'),
10  (9, 'William', 'Taylor', 'william@example.com', '432 Spruce St, Province'),
11  (10, 'Olivia', 'Adams', 'olivia@example.com', '765 Fir St, Territory');
12
13 • Select * from customers;

```

Result Grid					
Filter Rows: <input type="text"/>					
Edit:   					
Export/Import:  					
Wrap Cell Content: 					
customer_id	first_name	last_name	email	address	
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	
NULL	NULL	NULL	NULL	NULL	

## 2. Insert data into products table –

### Query:

*INSERT INTO products (product\_id, name, description, price, stockQuantity)*  
*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);

### Output:

```

1
2 • INSERT INTO products (product_id, name, description, price, stockQuantity) VALUES
3   (1, 'Laptop', 'High-performance laptop', 800.00, 10),
4   (2, 'Smartphone', 'Latest smartphone', 600.00, 15),
5   (3, 'Tablet', 'Portable tablet', 300.00, 20),
6   (4, 'Headphones', 'Noise-canceling', 150.00, 30),
7   (5, 'TV', '4K Smart TV', 900.00, 5),
8   (6, 'Coffee Maker', 'Automatic coffee maker', 50.00, 25),
9   (7, 'Refrigerator', 'Energy-efficient', 700.00, 10),
10  (8, 'Microwave Oven', 'Countertop microwave', 80.00, 15),
11  (9, 'Blender', 'High-speed blender', 70.00, 20),
12  (10, 'Vacuum Cleaner', 'Bagless vacuum cleaner', 120.00, 10);
13

```

product_id	name	description	price	stockQuantity
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
NULL	NULL	NULL	NULL	NULL

### 3. Insert data into cart table –

#### Query:

*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);

### Output:



The screenshot shows a database query editor with a toolbar at the top. The query text is as follows:

```

1 • INSERT INTO cart (cart_id, customer_id, product_id, quantity) VALUES
2   (1, 1, 1, 2),(2, 1, 3, 1),(3, 2, 2, 3),(4, 3, 4, 4),(5, 3, 5, 2),
3   (6, 4, 6, 1),(7, 5, 1, 1),(8, 6, 10, 2),(9, 6, 9, 3),(10, 7, 7, 2);
4
5
6
7 • Select * from cart;

```

Below the query editor is the 'Result Grid' section, which displays the results of the query. The grid has columns for cart\_id, customer\_id, product\_id, and quantity. The results are as follows:

	cart_id	customer_id	product_id	quantity
▶	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
*	NULL	NULL	NULL	NULL

#### 4. Insert data into orders table –

##### Query:

*INSERT INTO orders (order\_id, customer\_id, order\_date, total\_price, shipping\_address) VALUES*

*(1, 1, '2023-01-05', 1200.00, '123 Main St, City'),*

*(2, 2, '2023-02-10', 900.00, '456 Elm St, Town'),*

*(3, 3, '2023-03-15', 300.00, '789 Oak St, Village'),*

*(4, 4, '2023-04-20', 150.00, '101 Pine St, Suburb'),*

*(5, 5, '2023-05-25', 1800.00, '234 Cedar St, District'),*

*(6, 6, '2023-06-30', 400.00, '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');

### Output:

The screenshot shows a database IDE with the following SQL query in the editor:

```
1 • INSERT INTO orders (order_id, customer_id, order_date, total_price, shipping_address) VALUES
2 (1, 1, '2023-01-05', 1200.00, '123 Main St, City'),
3 (2, 2, '2023-02-10', 900.00, '456 Elm St, Town'),
4 (3, 3, '2023-03-15', 300.00, '789 Oak St, Village'),
5 (4, 4, '2023-04-20', 150.00, '101 Pine St, Suburb'),
6 (5, 5, '2023-05-25', 1800.00, '234 Cedar St, District'),
7 (6, 6, '2023-06-30', 400.00, '567 Birch St, County'),
8 (7, 7, '2023-07-05', 700.00, '890 Maple St, State'),
9 (8, 8, '2023-08-10', 160.00, '321 Redwood St, Country'),
10 (9, 9, '2023-09-15', 140.00, '432 Spruce St, Province'),
11 (10, 10, '2023-10-20', 1400.00, '765 Fir St, Territory');
12
13
14 • Select * from orders;
```

The result grid below the query shows the data inserted into the 'orders' table:

	order_id	customer_id	order_date	total_price	shipping_address
1	1	1	2023-01-05	1200.00	123 Main St, City
2	2	2	2023-02-10	900.00	456 Elm St, Town
3	3	3	2023-03-15	300.00	789 Oak St, Village
4	4	4	2023-04-20	150.00	101 Pine St, Suburb
5	5	5	2023-05-25	1800.00	234 Cedar St, District
6	6	6	2023-06-30	400.00	567 Birch St, County
7	7	7	2023-07-05	700.00	890 Maple St, State
8	8	8	2023-08-10	160.00	321 Redwood St, Country
9	9	9	2023-09-15	140.00	432 Spruce St, Province
10	10	10	2023-10-20	1400.00	765 Fir St, Territory
*	NULL	NULL	NULL	NULL	NULL

### 5. Insert data into order\_items table –

#### Query:

*INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity, item\_amount) 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);

#### Output:

```

1 • INSERT INTO order_items (order_item_id, order_id, product_id, quantity, item_amount)
2   VALUES(1, 1, 1, 2, 1600.00),(2, 1, 3, 1, 300.00),(3, 2, 2, 3, 1800.00),
3   (4, 3, 5, 2, 1800.00),(5, 4, 4, 4, 600.00),(6, 4, 6, 1, 50.00),
4   (7, 5, 1, 1, 800.00),(8, 5, 2, 2, 1200.00),(9, 6, 10, 2, 240.00),
5   (10, 6, 9, 3, 210.00);
6
7
8 • Select * from order_items;

```

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	order_item_id	order_id	product_id	quantity	item_amount
▶	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
✱	NULL	NULL	NULL	NULL	NULL

## d) SQL CHALLENGES:

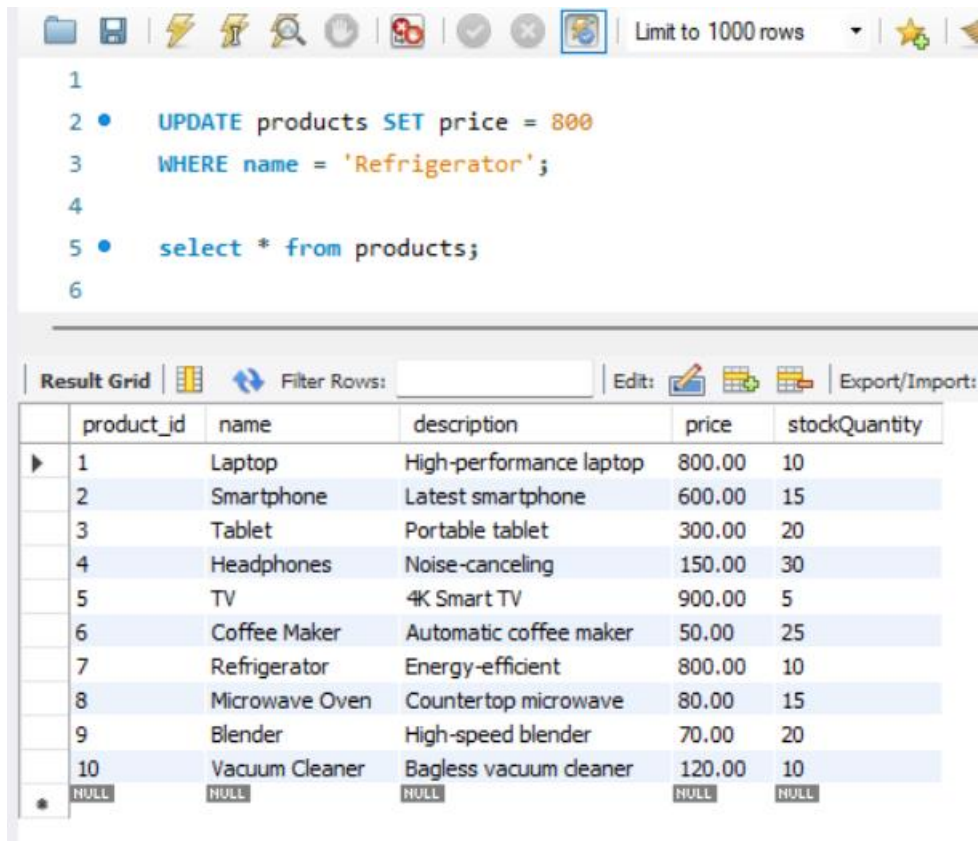
1. Update refrigerator product price to 800.

Query:

*UPDATE products SET price = 800*

*WHERE name = 'Refrigerator';*

Result:



The screenshot shows a database management interface. At the top, there is a toolbar with various icons and a 'Limit to 1000 rows' dropdown. Below the toolbar, a SQL query editor contains the following code:

```

1
2 • UPDATE products SET price = 800
3   WHERE name = 'Refrigerator';
4
5 • select * from products;
6

```

Below the query editor, there is a 'Result Grid' section. It includes a 'Filter Rows:' input field, an 'Edit:' button, and an 'Export/Import:' button. The result grid displays a table with the following data:

	product_id	name	description	price	stockQuantity
▶	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	800.00	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
•	NULL	NULL	NULL	NULL	NULL

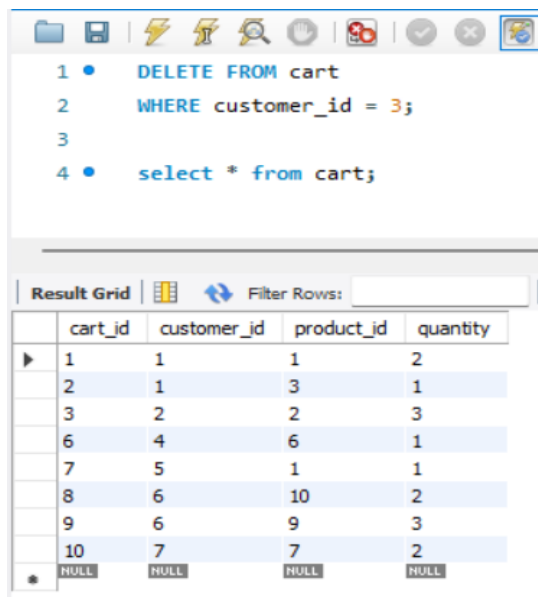
## 2. Remove all cart items for a specific customer. (eg : Customer\_id = 3)

### Query:

*DELETE FROM cart WHERE customer\_id = 3;*

### Result:

**Note:** Removing all cart items for Customer – 3.



1 • DELETE FROM cart  
2 WHERE customer\_id = 3;  
3  
4 • select \* from cart;

Result Grid

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

### 3. Retrieve Products Priced Below \$100.

#### Query:

*Select \* from products WHERE price < 100;*

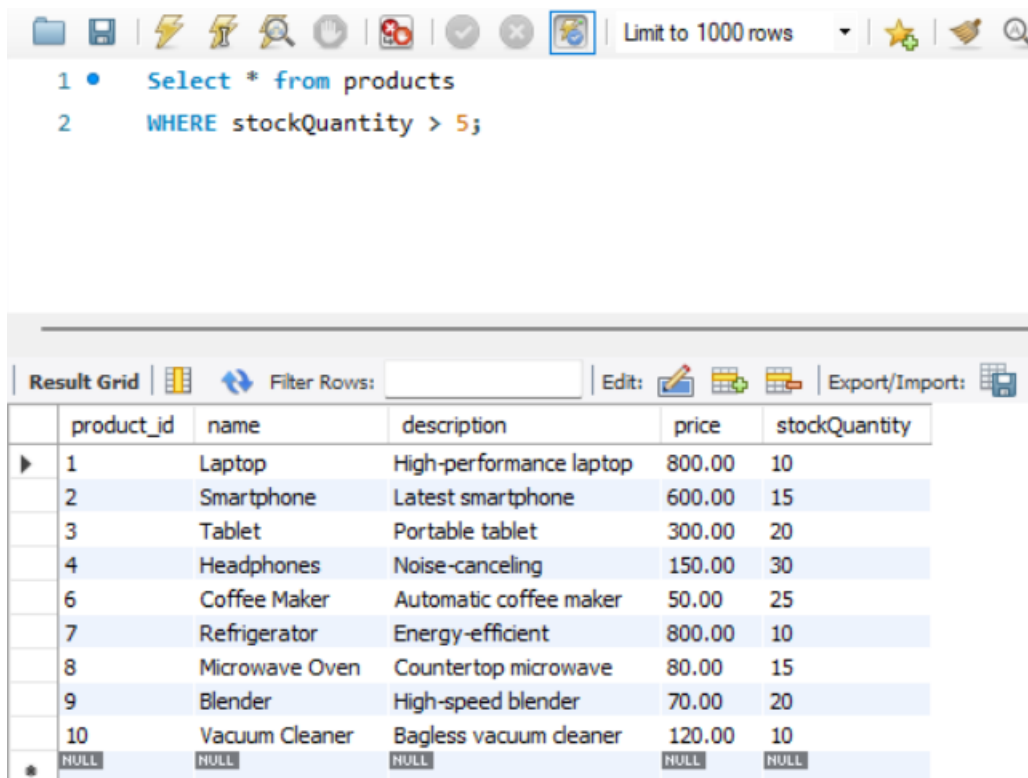
#### Result:

<

### 4. Find Products with Stock Quantity Greater Than 5.

#### Query:

*Select \* from products WHERE stockQuantity > 5;*

**Result:**


Limit to 1000 rows

```

1 • Select * from products
2   WHERE stockQuantity > 5;

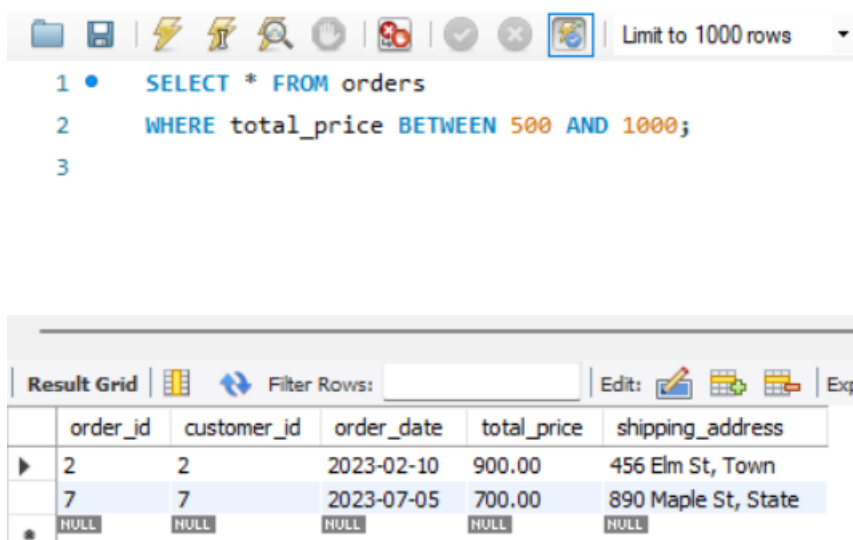
```

	product_id	name	description	price	stockQuantity
▶	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
	6	Coffee Maker	Automatic coffee maker	50.00	25
	7	Refrigerator	Energy-efficient	800.00	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
*	NULL	NULL	NULL	NULL	NULL

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

*SELECT \* FROM orders*

*WHERE total\_price BETWEEN 500 AND 1000;*

**Result:**


Limit to 1000 rows

```

1 • SELECT * FROM orders
2   WHERE total_price BETWEEN 500 AND 1000;
3

```

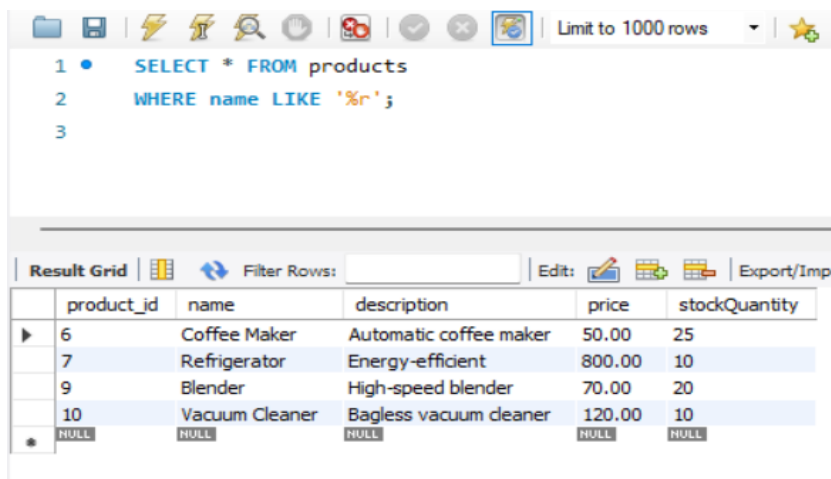
	order_id	customer_id	order_date	total_price	shipping_address
▶	2	2	2023-02-10	900.00	456 Elm St, Town
	7	7	2023-07-05	700.00	890 Maple St, State
*	NULL	NULL	NULL	NULL	NULL

## 6. Find Products which name end with letter 'r'.

### Query:

```
SELECT * FROM products
WHERE name LIKE '%r';
```

### Result:



The screenshot shows a database query tool interface. The query editor displays the following SQL query:

```
1 • SELECT * FROM products
2 WHERE name LIKE '%r';
3
```

The results are shown in a table with the following columns: product\_id, name, description, price, and stockQuantity. The results are as follows:

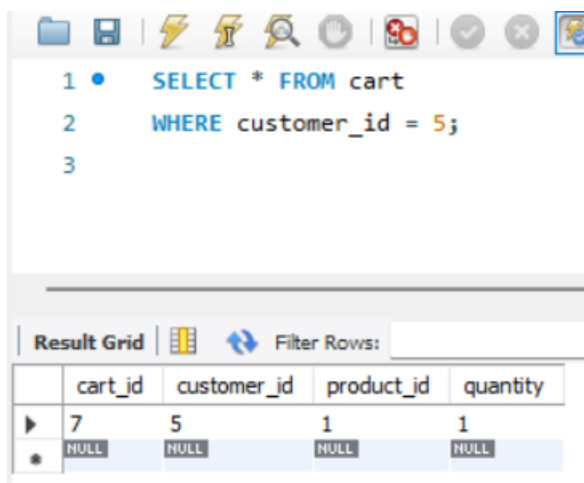
product_id	name	description	price	stockQuantity
6	Coffee Maker	Automatic coffee maker	50.00	25
7	Refrigerator	Energy-efficient	800.00	10
9	Blender	High-speed blender	70.00	20
10	Vacuum Cleaner	Bagless vacuum cleaner	120.00	10
NULL	NULL	NULL	NULL	NULL

## 7. Retrieve Cart Items for Customer 5.

### Query:

```
SELECT * FROM cart WHERE customer_id = 5;
```

### Result:



The screenshot shows a database query tool interface. The query editor displays the following SQL query:

```
1 • SELECT * FROM cart
2 WHERE customer_id = 5;
3
```

The results are shown in a table with the following columns: cart\_id, customer\_id, product\_id, and quantity. The results are as follows:

cart_id	customer_id	product_id	quantity
7	5	1	1
NULL	NULL	NULL	NULL



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

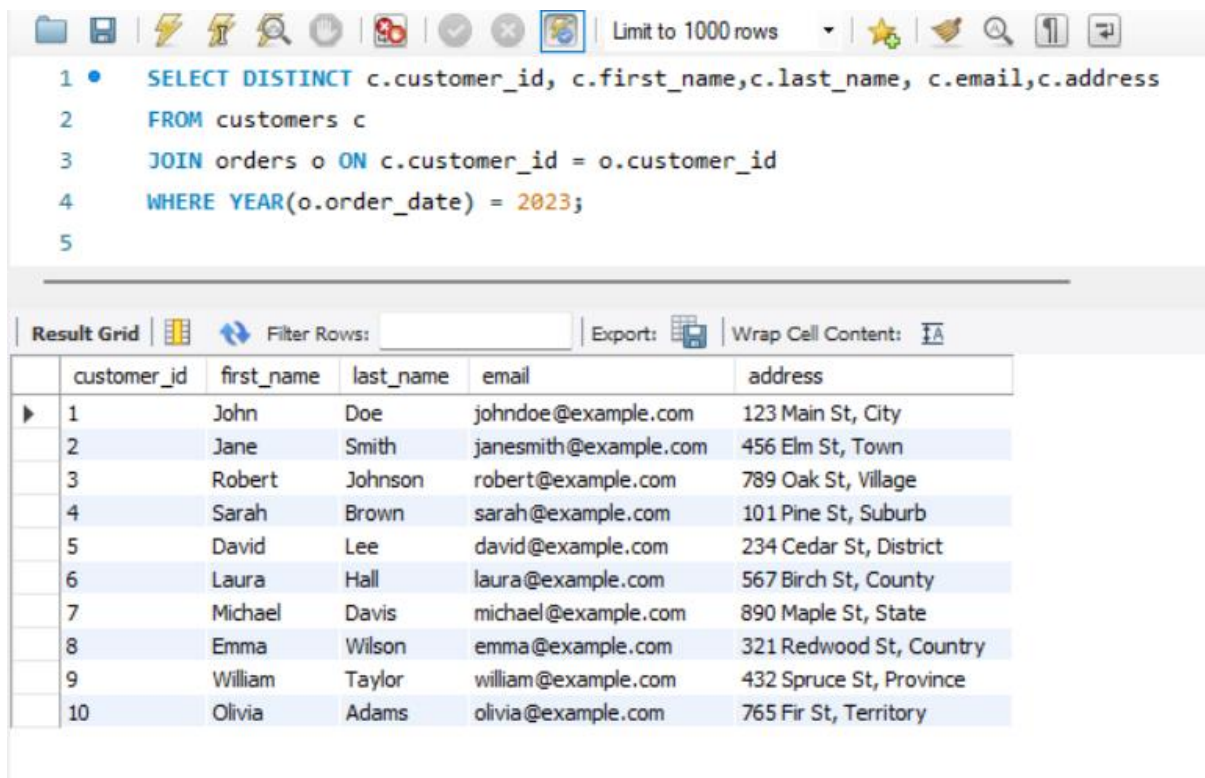
### Query:

```
SELECT DISTINCT c.customer_id, c.first_name, c.last_name, c.email, c.address
FROM customers c
```

```
JOIN orders o ON c.customer_id = o.customer_id
```

```
WHERE YEAR(o.order_date) = 2023;
```

### Result:



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 • SELECT DISTINCT c.customer_id, c.first_name, c.last_name, c.email, c.address
2 FROM customers c
3 JOIN orders o ON c.customer_id = o.customer_id
4 WHERE YEAR(o.order_date) = 2023;
5
```

Below the query editor is a 'Result Grid' showing 10 rows of data. The columns are: customer\_id, first\_name, last\_name, email, and address. The data is as follows:

	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.

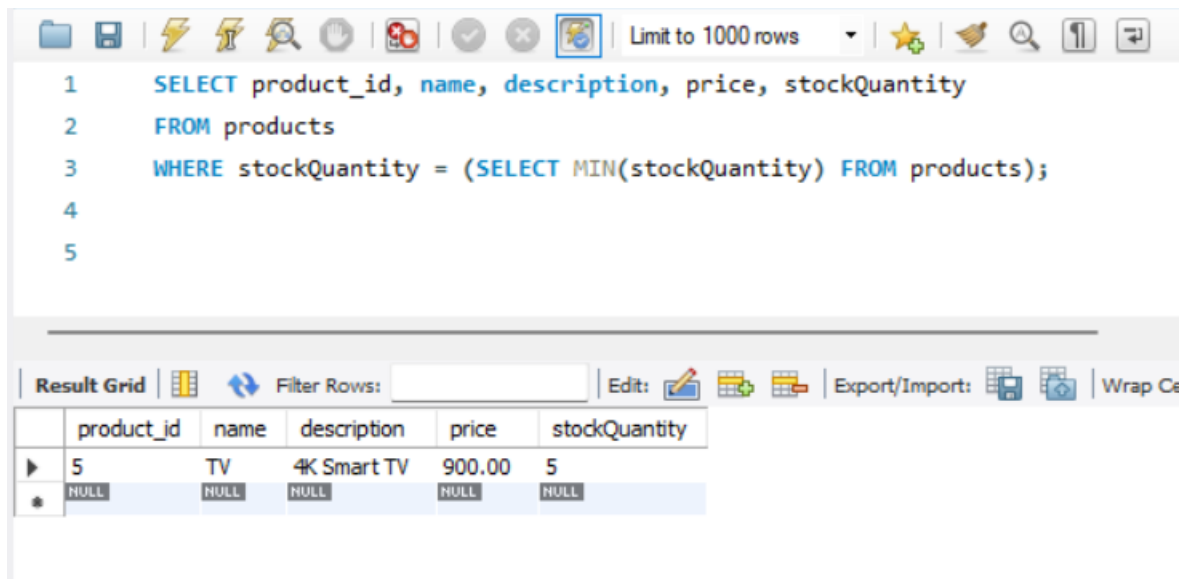
### Query:

```
SELECT product_id, name, description, price, stockQuantity
```

```
FROM products
```

```
WHERE stockQuantity = (SELECT MIN(stockQuantity) FROM products);
```

### Result:



```

1  SELECT product_id, name, description, price, stockQuantity
2  FROM products
3  WHERE stockQuantity = (SELECT MIN(stockQuantity) FROM products);
4
5

```

product_id	name	description	price	stockQuantity
5	TV	4K Smart TV	900.00	5
NULL	NULL	NULL	NULL	NULL

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

#### Query:

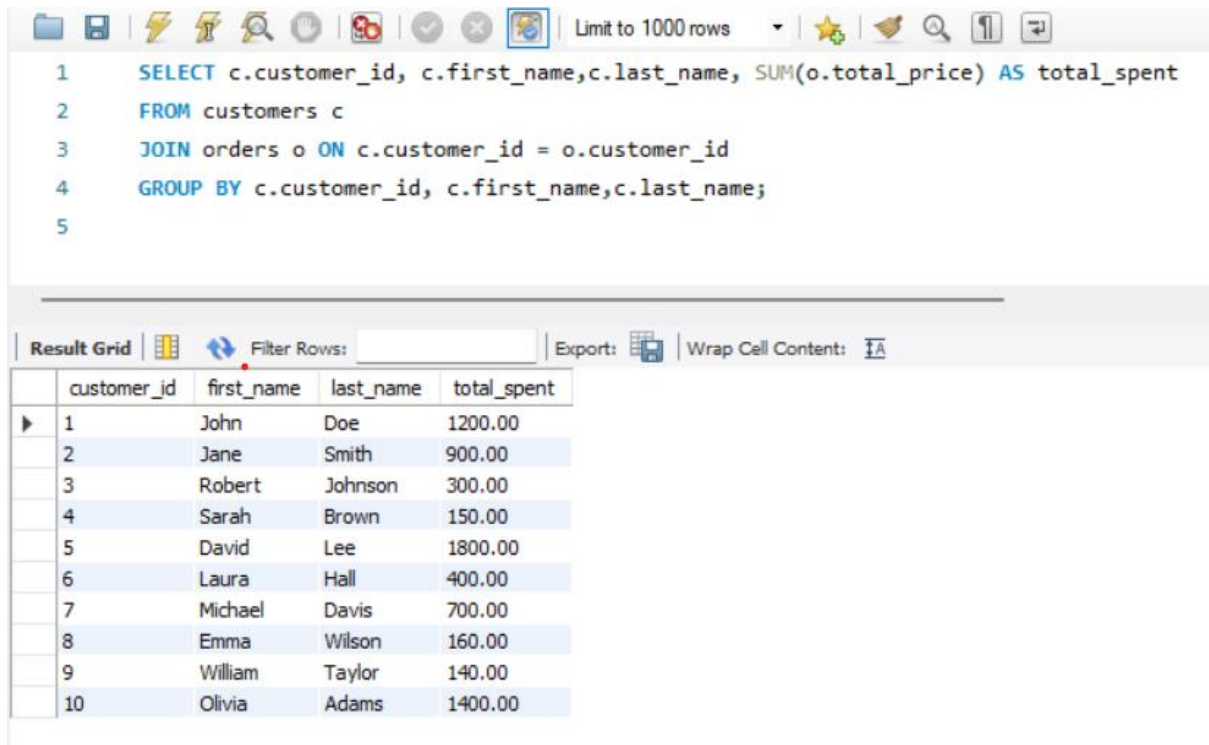
*SELECT c.customer\_id, c.first\_name, c.last\_name, SUM(o.total\_price) AS total\_spent*

*FROM customers c*

*JOIN orders o ON c.customer\_id = o.customer\_id*

*GROUP BY c.customer\_id, , c.first\_name, c.last\_name;*

#### Result:



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```

1  SELECT c.customer_id, c.first_name, c.last_name, SUM(o.total_price) AS total_spent
2  FROM customers c
3  JOIN orders o ON c.customer_id = o.customer_id
4  GROUP BY c.customer_id, c.first_name, c.last_name;
5

```

Below the query editor is the 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The result grid displays the following data:

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

### 11. Find the Average Order Amount for Each Customer.

#### Query:

```

SELECT c.customer_id, c.first_name, c.last_name, AVG(o.total_price) AS
avg_order_amount
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id;

```

#### Result:

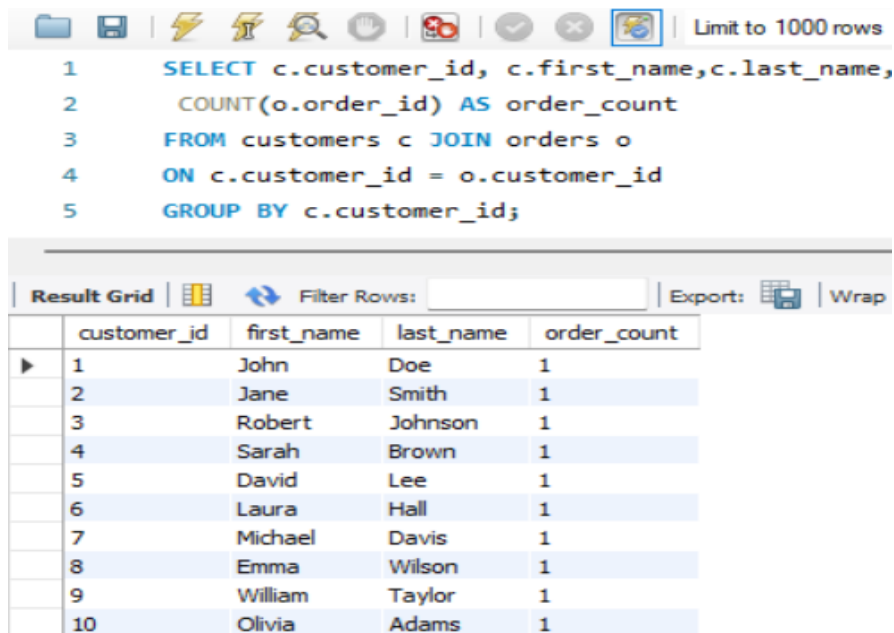
### 12. Count the Number of Orders Placed by Each Customer.

**Query:**

```

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

```

**Result:**


The screenshot shows a SQL query editor with a toolbar at the top. The query is entered in the main text area. Below the query, a 'Result Grid' is displayed, showing the results of the query. The grid has columns for customer\_id, first\_name, last\_name, and order\_count. There are 10 rows of data, each representing a customer with their first and last name and the number of orders they have placed.

	customer_id	first_name	last_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

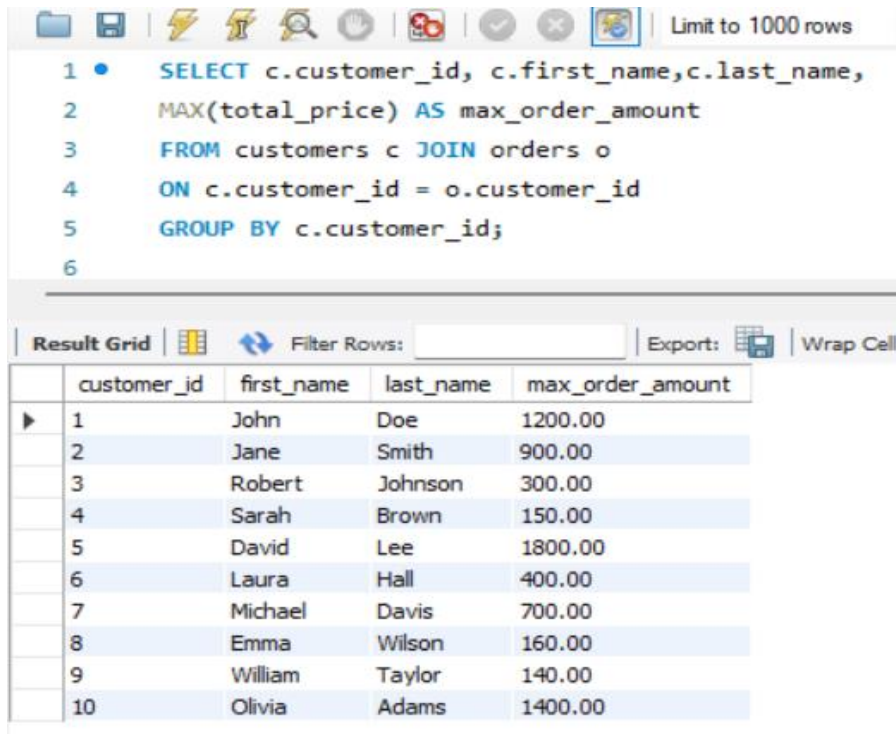
**13. Find the Maximum Order Amount for Each Customer****Query:**

```

SELECT c.customer_id, c.first_name, c.last_name, MAX(total_price) AS
max_order_amount FROM customers c JOIN orders o
ON c.customer_id = o.customer_id GROUP BY c.customer_id;

```

**Result:**



The screenshot shows a database query editor with a toolbar at the top. The SQL query is as follows:

```

1 • SELECT c.customer_id, c.first_name, c.last_name,
2     MAX(total_price) AS max_order_amount
3     FROM customers c JOIN orders o
4     ON c.customer_id = o.customer_id
5     GROUP BY c.customer_id;
6

```

Below the query editor is a 'Result Grid' with the following data:

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

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

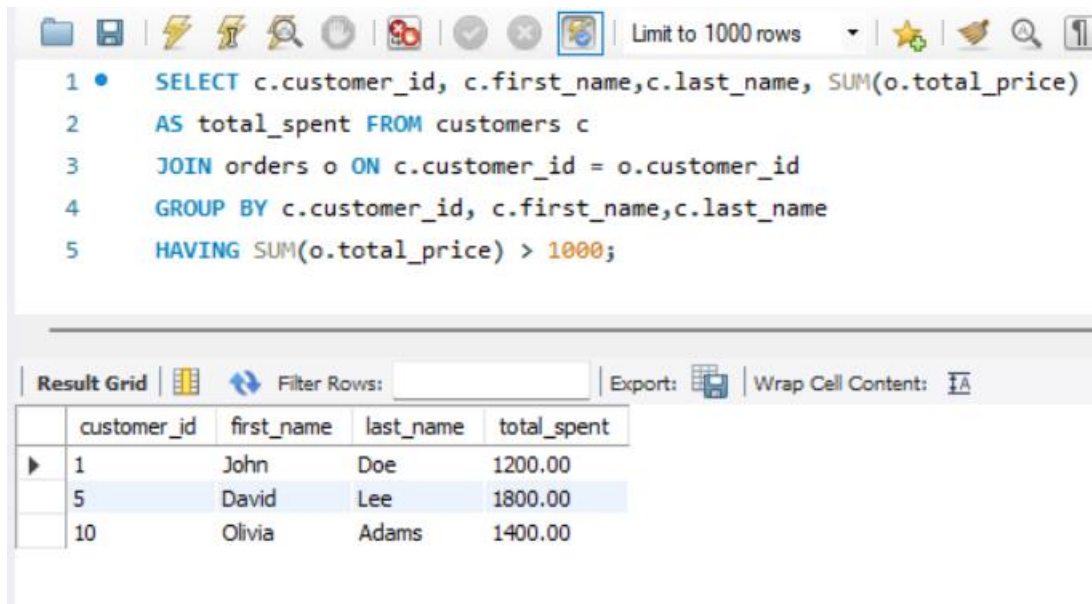
##### Query:

```

SELECT c.customer_id, c.first_name, c.last_name, SUM(o.total_price) AS
total_spent FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.first_name, c.last_name
HAVING SUM(o.total_price) > 1000;

```

##### Result:



```

1 • SELECT c.customer_id, c.first_name, c.last_name, SUM(o.total_price)
2   AS total_spent FROM customers c
3   JOIN orders o ON c.customer_id = o.customer_id
4   GROUP BY c.customer_id, c.first_name, c.last_name
5   HAVING SUM(o.total_price) > 1000;

```

	customer_id	first_name	last_name	total_spent
▶	1	John	Doe	1200.00
	5	David	Lee	1800.00
	10	Olivia	Adams	1400.00

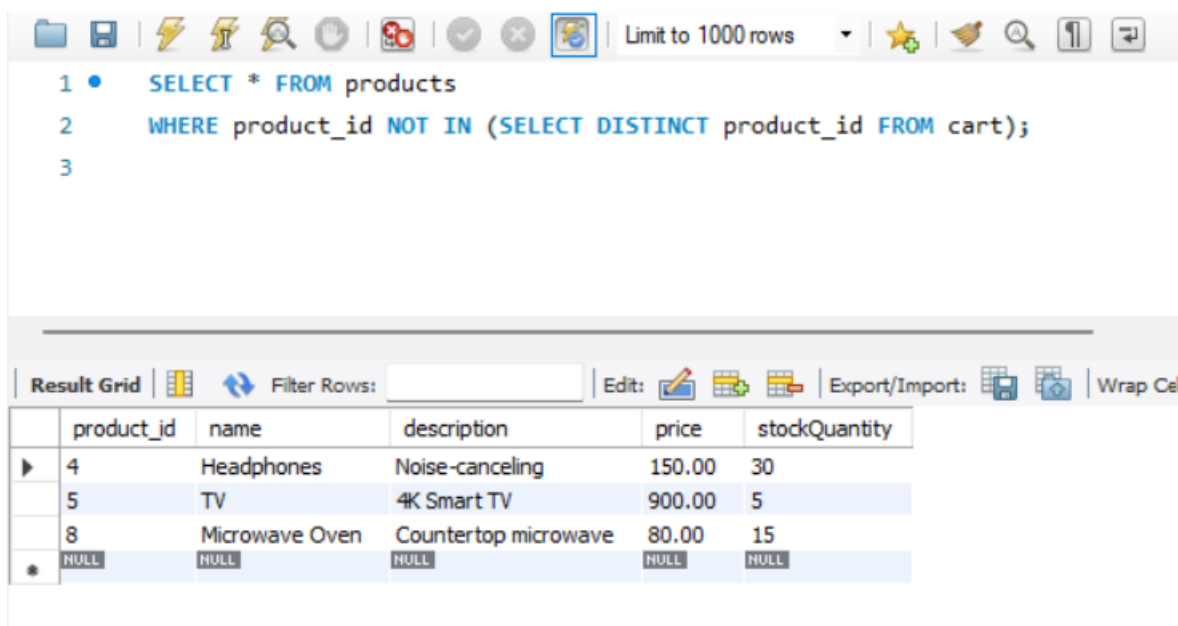
### 15.Subquery to Find Products Not in the Cart.

Query:

*SELECT \* FROM products*

*WHERE product\_id NOT IN (SELECT DISTINCT product\_id FROM cart);*

Result:



```

1 • SELECT * FROM products
2   WHERE product_id NOT IN (SELECT DISTINCT product_id FROM cart);
3

```

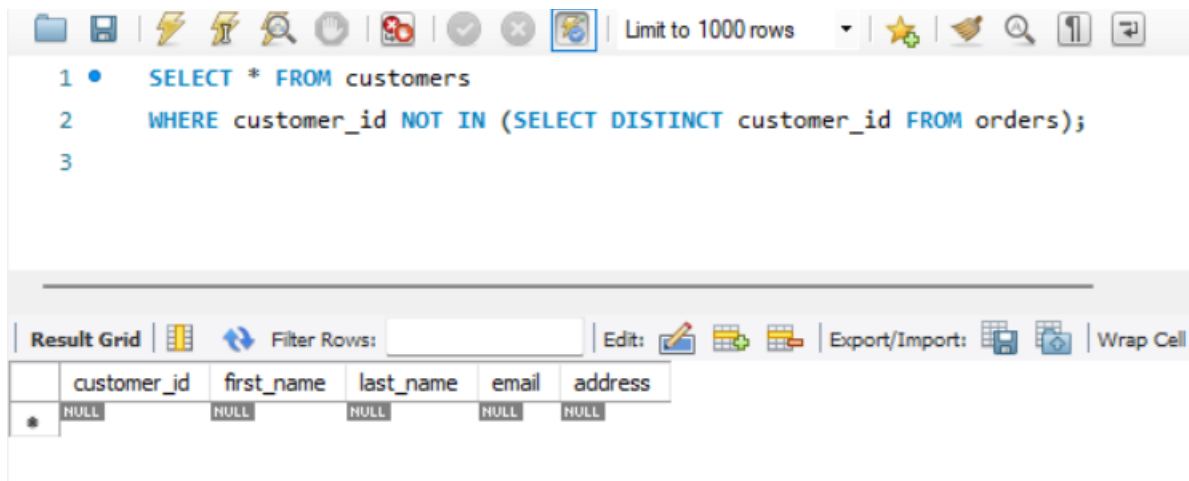
	product_id	name	description	price	stockQuantity
▶	4	Headphones	Noise-canceling	150.00	30
	5	TV	4K Smart TV	900.00	5
	8	Microwave Oven	Countertop microwave	80.00	15
*	NULL	NULL	NULL	NULL	NULL

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

### Query:

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

### Result:



The screenshot shows a SQL query editor with a toolbar at the top. The query is entered in the main text area:

```
1 • SELECT * FROM customers
2 WHERE customer_id NOT IN (SELECT DISTINCT customer_id FROM orders);
3
```

Below the query editor is the 'Result Grid' section. It has a toolbar with options like 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell'. The result grid itself shows a single row with five columns: 'customer\_id', 'first\_name', 'last\_name', 'email', and 'address'. All five cells in this row contain the value 'NULL'.

customer_id	first_name	last_name	email	address
NULL	NULL	NULL	NULL	NULL

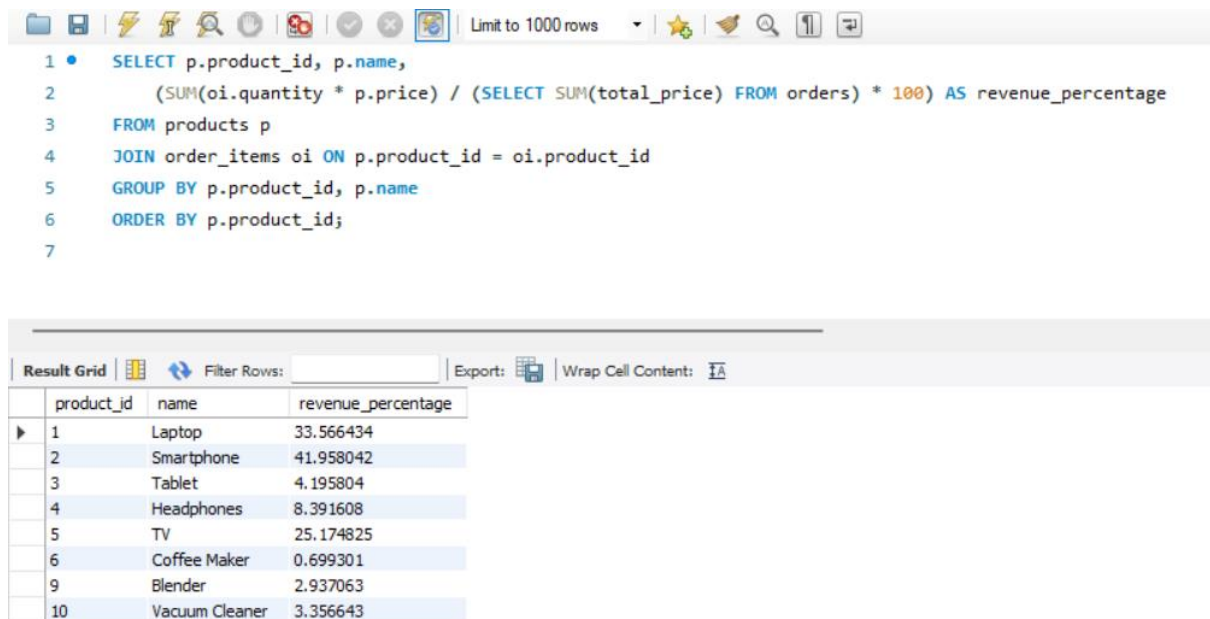
## 17.Subquery to Calculate the Percentage of Total Revenue for a Product.

### Query:

```
SELECT p.product_id, p.name,
(SUM(oi.quantity * p.price) / (SELECT SUM(total_price) FROM orders) * 100)
AS revenue_percentage
FROM products p
JOIN order_items oi ON p.product_id = oi.product_id
GROUP BY p.product_id, p.name
ORDER BY p.product_id;
```

### Result:





The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```

1 • SELECT p.product_id, p.name,
2     (SUM(oi.quantity * p.price) / (SELECT SUM(total_price) FROM orders) * 100) AS revenue_percentage
3 FROM products p
4 JOIN order_items oi ON p.product_id = oi.product_id
5 GROUP BY p.product_id, p.name
6 ORDER BY p.product_id;
7

```

Below the query editor is a 'Result Grid' showing the results of the query. The grid has three columns: product\_id, name, and revenue\_percentage. The results are as follows:

	product_id	name	revenue_percentage
▶	1	Laptop	33.566434
	2	Smartphone	41.958042
	3	Tablet	4.195804
	4	Headphones	8.391608
	5	TV	25.174825
	6	Coffee Maker	0.699301
	9	Blender	2.937063
	10	Vacuum Cleaner	3.356643

## 18.Subquery to Find Products with Low Stock.

### Query:

*SELECT name, stockQuantity*

*FROM products*

*WHERE stockQuantity = (SELECT MIN(stockQuantity) FROM products);*

### Result:

```

1 • SELECT product_id,name,description,price, stockQuantity
2 FROM products
3 WHERE stockQuantity = (SELECT MIN(stockQuantity) FROM products);
4

```

product_id	name	description	price	stockQuantity
5	TV	4K Smart TV	900.00	5
NULL	NULL	NULL	NULL	NULL

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

#### Query:

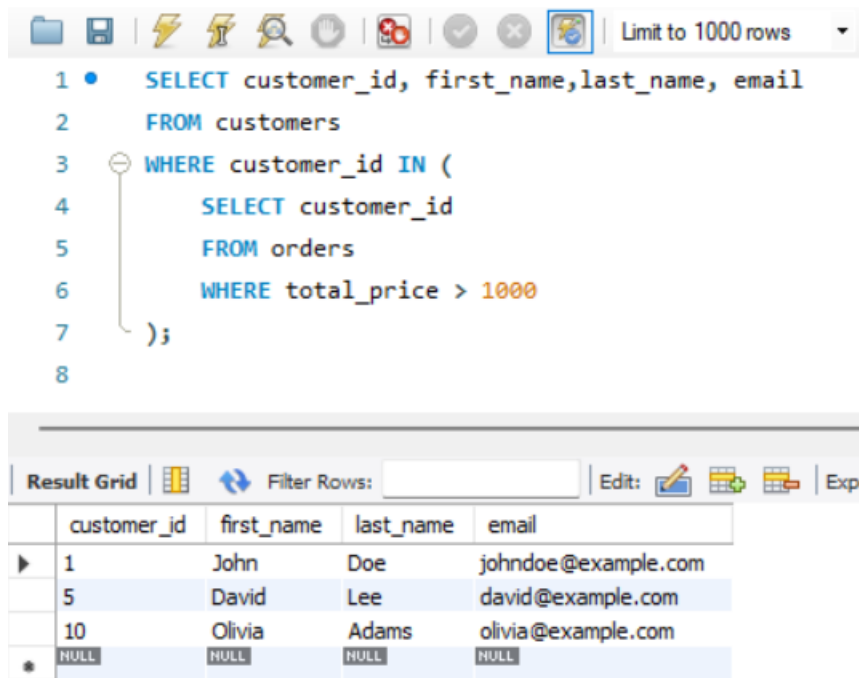
```

SELECT customer_id, first_name, last_name, email
FROM customers
WHERE customer_id IN (SELECT customer_id FROM orders
WHERE total_price > 1000);

```

#### Result:

**Note:** Assuming that the Order Values which are above \$1000 is considered as the High-Value orders.



The screenshot displays a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a dropdown menu set to "Limit to 1000 rows". The SQL editor contains the following query:

```
1 • SELECT customer_id, first_name, last_name, email
2 FROM customers
3 WHERE customer_id IN (
4     SELECT customer_id
5     FROM orders
6     WHERE total_price > 1000
7 );
8
```

Below the editor is the "Result Grid" section, which includes a "Filter Rows" input field and an "Edit" button. The results are shown in a table with the following data:

	customer_id	first_name	last_name	email
▶	1	John	Doe	johndoe@example.com
	5	David	Lee	david@example.com
	10	Olivia	Adams	olivia@example.com
•	NULL	NULL	NULL	NULL

