Hexaware Training Assessment

Coding Challenge (SQL)

Submitted By

Name: Mukund Suresh Sutar

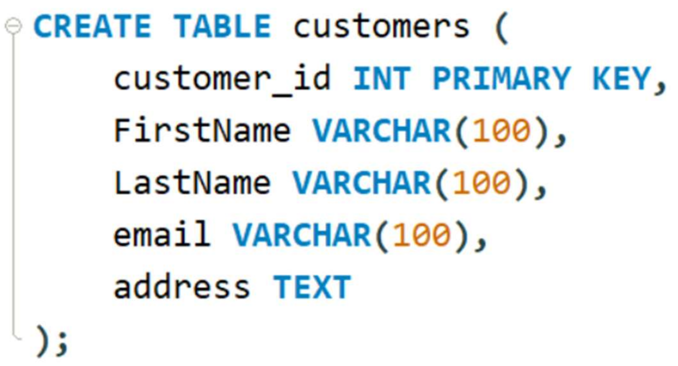
Email: mukundmoto89@gmail.com

GitHub Link: https://github.com/mukkund05/Coding_challange_SQL

Title: Coding Challenge 6 - Ecommerce

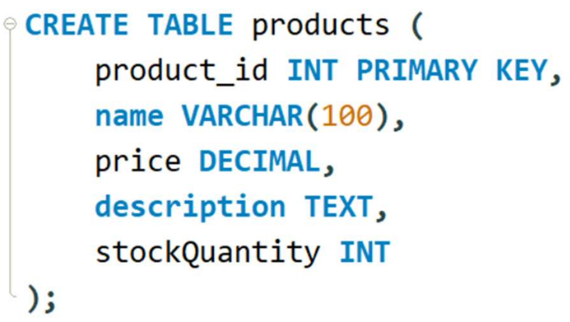
TABLES:

1. **Customers table**



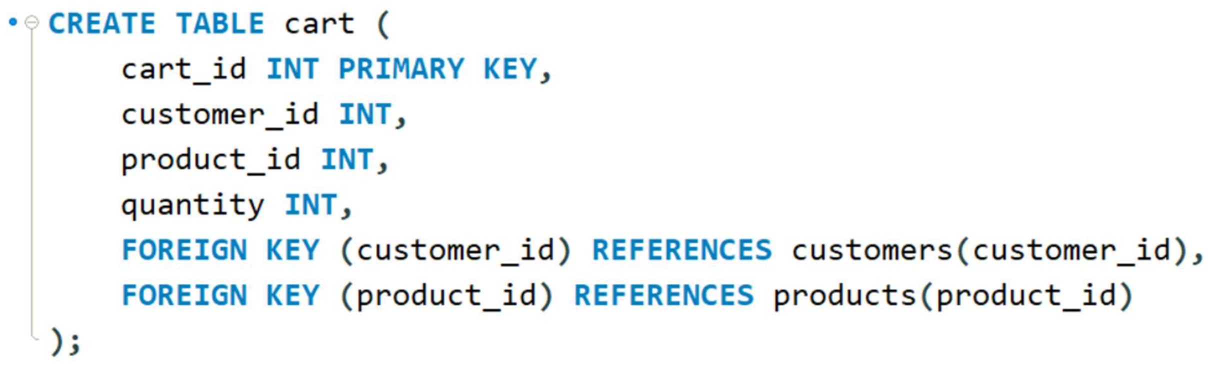
```
CREATE TABLE customers (
    customer_id INT PRIMARY KEY,
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
    email VARCHAR(100),
    address TEXT
);
```

2. **Product table**



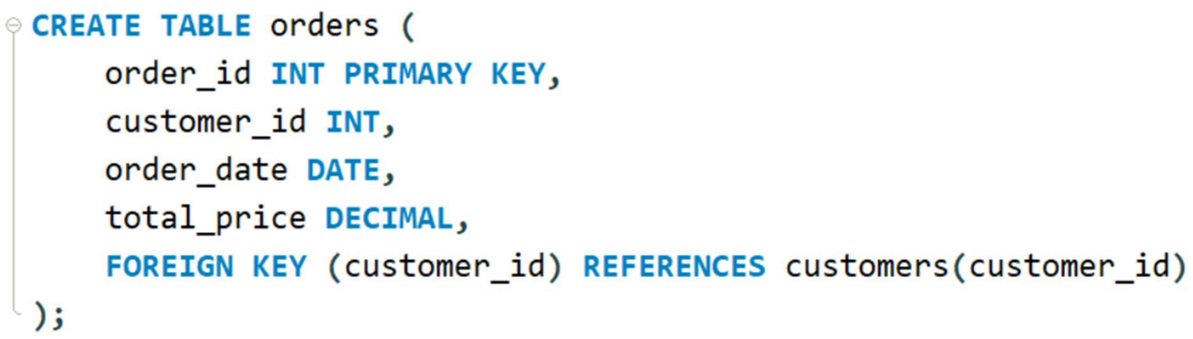
```
CREATE TABLE products (
    product_id INT PRIMARY KEY,
    name VARCHAR(100),
    price DECIMAL,
    description TEXT,
    stockQuantity INT
);
```

3. Cart table



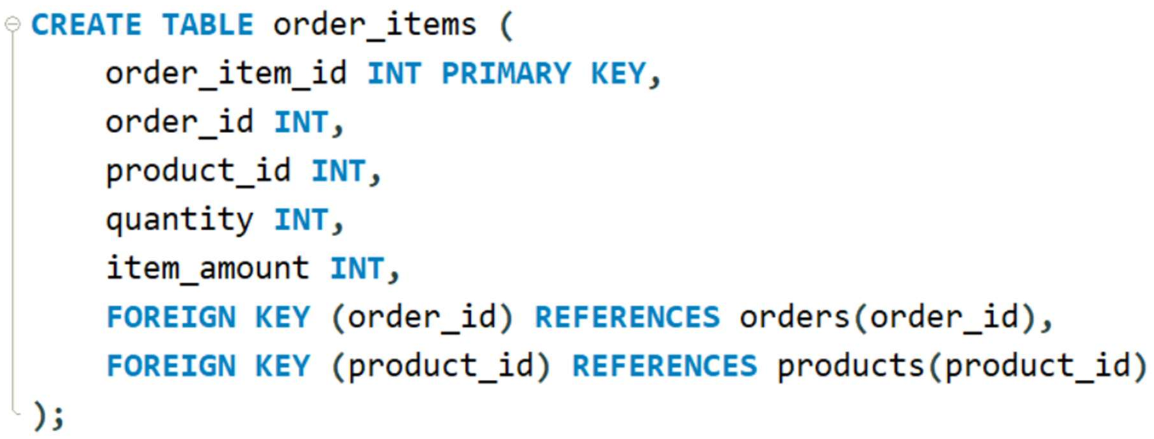
```sql
CREATE TABLE cart (
    cart_id INT PRIMARY KEY,
    customer_id INT,
    product_id INT,
    quantity INT,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

4. Orders table



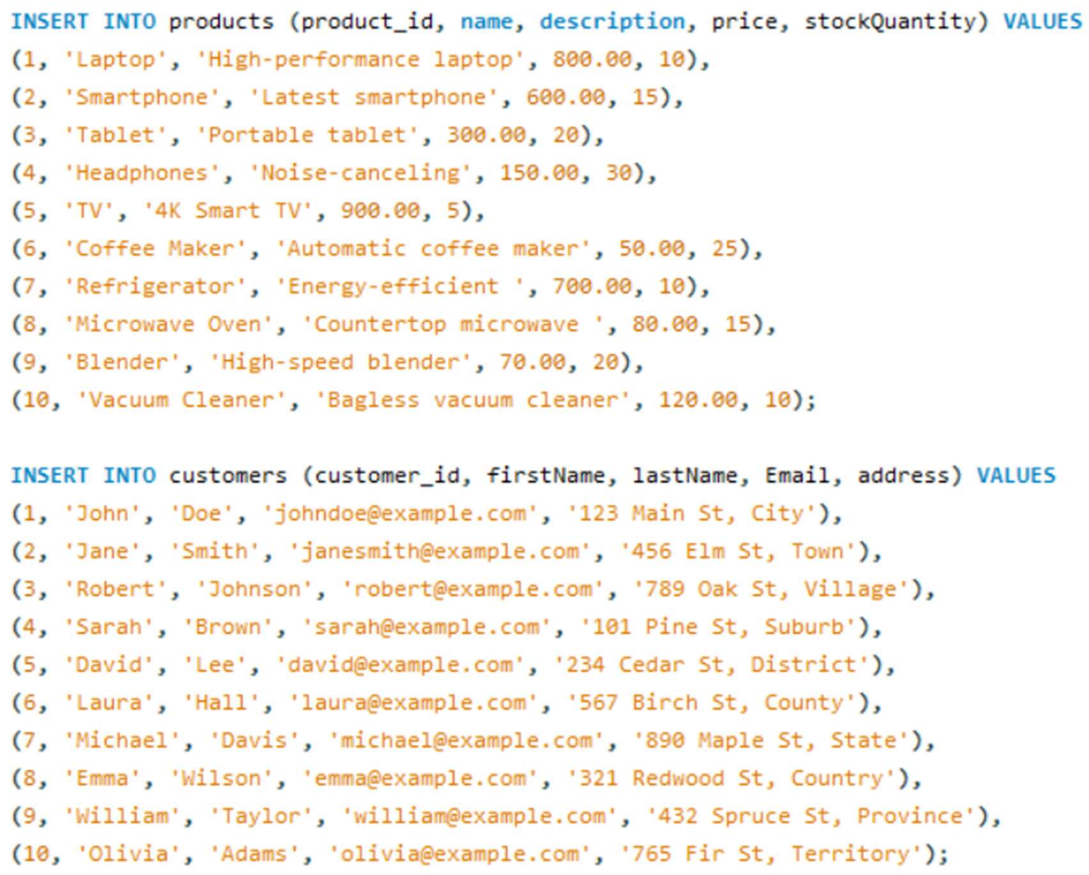
```sql
CREATE TABLE orders (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    total_price DECIMAL,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
```

5. Order_items table

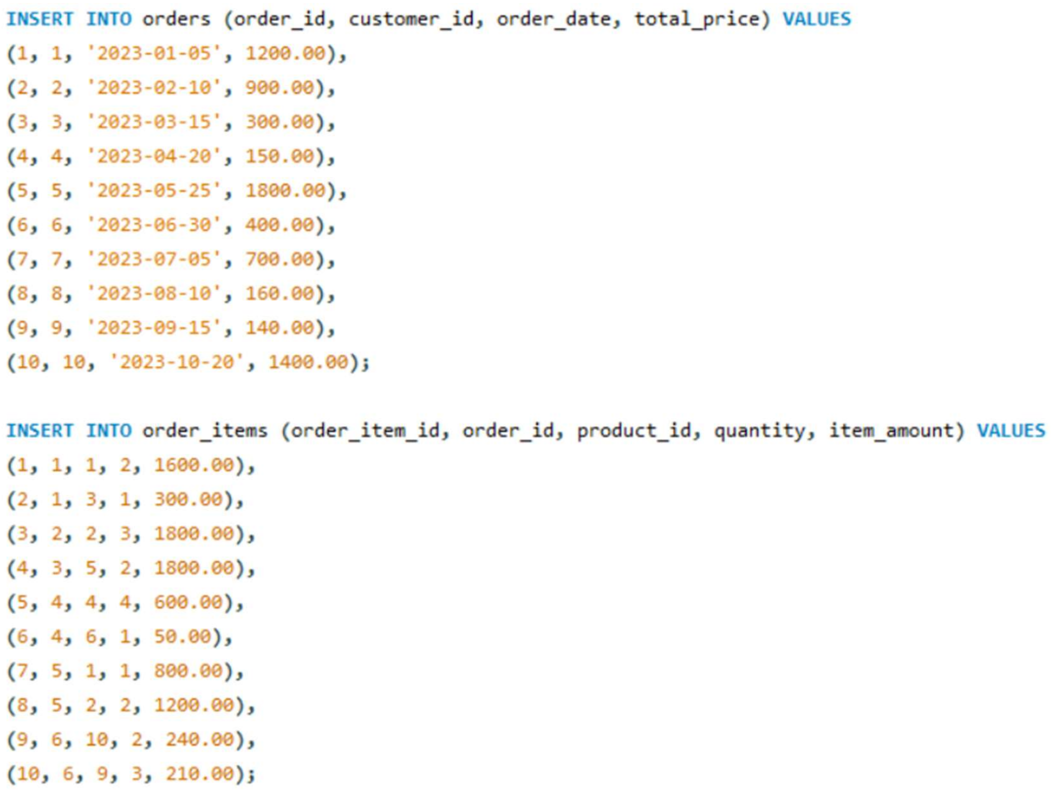


```sql
CREATE TABLE order_items (
    order_item_id INT PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT,
    item_amount INT,
    FOREIGN KEY (order_id) REFERENCES orders(order_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

Data:



```sql
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);

INSERT INTO customers (customer_id, firstName, lastName, 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');
```



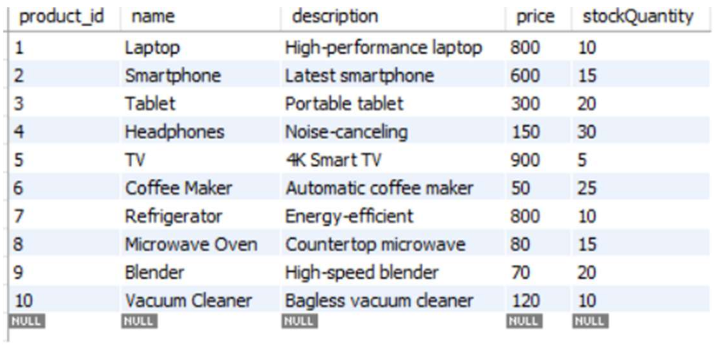
```sql
INSERT INTO orders (order_id, customer_id, order_date, total_price) VALUES
(1, 1, '2023-01-05', 1200.00),
(2, 2, '2023-02-10', 900.00),
(3, 3, '2023-03-15', 300.00),
(4, 4, '2023-04-20', 150.00),
(5, 5, '2023-05-25', 1800.00),
(6, 6, '2023-06-30', 400.00),
(7, 7, '2023-07-05', 700.00),
(8, 8,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

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);
```

```
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);
```

Questions:

1. Update refrigerator product price to 800.

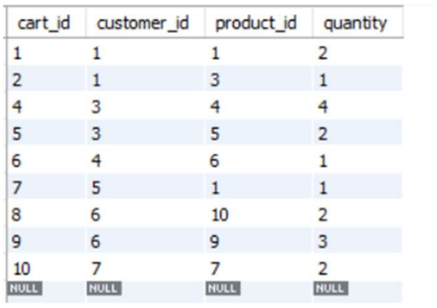
```
UPDATE products
SET price = 800
WHERE name ="Refrigerator";
```



product_id	name	description	price	stockQuantity
1	Laptop	High-performance laptop	800	10
2	Smartphone	Latest smartphone	600	15
3	Tablet	Portable tablet	300	20
4	Headphones	Noise-canceling	150	30
5	TV	4K Smart TV	900	5
6	Coffee Maker	Automatic coffee maker	50	25
7	Refrigerator	Energy-efficient	800	10
8	Microwave Oven	Countertop microwave	80	15
9	Blender	High-speed blender	70	20
10	Vacuum Cleaner	Bagless vacuum cleaner	120	10
NULL	NULL	NULL	NULL	NULL

2. Remove all cart items for a specific customer.

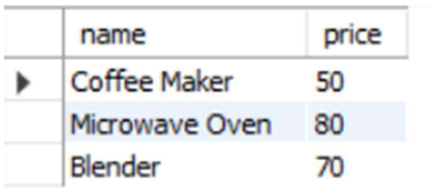
```
DELETE FROM cart
WHERE customer_id = 2;
```



cart_id	customer_id	product_id	quantity
1	1	1	2
2	1	3	1
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

3. Retrieve Products Priced Below $100.

```
select name, price from products where price < 100;
```



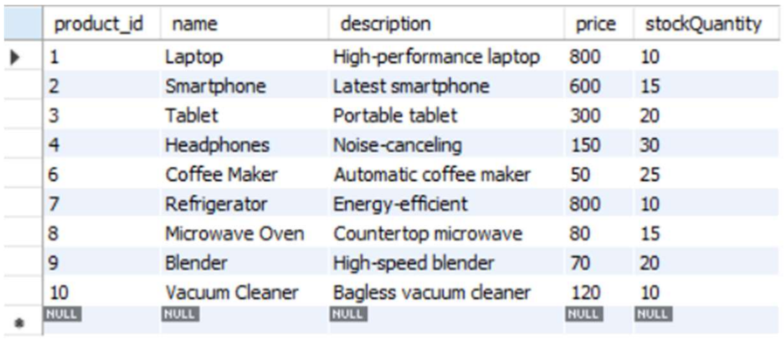
name	price
Coffee Maker	50
Microwave Oven	80
Blender	70

4. Find Products with Stock Quantity Greater Than 5.

Query:

```
select * from products where stockQuantity > 5;
```

Output:



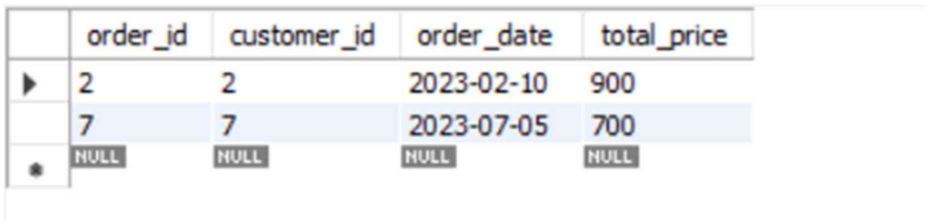
product_id	name	description	price	stockQuantity
1	Laptop	High-performance laptop	800	10
2	Smartphone	Latest smartphone	600	15
3	Tablet	Portable tablet	300	20
4	Headphones	Noise-canceling	150	30
6	Coffee Maker	Automatic coffee maker	50	25
7	Refrigerator	Energy-efficient	800	10
8	Microwave Oven	Countertop microwave	80	15
9	Blender	High-speed blender	70	20
10	Vacuum Cleaner	Bagless vacuum cleaner	120	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;
```

Output:



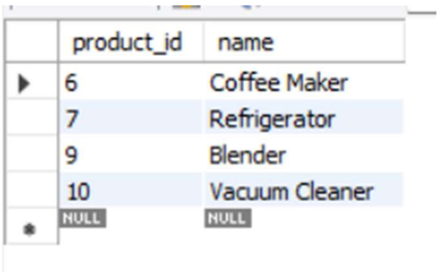
order_id	customer_id	order_date	total_price
2	2	2023-02-10	900
7	7	2023-07-05	700
NULL	NULL	NULL	NULL

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

Query:

```
select product_id, name from products where name like "%r";
```

Output:



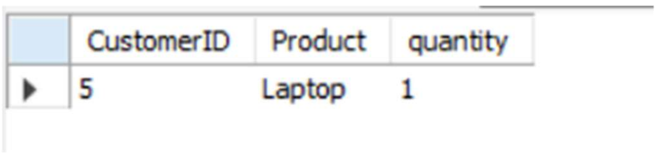
```
product_id  name
6           Coffee Maker
7           Refrigerator
9           Blender
10          Vacuum Cleaner
NULL        NULL
```

7. Retrieve Cart Items for Customer 5.

Query:

```
select  c.customer_id as CustomerID, p.name as Product, c.quantity
from cart c
JOIN products p ON
c.product_id = p.product_id
where c.customer_id = 5;
```

Output:



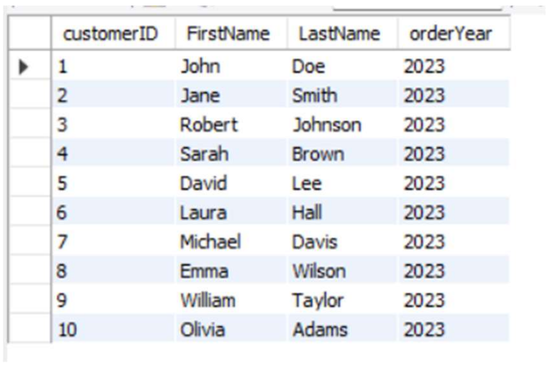
```
CustomerID  Product  quantity
5           Laptop   1
```

8. Find Customers Who Placed Orders in 2023.

Query:

```
select
c.customer_id as customerID,
c.FirstName as FirstName,
c.lastName as LastName,
year(o.order_date) as orderYear
from customers c
JOIN orders o
ON c.customer_id = o.customer_id
WHERE year(o.order_date) = 2023;
```

Output:



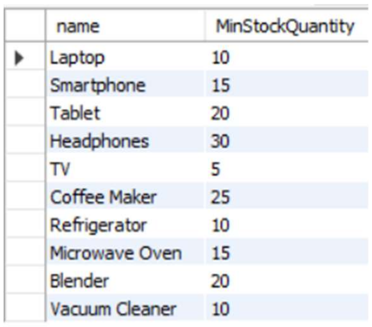
	customerID	FirstName	LastName	orderYear
▶	1	John	Doe	2023
	2	Jane	Smith	2023
	3	Robert	Johnson	2023
	4	Sarah	Brown	2023
	5	David	Lee	2023
	6	Laura	Hall	2023
	7	Michael	Davis	2023
	8	Emma	Wilson	2023
	9	William	Taylor	2023
	10	Olivia	Adams	2023

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

Query:

```
select name, MIN(stockQuantity) as MinStockQuantity
from products
group by name;
```

Output:



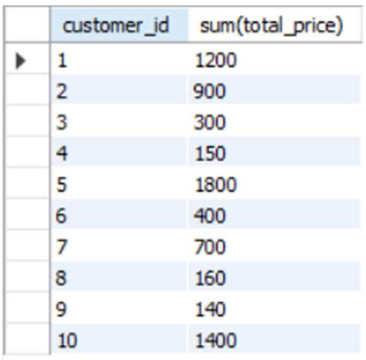
	name	MinStockQuantity
▶	Laptop	10
	Smartphone	15
	Tablet	20
	Headphones	30
	TV	5
	Coffee Maker	25
	Refrigerator	10
	Microwave Oven	15
	Blender	20
	Vacuum Cleaner	10

10. Calculate the Total Amount Spent by Each Customer.

Query:

```
select customer_id, sum(total_price) from orders
group by customer_id;
```

Output:



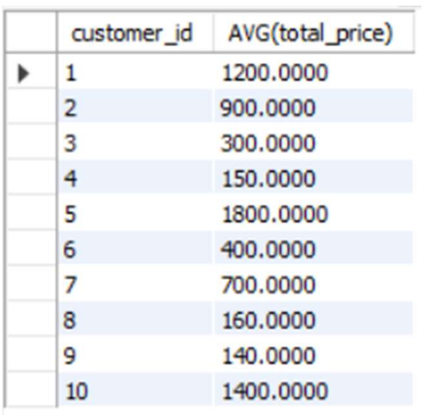
customer_id	sum(total_price)
1	1200
2	900
3	300
4	150
5	1800
6	400
7	700
8	160
9	140
10	1400

11. Find the Average Order Amount for Each Customer.

Query:

```
select customer_id, AVG(total_price) from orders
group by customer_id;
```

Output:



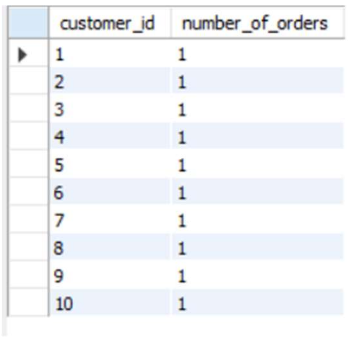
customer_id	AVG(total_price)
1	1200.0000
2	900.0000
3	300.0000
4	150.0000
5	1800.0000
6	400.0000
7	700.0000
8	160.0000
9	140.0000
10	1400.0000

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

Query:

```
select customer_id, COUNT(order_id) AS number_of_orders
from orders
group by customer_id;
```

Output:



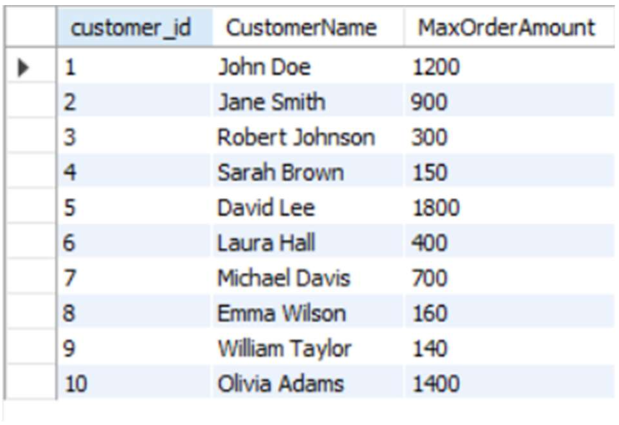
customer_id	number_of_orders
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1

13. Find the Maximum Order Amount for Each Customer.

Query:

```
select  o.customer_id ,
concat(c.firstName," ", c.lastName) as CustomerName,
max(o.total_price) as MaxOrderAmount
from orders o
join customers c
on c.customer_id = o.customer_id
group by o.customer_id;
```

Output:



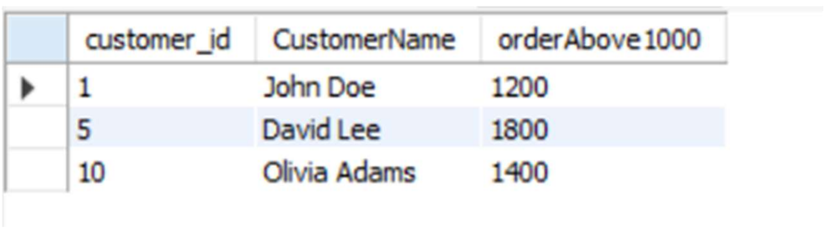
customer_id	CustomerName	MaxOrderAmount
1	John Doe	1200
2	Jane Smith	900
3	Robert Johnson	300
4	Sarah Brown	150
5	David Lee	1800
6	Laura Hall	400
7	Michael Davis	700
8	Emma Wilson	160
9	William Taylor	140
10	Olivia Adams	1400

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

Query:

```
select  o.customer_id ,
concat(c.firstName," ", c.lastName) as CustomerName,
o.total_price as orderAbove1000
from orders o
join customers c
on c.customer_id = o.customer_id
where o.total_price > 1000;
```

Output:



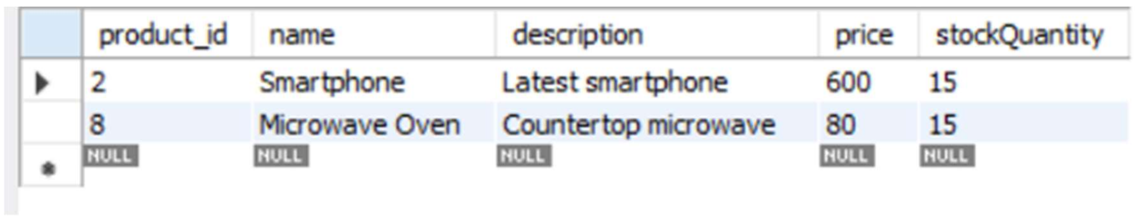
customer_id	CustomerName	orderAbove1000
1	John Doe	1200
5	David Lee	1800
10	Olivia Adams	1400

15. Subquery to Find Products Not in the Cart.

Query:

```
select *
from products
where product_id not in (select product_id from cart);
```

Output:

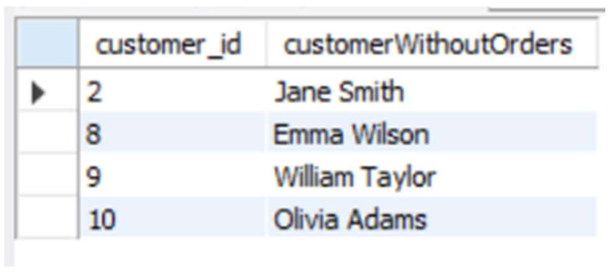


product_id	name	description	price	stockQuantity
2	Smartphone	Latest smartphone	600	15
8	Microwave Oven	Countertop microwave	80	15
NULL	NULL	NULL	NULL	NULL

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

Query:

```
select customer_id, concat(firstName," ", Lastname) as customerWithoutOrders from customers
WHERE customer_id NOT IN (select customer_id from cart);
```



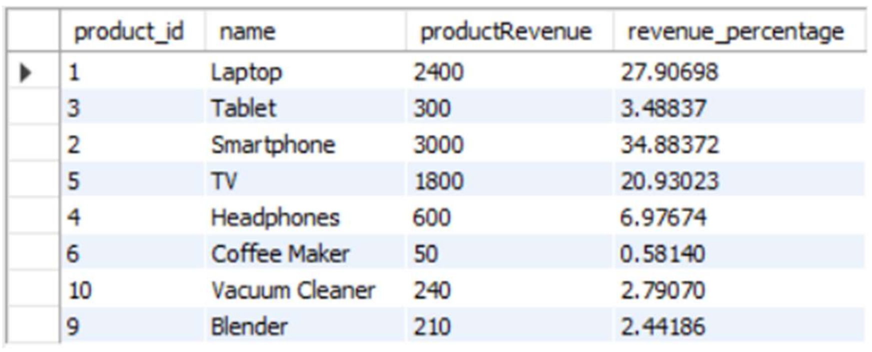
customer_id	customerWithoutOrders
2	Jane Smith
8	Emma Wilson
9	William Taylor
10	Olivia Adams

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

Query:

```
select o.product_id, p.name, sum(o.item_amount) as productRevenue,
(SUM(o.item_amount) * 100.0) / (SELECT SUM(item_amount) FROM order_items) AS revenue_percentage
 from order_items o
join products p
on p.product_id = o.product_id
group by product_id;
```

Output:



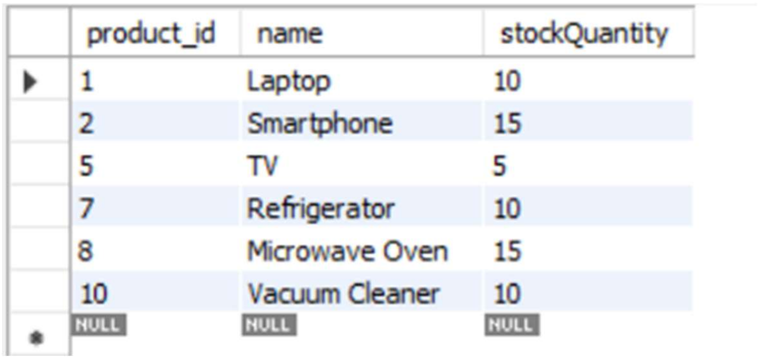
product_id	name	productRevenue	revenue_percentage
1	Laptop	2400	27.90698
3	Tablet	300	3.48837
2	Smartphone	3000	34.88372
5	TV	1800	20.93023
4	Headphones	600	6.97674
6	Coffee Maker	50	0.58140
10	Vacuum Cleaner	240	2.79070
9	Blender	210	2.44186

18. Subquery to Find Products with Low Stock.

Query:

```
select product_id, name, stockQuantity from products
WHERE stockQuantity < (select avg(stockQuantity) from products);
```

Output:



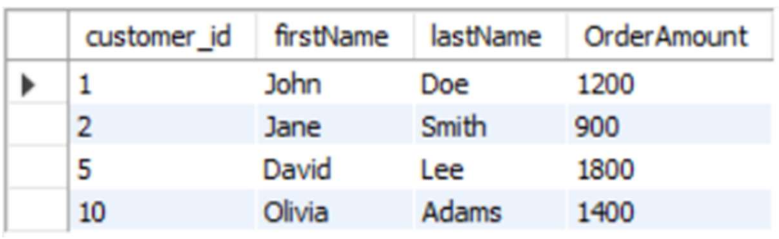
product_id	name	stockQuantity
1	Laptop	10
2	Smartphone	15
5	TV	5
7	Refrigerator	10
8	Microwave Oven	15
10	Vacuum Cleaner	10
NULL	NULL	NULL

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

Query:

```
select c.customer_id, c.firstName, c.lastName, o.total_price as OrderAmount from customers c
join orders o
on c.customer_id  = o.customer_id
where o.total_price > (select avg(total_price) from orders);
```

Output:



customer_id	firstName	lastName	OrderAmount
1	John	Doe	1200
2	Jane	Smith	900
5	David	Lee	1800
10	Olivia	Adams	1400