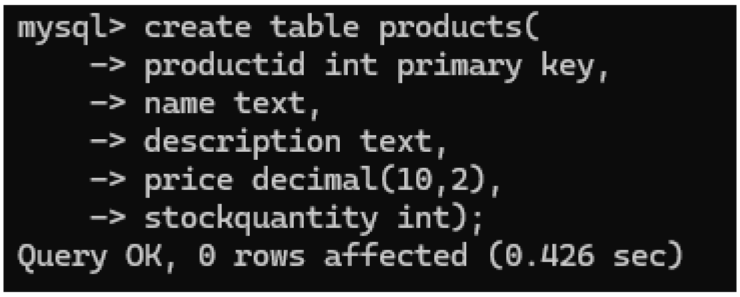
Coding Challenge:

Ecommerce – SQL

By Esaq A

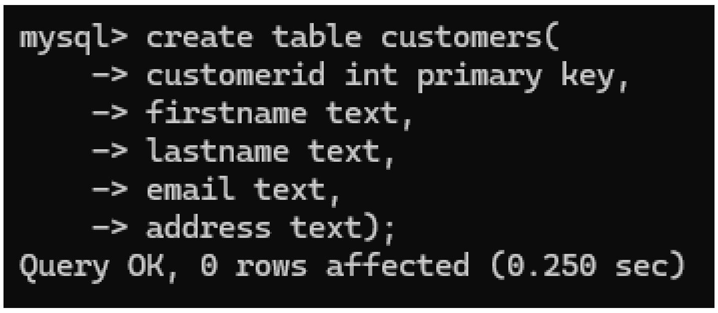
I. Creating Tables:

- products table:



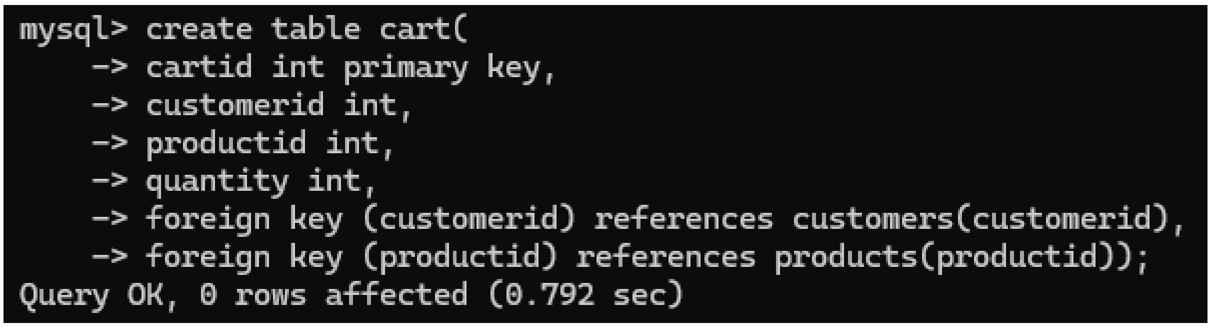
```
mysql> create table products(
    -> productid int primary key,
    -> name text,
    -> description text,
    -> price decimal(10,2),
    -> stockquantity int);
Query OK, 0 rows affected (0.426 sec)
```

- customers table:



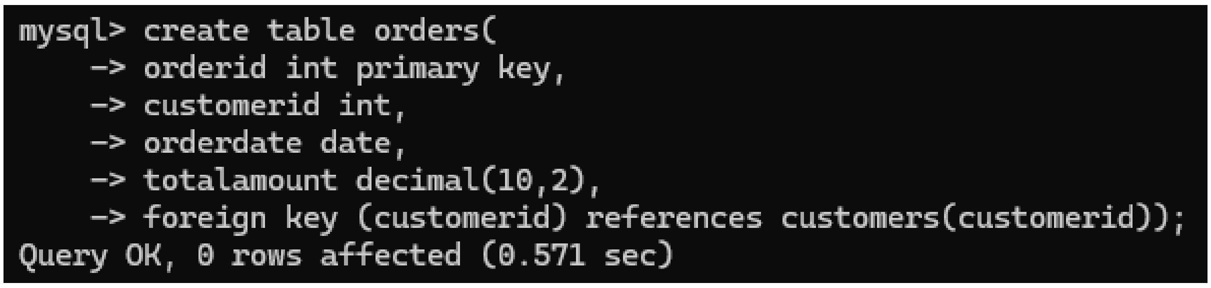
```
mysql> create table customers(
    -> customerid int primary key,
    -> firstname text,
    -> lastname text,
    -> email text,
    -> address text);
Query OK, 0 rows affected (0.250 sec)
```

- cart table:



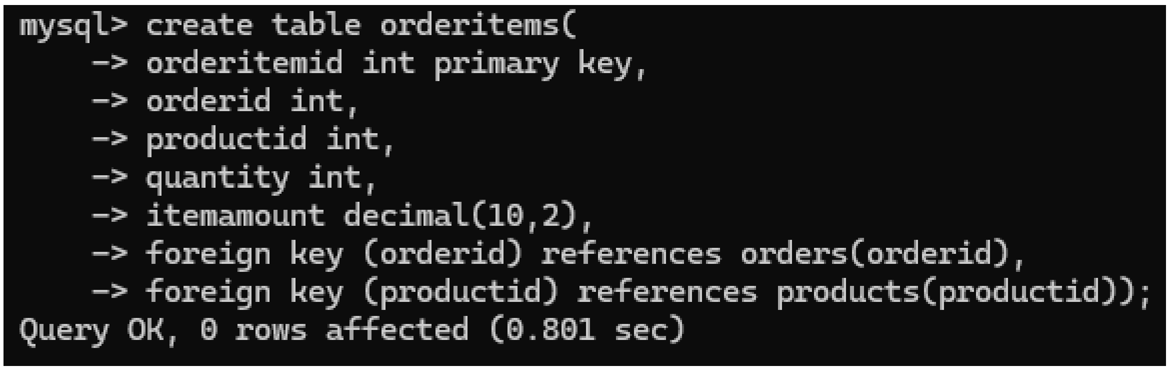
```
mysql> create table cart(
    -> cartid int primary key,
    -> customerid int,
    -> productid int,
    -> quantity int,
    -> foreign key (customerid) references customers(customerid),
    -> foreign key (productid) references products(productid));
Query OK, 0 rows affected (0.792 sec)
```

- orders table:



```
mysql> create table orders(
    -> orderid int primary key,
    -> customerid int,
    -> orderdate date,
    -> totalamount decimal(10,2),
    -> foreign key (customerid) references customers(customerid));
Query OK, 0 rows affected (0.571 sec)
```

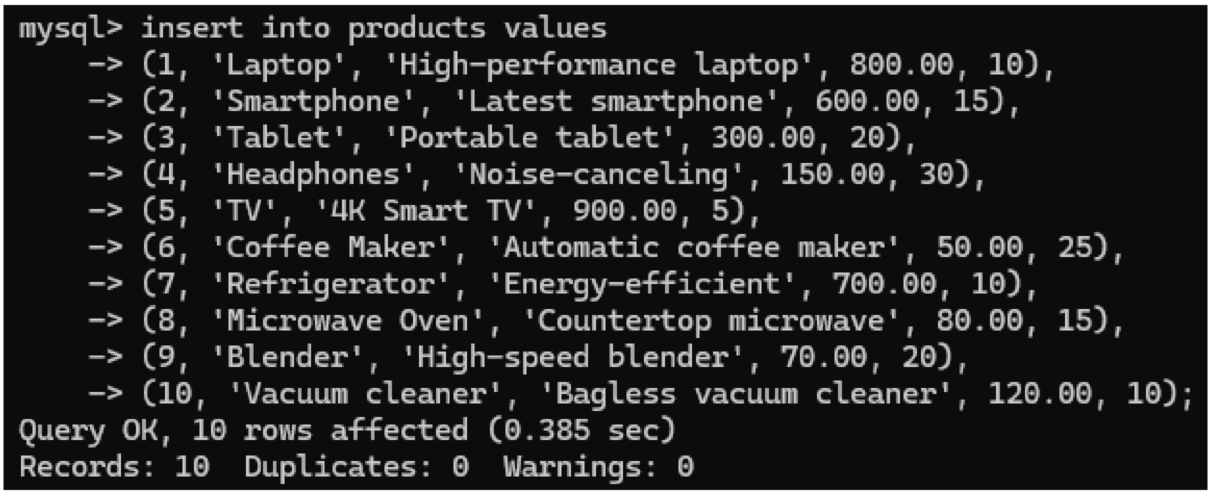
- orderitems table:



```
mysql> create table orderitems(
    -> orderitemid int primary key,
    -> orderid int,
    -> productid int,
    -> quantity int,
    -> itemamount decimal(10,2),
    -> foreign key (orderid) references orders(orderid),
    -> foreign key (productid) references products(productid));
Query OK, 0 rows affected (0.801 sec)
```

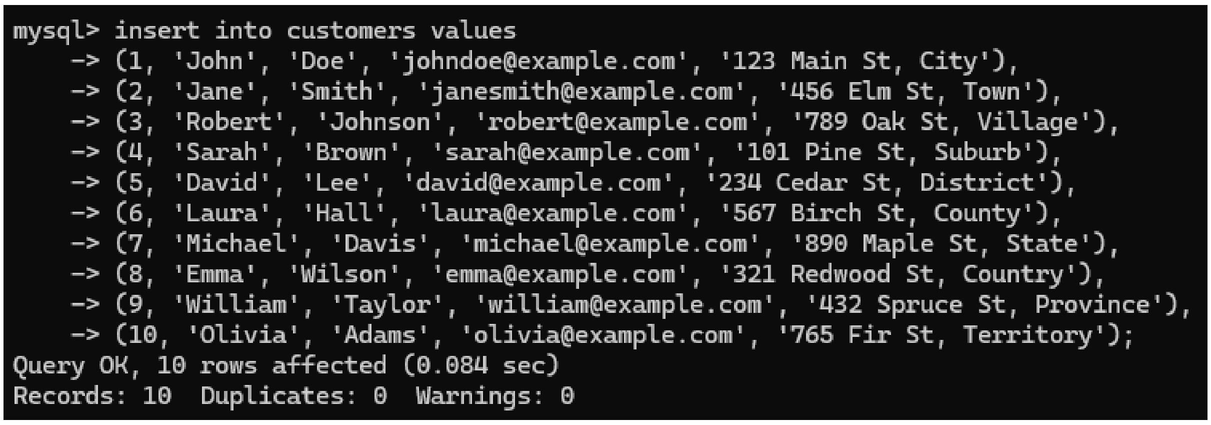
II. Inserting Values:

- products table:



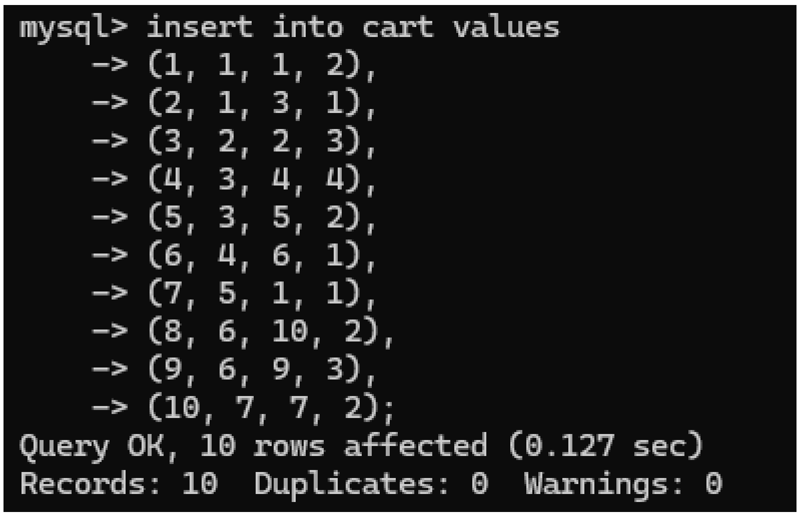
```
mysql> insert into products values
    -> (1, 'Laptop', 'High-performance laptop', 800.00, 10),
    -> (2, 'Smartphone', 'Latest smartphone', 600.00, 15),
    -> (3, 'Tablet', 'Portable tablet', 300.00, 20),
    -> (4, 'Headphones', 'Noise-canceling', 150.00, 30),
    -> (5, 'TV', '4K Smart TV', 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);
Query OK, 10 rows affected (0.385 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

- customers table:



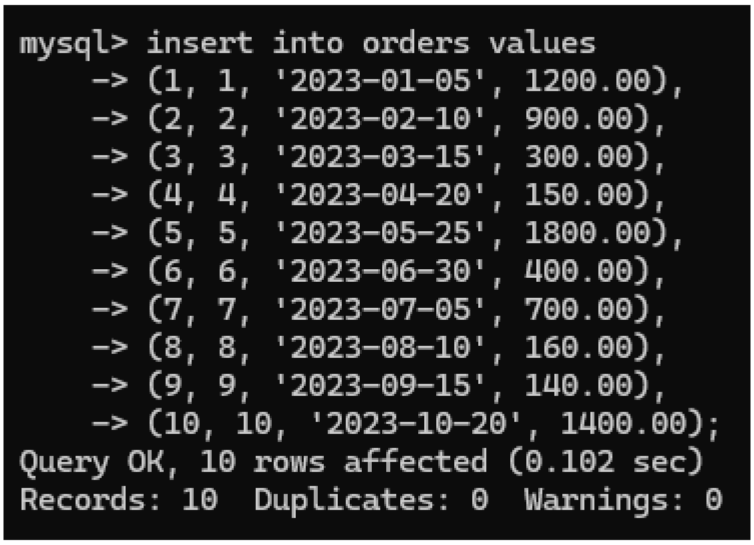
```
mysql> insert into customers values
    -> (1, 'John', 'Doe', 'johndoe@example.com', '123 Main St, City'),
    -> (2, 'Jane', 'Smith', 'janesmith@example.com', '456 Elm St, Town'),
    -> (3, 'Robert', 'Johnson', 'robert@example.com', '789 Oak St, Village'),
    -> (4, 'Sarah', 'Brown', 'sarah@example.com', '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');
Query OK, 10 rows affected (0.084 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

- cart table:



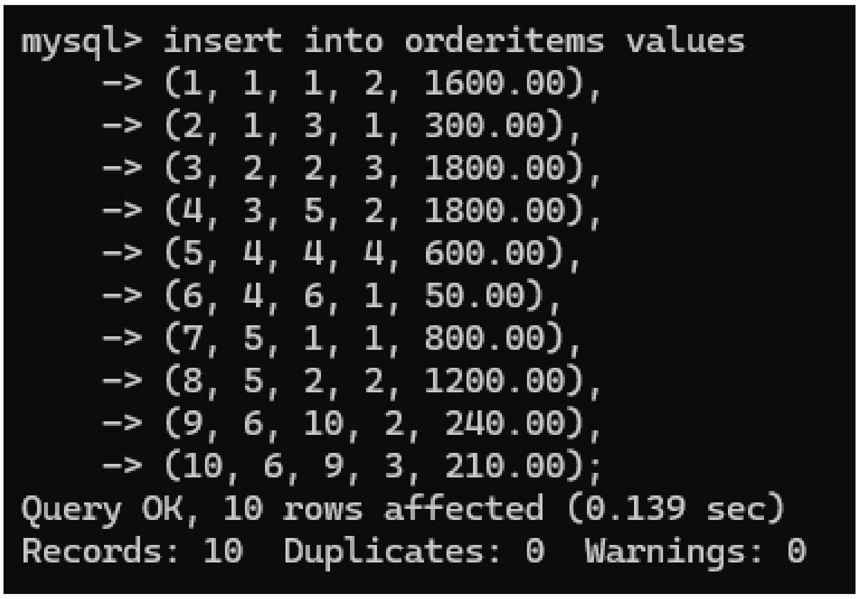
```
mysql> insert into cart values
    -> (1, 1, 1, 2),
    -> (2, 1, 3, 1),
    -> (3, 2, 2, 3),
    -> (4, 3, 4, 4),
    -> (5, 3, 5, 2),
    -> (6, 4, 6, 1),
    -> (7, 5, 1, 1),
    -> (8, 6, 10, 2),
    -> (9, 6, 9, 3),
    -> (10, 7, 7, 2);
Query OK, 10 rows affected (0.127 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

- orders table:



```
mysql> insert into orders 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);
Query OK, 10 rows affected (0.102 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

- orderitems table:

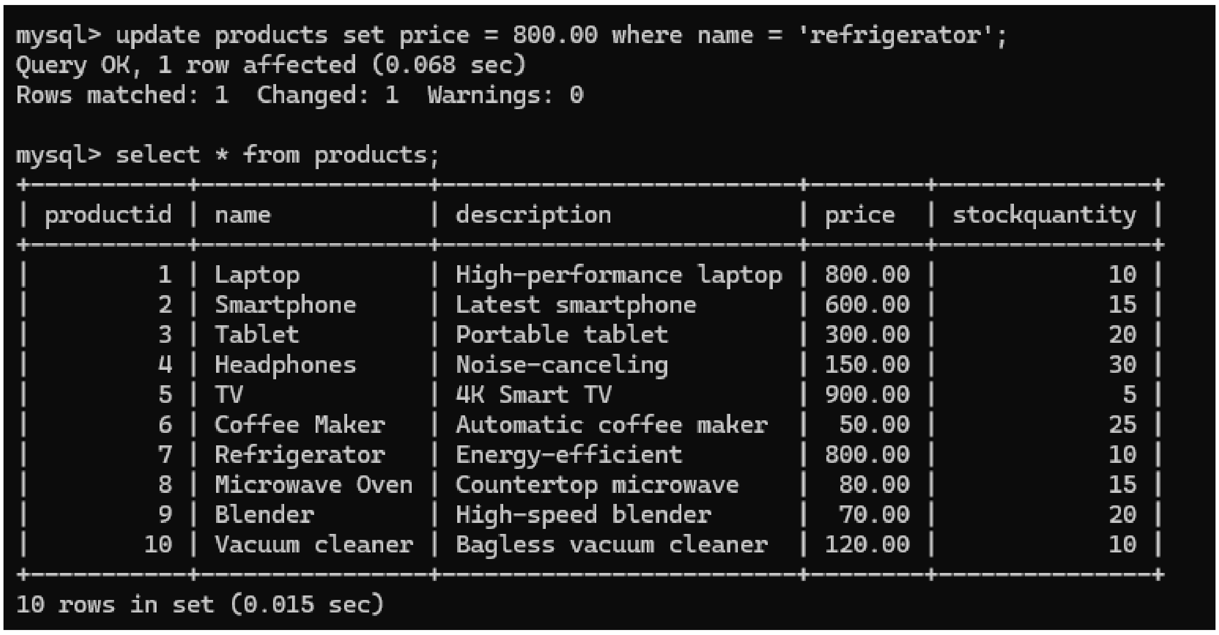


```
mysql> insert into orderitems values
    -> (1, 1, 1, 2, 1600.00),
    -> (2, 1, 3, 1, 300.00),
    -> (3, 2, 2, 3, 1800.00),
    -> (4, 3, 5, 2, 1800.00),
    -> (5, 4, 4, 4, 600.00),
    -> (6, 4, 6, 1, 50.00),
    -> (7, 5, 1, 1, 800.00),
    -> (8, 5, 2, 2, 1200.00),
    -> (9, 6, 10, 2, 240.00),
    -> (10, 6, 9, 3, 210.00);
Query OK, 10 rows affected (0.139 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

III. Queries:

1. Update refrigerator product price to 800.

update products set price = 800.00 where name = 'refrigerator';

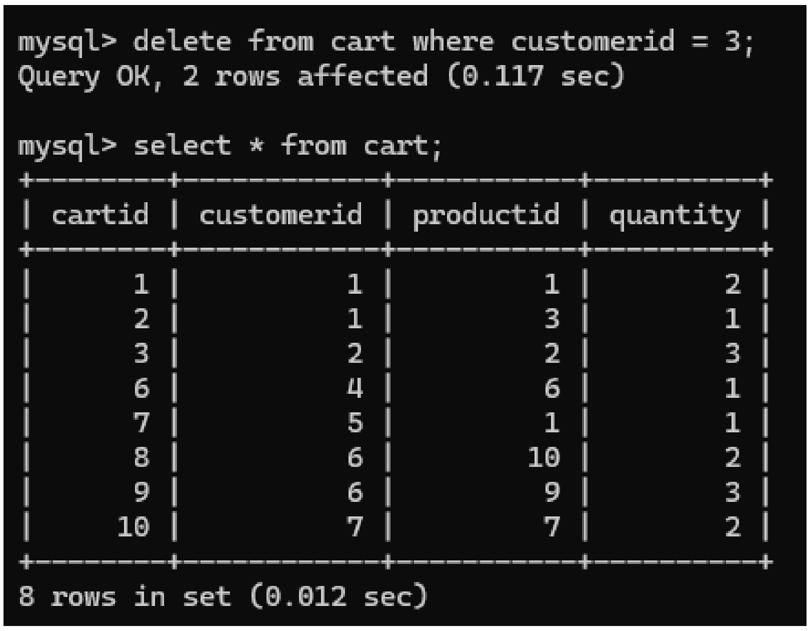


```
mysql> update products set price = 800.00 where name = 'refrigerator';
Query OK, 1 row affected (0.068 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from products;
+-----------+----------------+-------------------------+--------+---------------+
| productid | 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 |
+-----------+----------------+-------------------------+--------+---------------+
10 rows in set (0.015 sec)
```

2. Remove all cart items for a specific customer.

delete from cart where customerid = 3;

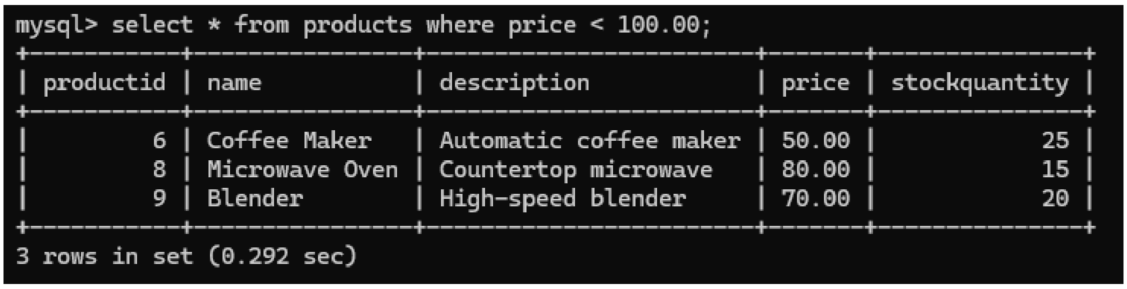


```
mysql> delete from cart where customerid = 3;
Query OK, 2 rows affected (0.117 sec)

mysql> select * from cart;
+--------+------------+-----------+----------+
| cartid | customerid | productid | quantity |
+--------+------------+-----------+----------+
|      1 |          1 |         1 |        2 |
|      2 |          1 |         3 |        1 |
|      3 |          2 |         2 |        3 |
|      6 |          4 |         6 |        1 |
|      7 |          5 |         1 |        1 |
|      8 |          6 |        10 |        2 |
|      9 |          6 |         9 |        3 |
|     10 |          7 |         7 |        2 |
+--------+------------+-----------+----------+
8 rows in set (0.012 sec)
```

3. Retrieve Products Priced Below $100.

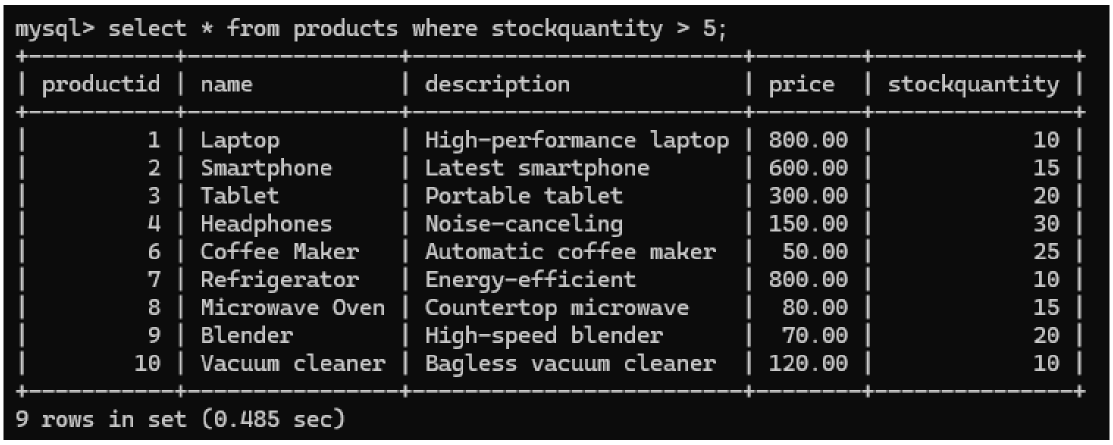
*select * from products where price < 100.00;*



```
mysql> select * from products where price < 100.00;
+-----------+----------------+-----------------------+-------+---------------+
| productid | name           | description           | price | stockquantity |
+-----------+----------------+-----------------------+-------+---------------+
|         6 | Coffee Maker   | Automatic coffee maker | 50.00 |            25 |
|         8 | Microwave Oven | Countertop microwave  | 80.00 |            15 |
|         9 | Blender        | High-speed blender    | 70.00 |            20 |
+-----------+----------------+-----------------------+-------+---------------+
3 rows in set (0.292 sec)
```

4. Find Products with Stock Quantity Greater Than 5.

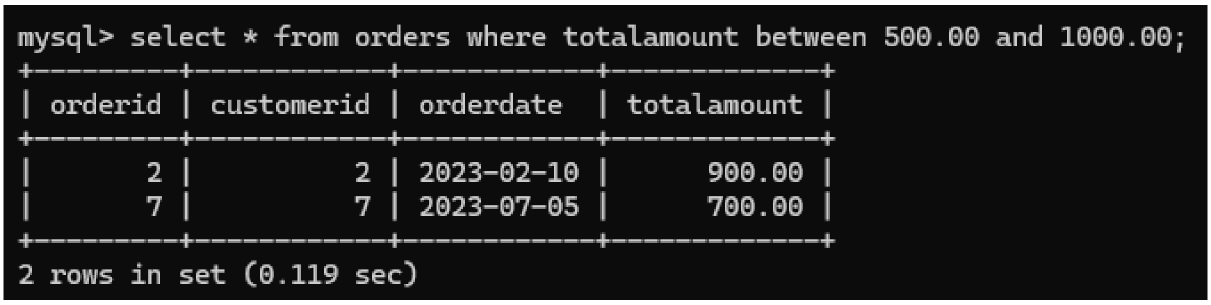
*select * from products where stockquantity > 5;*



```
mysql> select * from products where stockquantity > 5;
+-----------+----------------+-------------------------+--------+---------------+
| productid | 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 |
+-----------+----------------+-------------------------+--------+---------------+
9 rows in set (0.485 sec)
```

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

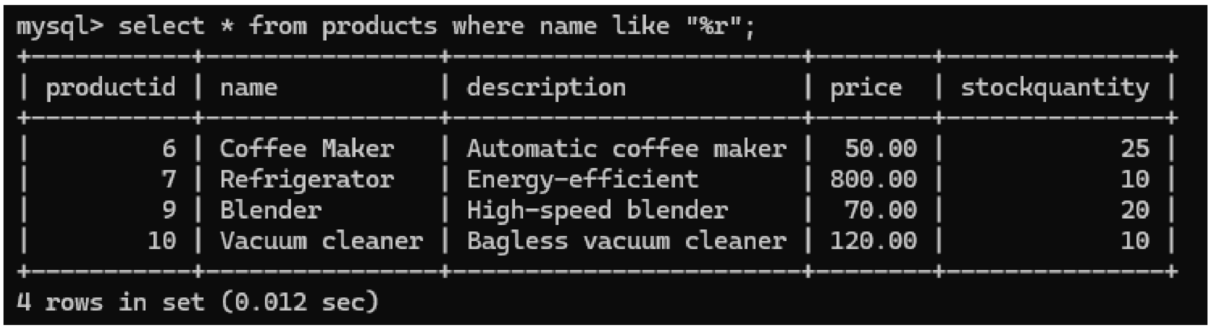
*select * from orders where totalamount between 500.00 and 1000.00;*



```
mysql> select * from orders where totalamount between 500.00 and 1000.00;
+---------+------------+------------+-------------+
| orderid | customerid | orderdate  | totalamount |
+---------+------------+------------+-------------+
|       2 |          2 | 2023-02-10 |      900.00 |
|       7 |          7 | 2023-07-05 |      700.00 |
+---------+------------+------------+-------------+
2 rows in set (0.119 sec)
```

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

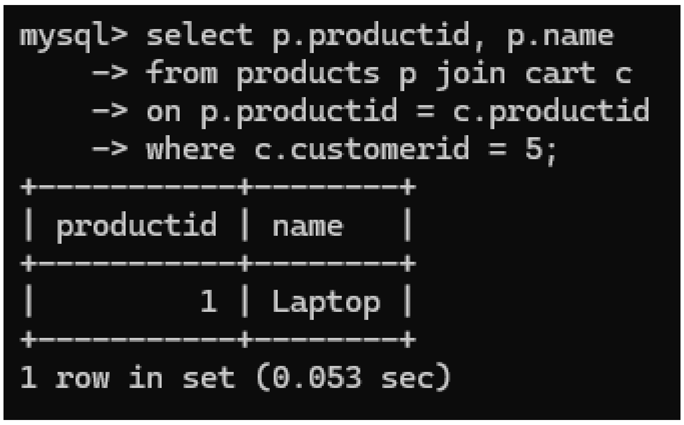
*select * from products where name like “%r”;*



```
mysql> select * from products where name like "%r";
+-----------+----------------+-----------------------+--------+---------------+
| productid | 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 |
+-----------+----------------+-----------------------+--------+---------------+
4 rows in set (0.012 sec)
```

7. Retrieve Cart Items for Customer 5.

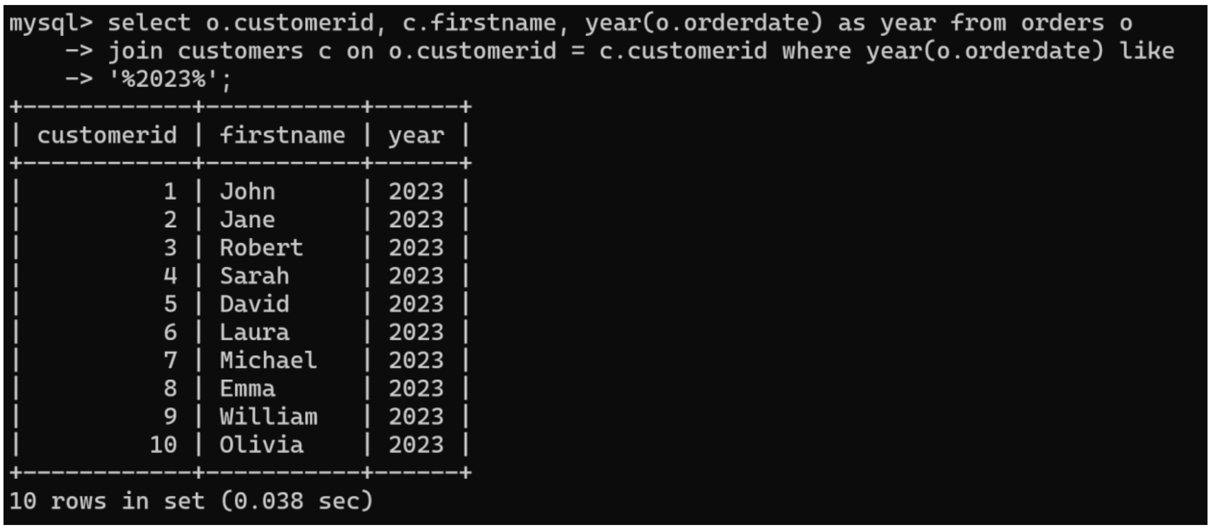
select p.productid, p.name from products p join cart c on p.productid = c.productid where c.customerid = 5;



```
mysql> select p.productid, p.name
    -> from products p join cart c
    -> on p.productid = c.productid
    -> where c.customerid = 5;
+-----------+--------+
| productid | name   |
+-----------+--------+
|         1 | Laptop |
+-----------+--------+
1 row in set (0.053 sec)
```

8. Find Customers Who Placed Orders in 2023.

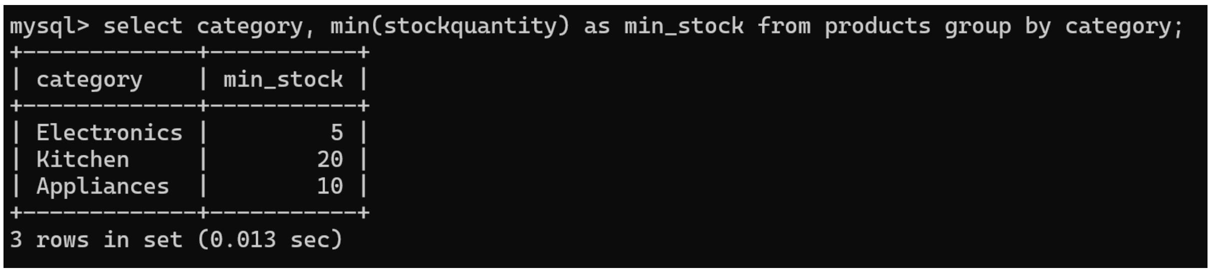
select o.customerid, c.firstname, year(o.orderdate) as year from orders o join customers c on o.customerid = c.customerid where year(o.orderdate) like "%2023%";



```
mysql> select o.customerid, c.firstname, year(o.orderdate) as year from orders o
    -> join customers c on o.customerid = c.customerid where year(o.orderdate) like
    -> '%2023%';
+------------+-----------+------+
| customerid | firstname | year |
+------------+-----------+------+
|          1 | John      | 2023 |
|          2 | Jane      | 2023 |
|          3 | Robert    | 2023 |
|          4 | Sarah     | 2023 |
|          5 | David     | 2023 |
|          6 | Laura     | 2023 |
|          7 | Michael   | 2023 |
|          8 | Emma      | 2023 |
|          9 | William   | 2023 |
|         10 | Olivia    | 2023 |
+------------+-----------+------+
10 rows in set (0.038 sec)
```

9. Determine the Minimum Stock Quantity for Each Product Category

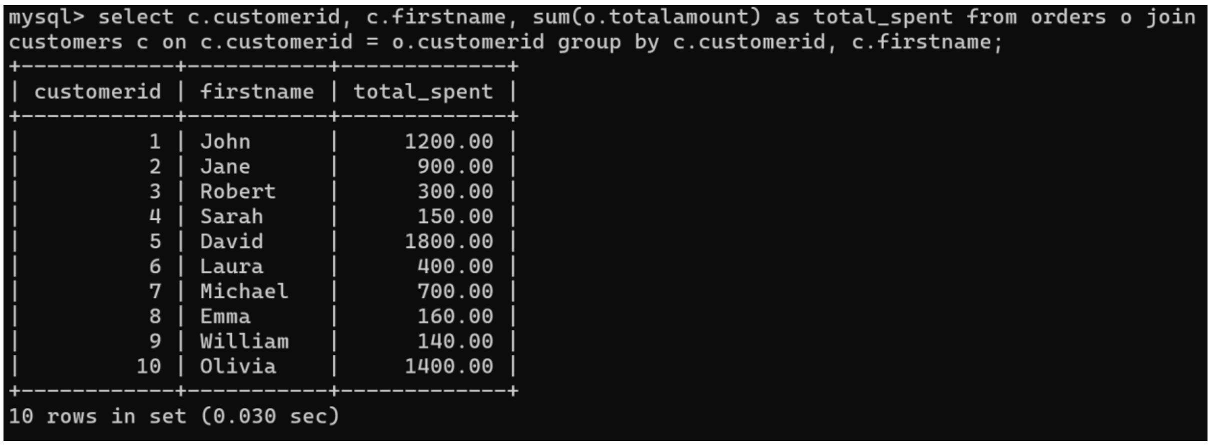
Select category, min(stockquantity) as min_stock from products group by catergory; . **(had to alter the table)**



```
mysql> select category, min(stockquantity) as min_stock from products group by category;
+-------------+-----------+
| category    | min_stock |
+-------------+-----------+
| Electronics |         5 |
| Kitchen     |        20 |
| Appliances  |        10 |
+-------------+-----------+
3 rows in set (0.013 sec)
```

10. Calculate the Total Amount Spent by Each Customer.

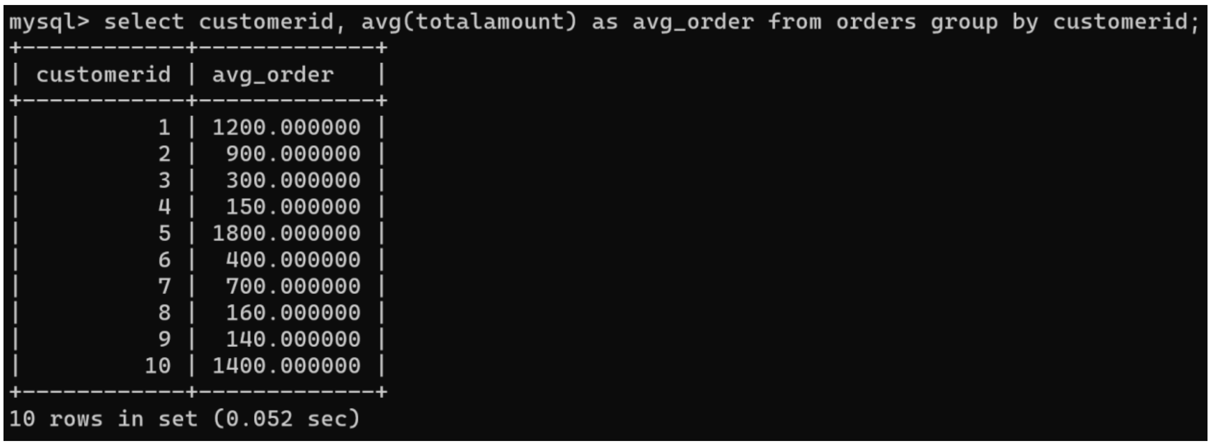
select c.customerid, c.firstname, sum(o.totalamount) as total_spent from orders o join customers c on c.customerid = o.customerid group by c,customerid, c.firstname;



```
mysql> select c.customerid, c.firstname, sum(o.totalamount) as total_spent from orders o join
customers c on c.customerid = o.customerid group by c.customerid, c.firstname;
+------------+-----------+-------------+
| customerid | firstname | total_spent |
+------------+-----------+-------------+
|          1 | John      |     1200.00 |
|          2 | Jane      |      900.00 |
|          3 | Robert    |      300.00 |
|          4 | Sarah     |      150.00 |
|          5 | David     |     1800.00 |
|          6 | Laura     |      400.00 |
|          7 | Michael   |      700.00 |
|          8 | Emma      |      160.00 |
|          9 | William   |      140.00 |
|         10 | Olivia    |     1400.00 |
+------------+-----------+-------------+
10 rows in set (0.030 sec)
```

11. Find the Average Order Amount for Each Customer.

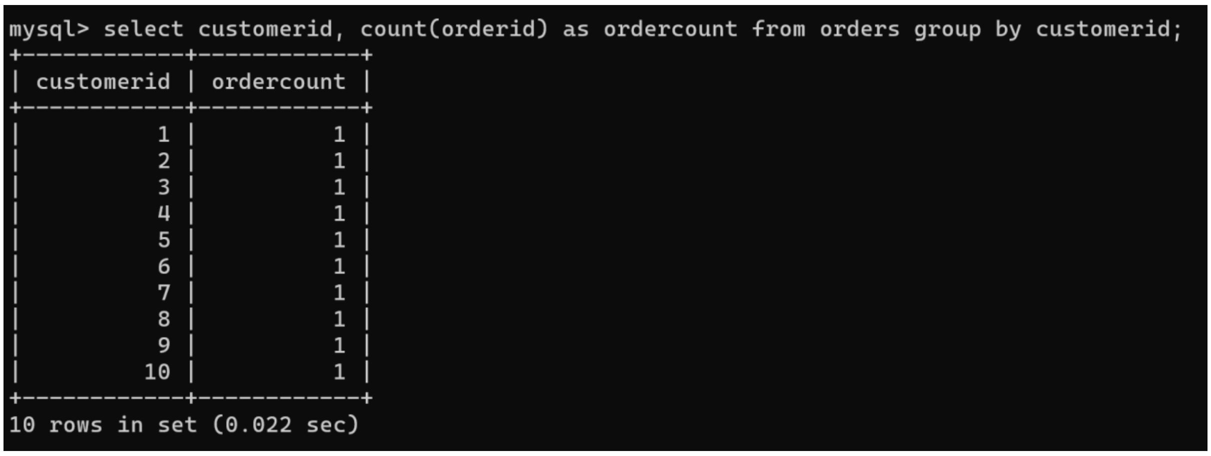
select customerid, avg(totalamount) as avg_order from orders group by customerid;



```
mysql> select customerid, avg(totalamount) as avg_order from orders group by customerid;
+------------+-------------+
| customerid | avg_order   |
+------------+-------------+
|          1 | 1200.000000 |
|          2 |  900.000000 |
|          3 |  300.000000 |
|          4 |  150.000000 |
|          5 | 1800.000000 |
|          6 |  400.000000 |
|          7 |  700.000000 |
|          8 |  160.000000 |
|          9 |  140.000000 |
|         10 | 1400.000000 |
+------------+-------------+
10 rows in set (0.052 sec)
```

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

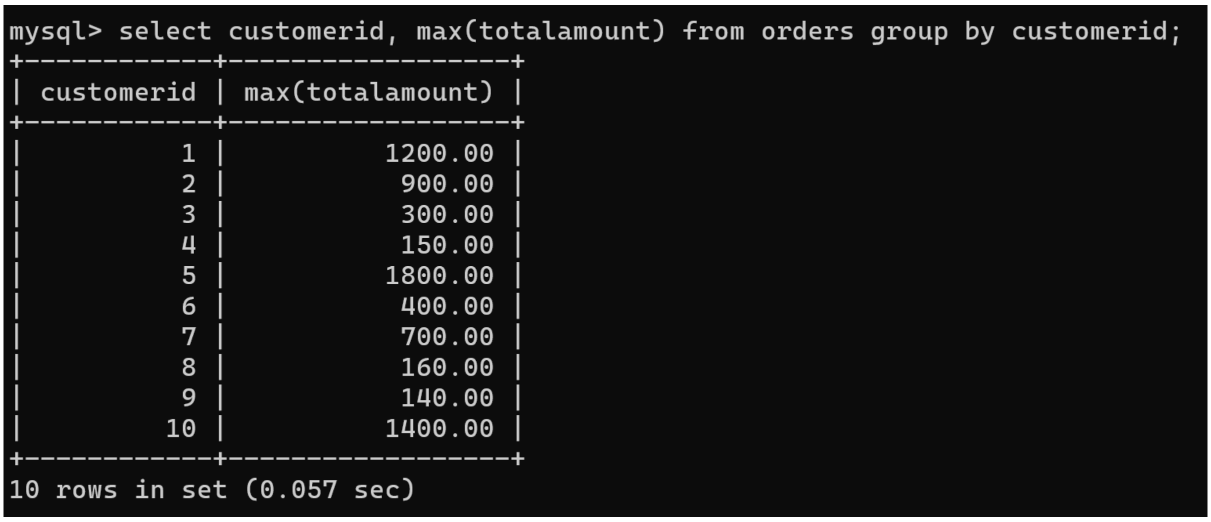
select customerid, count(orderid) as ordercount from orders group by customerid;



```
mysql> select customerid, count(orderid) as ordercount from orders group by customerid;
+------------+------------+
| customerid | ordercount |
+------------+------------+
|          1 |          1 |
|          2 |          1 |
|          3 |          1 |
|          4 |          1 |
|          5 |          1 |
|          6 |          1 |
|          7 |          1 |
|          8 |          1 |
|          9 |          1 |
|         10 |          1 |
+------------+------------+
10 rows in set (0.022 sec)
```

13. Find the Maximum Order Amount for Each Customer.

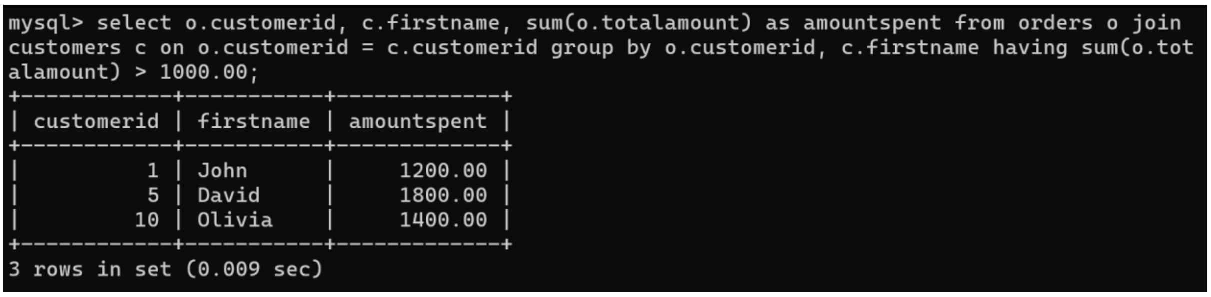
select customerid, max(totalamount) from orders group by customerid;



```
mysql> select customerid, max(totalamount) from orders group by customerid;
+------------+------------------+
| customerid | max(totalamount) |
+------------+------------------+
|          1 |          1200.00 |
|          2 |           900.00 |
|          3 |           300.00 |
|          4 |           150.00 |
|          5 |          1800.00 |
|          6 |           400.00 |
|          7 |           700.00 |
|          8 |           160.00 |
|          9 |           140.00 |
|         10 |          1400.00 |
+------------+------------------+
10 rows in set (0.057 sec)
```

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

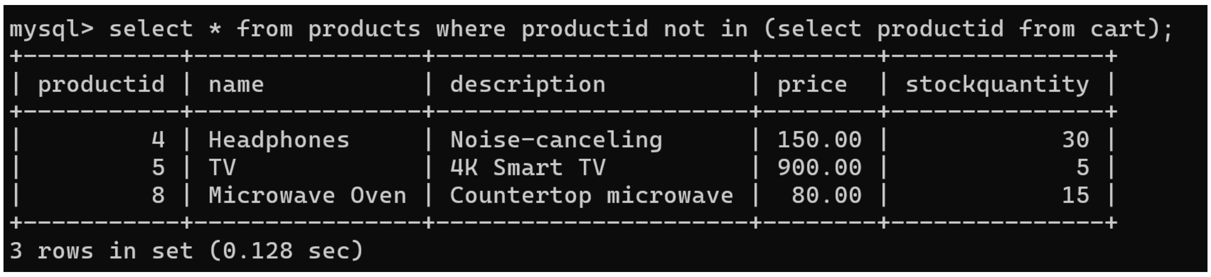
select o.customerid, c.firstname, sum(o.totalamount) as amountspent from orders o join customers c on o.customerid = c.customerid group by o.customerid, c.firstname where sum(o.totalamount) > 1000.00;



```
mysql> select o.customerid, c.firstname, sum(o.totalamount) as amountspent from orders o join
customers c on o.customerid = c.customerid group by o.customerid, c.firstname having sum(o.tot
alamount) > 1000.00;
+------------+-----------+-------------+
| customerid | firstname | amountspent |
+------------+-----------+-------------+
|          1 | John      |     1200.00 |
|          5 | David     |     1800.00 |
|         10 | Olivia    |     1400.00 |
+------------+-----------+-------------+
3 rows in set (0.009 sec)
```

15. Subquery to Find Products Not in the Cart.

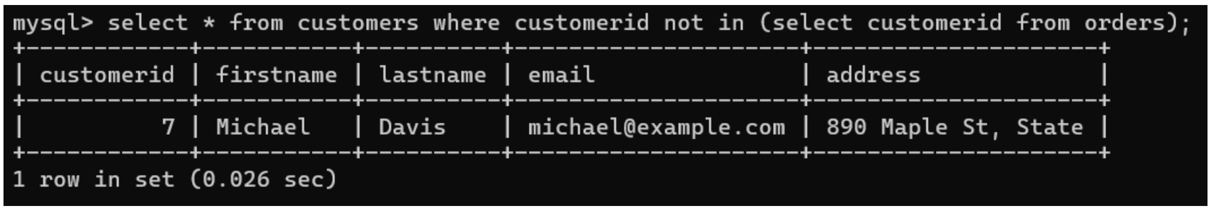
*select * from products where productid not in (select productid from cart);*



```
mysql> select * from products where productid not in (select productid from cart);
+-----------+----------------+----------------------+--------+---------------+
| productid | 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 |
+-----------+----------------+----------------------+--------+---------------+
3 rows in set (0.128 sec)
```

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

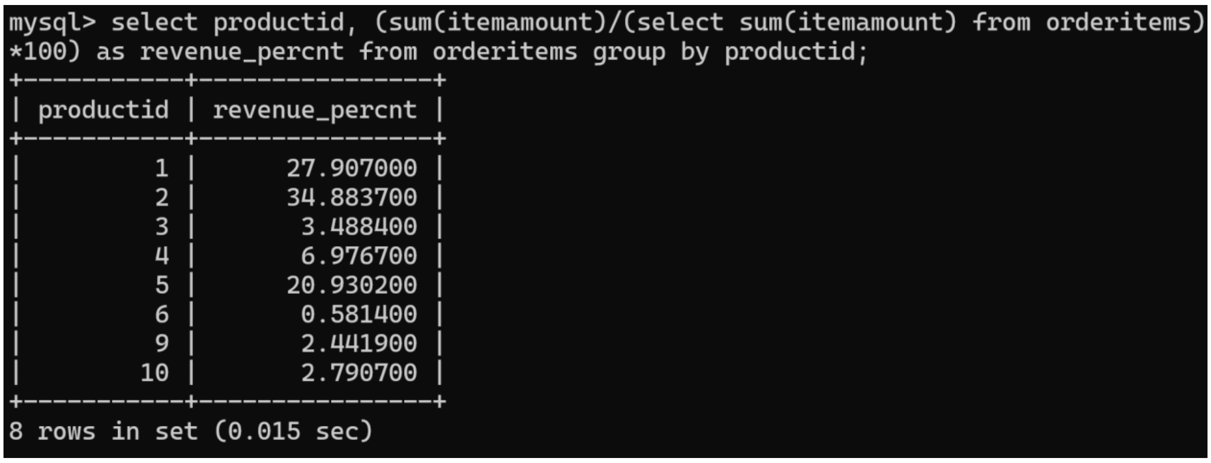
*select * from customers where customerid not in (select customerid from orders);*



```
mysql> select * from customers where customerid not in (select customerid from orders);
+------------+-----------+----------+---------------------+---------------------+
| customerid | firstname | lastname | email               | address             |
+------------+-----------+----------+---------------------+---------------------+
|          7 | Michael   | Davis    | michael@example.com | 890 Maple St, State |
+------------+-----------+----------+---------------------+---------------------+
1 row in set (0.026 sec)
```

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

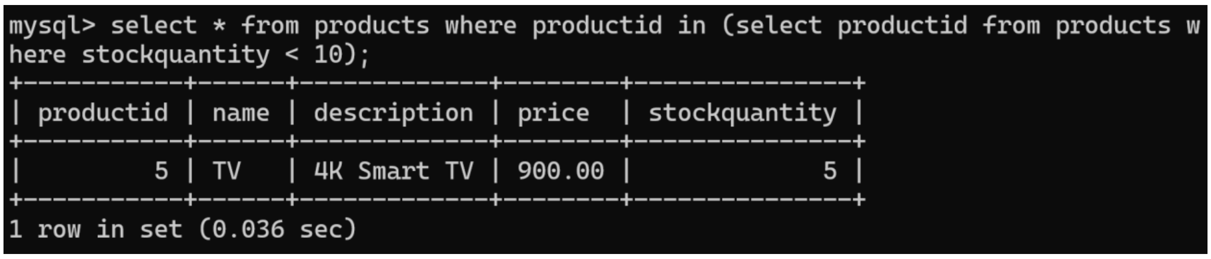
*select productid, (sum(itemamount)/(select sum(itemamount) from orderitems)*100 as revenue_percnt from orderitems group by productid;*



```
mysql> select productid, (sum(itemamount)/(select sum(itemamount) from orderitems)
*100) as revenue_percnt from orderitems group by productid;
+-----------+----------------+
| productid | revenue_percnt |
+-----------+----------------+
|         1 |      27.907000 |
|         2 |      34.883700 |
|         3 |       3.488400 |
|         4 |       6.976700 |
|         5 |      20.930200 |
|         6 |       0.581400 |
|         9 |       2.441900 |
|        10 |       2.790700 |
+-----------+----------------+
8 rows in set (0.015 sec)
```

18. Subquery to Find Products with Low Stock.

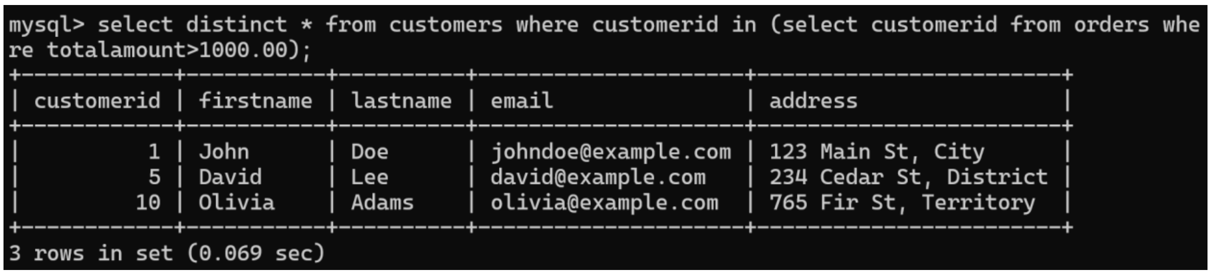
*select * from product where productid in (select productid from products where stockquantity < 10);*



```
mysql> select * from products where productid in (select productid from products w
here stockquantity < 10);
+-----------+------+-------------+--------+---------------+
| productid | name | description | price  | stockquantity |
+-----------+------+-------------+--------+---------------+
|         5 | TV   | 4K Smart TV | 900.00 |             5 |
+-----------+------+-------------+--------+---------------+
1 row in set (0.036 sec)
```

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

*select distinct * from customers where customerid in (select customerid from orders where totalamount>1000.00);*



```
mysql> select distinct * from customers where customerid in (select customerid from orders whe
re totalamount>1000.00);
+------------+-----------+----------+---------------------+------------------------+
| customerid | firstname | lastname | email               | address                |
+------------+-----------+----------+---------------------+------------------------+
|          1 | John      | Doe      | johndoe@example.com | 123 Main St, City      |
|          5 | David     | Lee      | david@example.com   | 234 Cedar St, District |
|         10 | Olivia    | Adams    | olivia@example.com  | 765 Fir St, Territory  |
+------------+-----------+----------+---------------------+------------------------+
3 rows in set (0.069 sec)
```