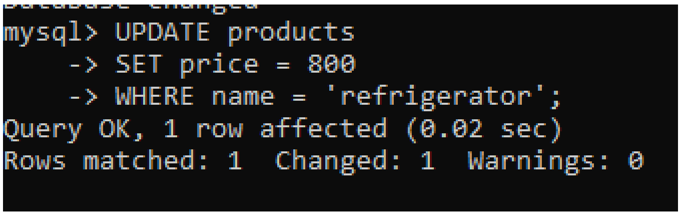
Assessments -3

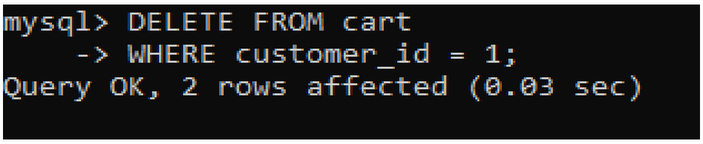
Coding Challenges - PatPals, The Pet Adoption Platform

1) Update refrigerator product price to 800.



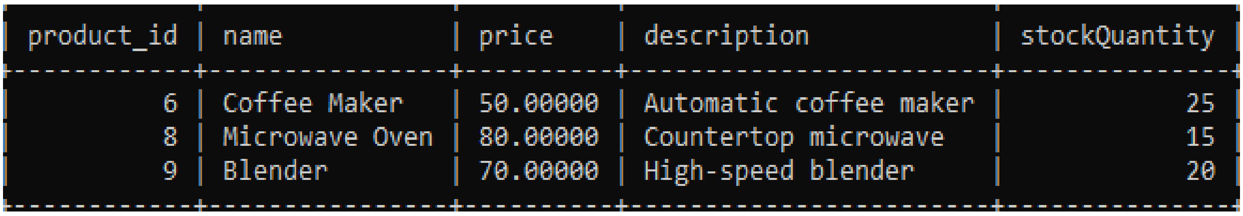
```
Database changed
mysql> UPDATE products
    -> SET price = 800
    -> WHERE name = 'refrigerator';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

2) Remove all cart items for a specific customer.



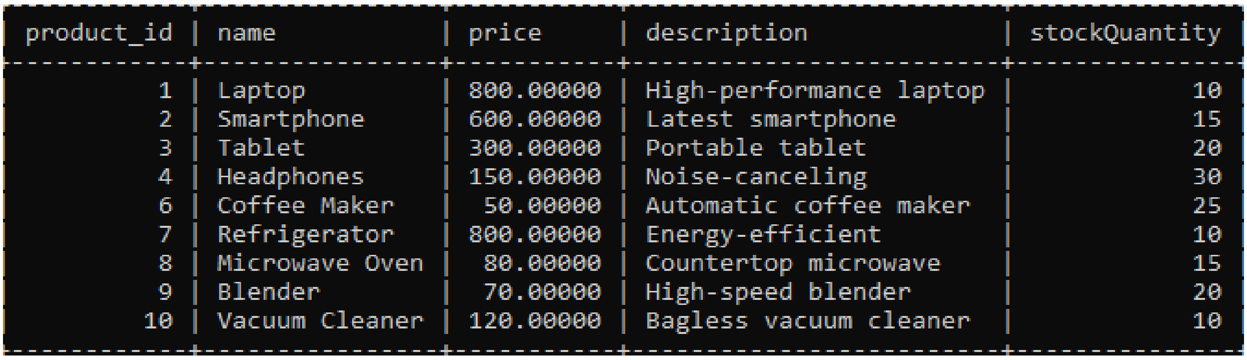
```
mysql> DELETE FROM cart
    -> WHERE customer_id = 1;
Query OK, 2 rows affected (0.03 sec)
```

3) Retrieve Products Priced Below $100.



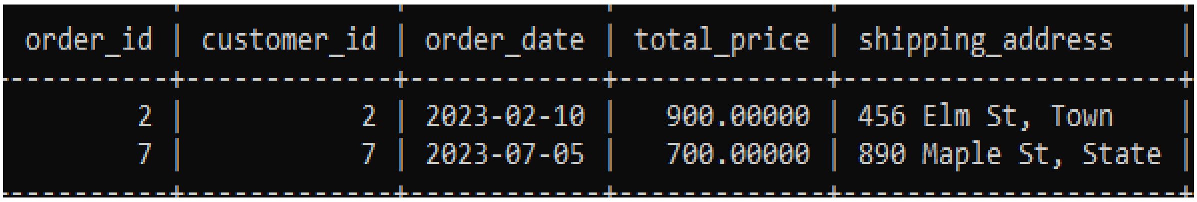
```
| product_id | name           | price    | description            | stockQuantity |
+------------+----------------+----------+------------------------+---------------+
|          6 | Coffee Maker   | 50.00000 | Automatic coffee maker |            25 |
|          8 | Microwave Oven | 80.00000 | Countertop microwave   |            15 |
|          9 | Blender        | 70.00000 | High-speed blender     |            20 |
+------------+----------------+----------+------------------------+---------------+
```

4) Find Products with Stock Quantity Greater Than 5.



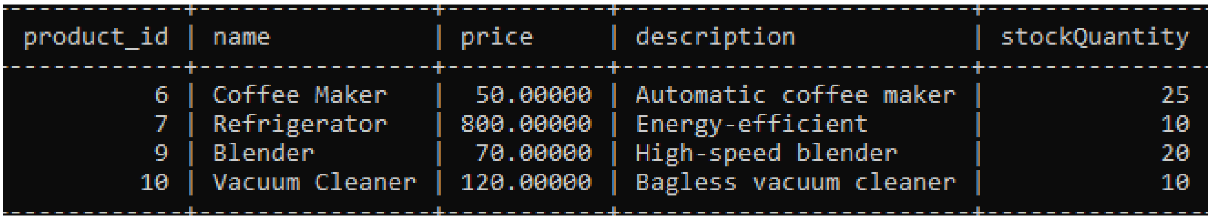
```
+------------+----------------+-----------+------------------------+---------------+
| product_id | name           | price     | description            | stockQuantity |
+------------+----------------+-----------+------------------------+---------------+
|          1 | Laptop         | 800.00000 | High-performance laptop |           10 |
|          2 | Smartphone     | 600.00000 | Latest smartphone      |            15 |
|          3 | Tablet         | 300.00000 | Portable tablet        |            20 |
|          4 | Headphones     | 150.00000 | Noise-canceling        |            30 |
|          6 | Coffee Maker   |  50.00000 | Automatic coffee maker |            25 |
|          7 | Refrigerator   | 800.00000 | Energy-efficient       |            10 |
|          8 | Microwave Oven |  80.00000 | Countertop microwave   |            15 |
|          9 | Blender        |  70.00000 | High-speed blender     |            20 |
|         10 | Vacuum Cleaner | 120.00000 | Bagless vacuum cleaner |            10 |
+------------+----------------+-----------+------------------------+---------------+
```

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



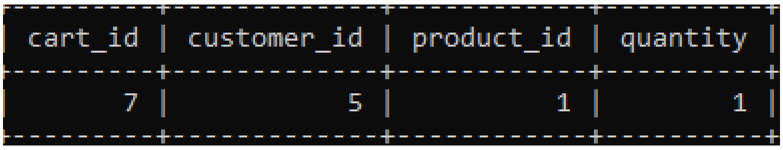
```
 order_id | customer_id | order_date | total_price | shipping_address    |
----------+-------------+------------+-------------+---------------------+
        2 |           2 | 2023-02-10 |   900.00000 | 456 Elm St, Town    |
        7 |           7 | 2023-07-05 |   700.00000 | 890 Maple St, State |
----------+-------------+------------+-------------+---------------------+
```

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



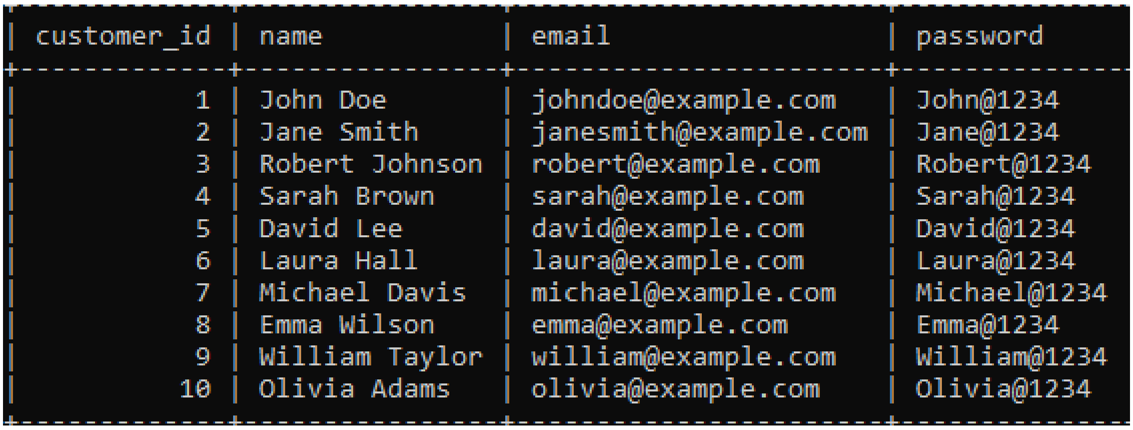
```
------------+----------------+-----------+-----------------------+--------------
 product_id | name           | price     | description           | stockQuantity
------------+----------------+-----------+-----------------------+--------------
          6 | Coffee Maker   |  50.00000 | Automatic coffee maker |            25
          7 | Refrigerator   | 800.00000 | Energy-efficient      |            10
          9 | Blender        |  70.00000 | High-speed blender    |            20
         10 | Vacuum Cleaner | 120.00000 | Bagless vacuum cleaner |            10
------------+----------------+-----------+-----------------------+--------------
```

7) Retrieve Cart Items for Customer 5.



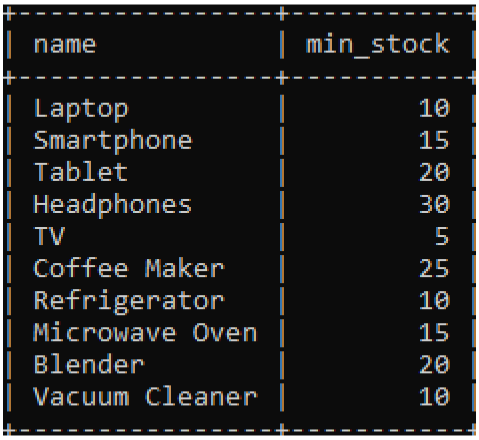
```
---------+-------------+------------+----------+
 cart_id | customer_id | product_id | quantity |
---------+-------------+------------+----------+
       7 |           5 |          1 |        1 |
---------+-------------+------------+----------+
```

8) Find Customers Who Placed Orders in 2023.



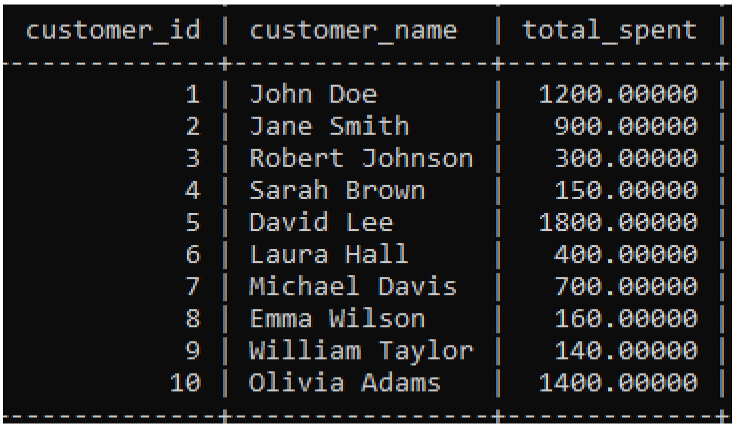
```
+-------------+----------------+-----------------------+--------------
| customer_id | name           | email                 | password
+-------------+----------------+-----------------------+--------------
|           1 | John Doe       | johndoe@example.com   | John@1234
|           2 | Jane Smith     | janesmith@example.com | Jane@1234
|           3 | Robert Johnson | robert@example.com    | Robert@1234
|           4 | Sarah Brown    | sarah@example.com     | Sarah@1234
|           5 | David Lee      | david@example.com     | David@1234
|           6 | Laura Hall     | laura@example.com     | Laura@1234
|           7 | Michael Davis  | michael@example.com   | Michael@1234
|           8 | Emma Wilson    | emma@example.com      | Emma@1234
|           9 | William Taylor | william@example.com   | William@1234
|          10 | Olivia Adams   | olivia@example.com    | Olivia@1234
+-------------+----------------+-----------------------+--------------
```

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



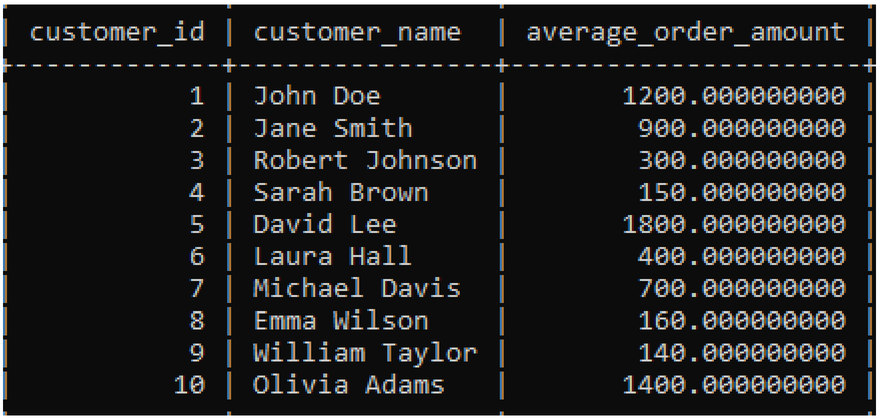
```
+----------------+-----------+
| name           | min_stock |
+----------------+-----------+
| Laptop         |        10 |
| Smartphone     |        15 |
| Tablet         |        20 |
| Headphones     |        30 |
| TV             |         5 |
| Coffee Maker   |        25 |
| Refrigerator   |        10 |
| Microwave Oven |        15 |
| Blender        |        20 |
| Vacuum Cleaner |        10 |
+----------------+-----------+
```

10) Calculate the Total Amount Spent by Each Customer.



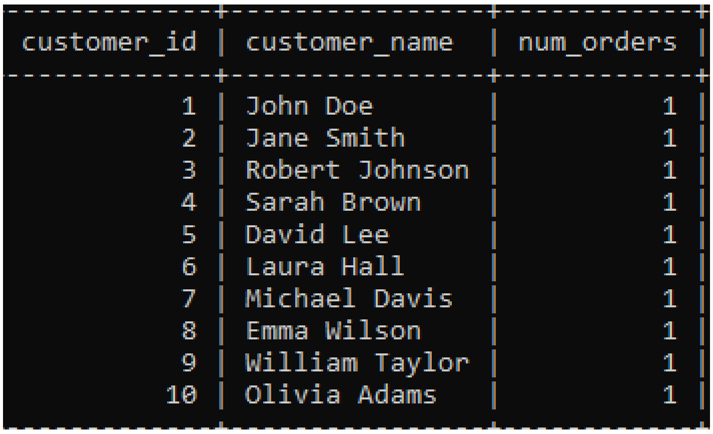
```
 customer_id | customer_name   | total_spent |
-------------+-----------------+-------------+
           1 | John Doe        |  1200.00000 |
           2 | Jane Smith      |   900.00000 |
           3 | Robert Johnson  |   300.00000 |
           4 | Sarah Brown     |   150.00000 |
           5 | David Lee       |  1800.00000 |
           6 | Laura Hall      |   400.00000 |
           7 | Michael Davis   |   700.00000 |
           8 | Emma Wilson     |   160.00000 |
           9 | William Taylor  |   140.00000 |
          10 | Olivia Adams    |  1400.00000 |
-------------+-----------------+-------------+
```

11) Find the Average Order Amount for Each Customer.



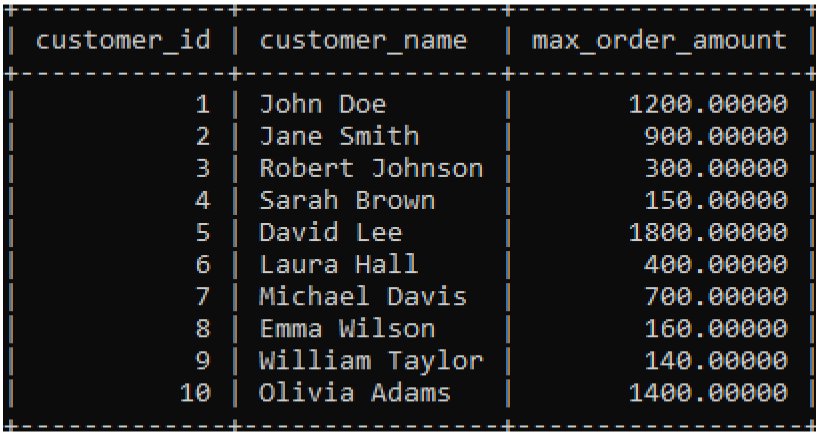
```
| customer_id | customer_name   | average_order_amount |
+-------------+-----------------+----------------------+
|           1 | John Doe        |       1200.000000000 |
|           2 | Jane Smith      |        900.000000000 |
|           3 | Robert Johnson  |        300.000000000 |
|           4 | Sarah Brown     |        150.000000000 |
|           5 | David Lee       |       1800.000000000 |
|           6 | Laura Hall      |        400.000000000 |
|           7 | Michael Davis   |        700.000000000 |
|           8 | Emma Wilson     |        160.000000000 |
|           9 | William Taylor  |        140.000000000 |
|          10 | Olivia Adams    |       1400.000000000 |
```

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



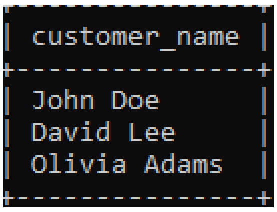
```
+-------------+-----------------+------------+
| customer_id | customer_name   | num_orders |
+-------------+-----------------+------------+
|           1 | John Doe        |          1 |
|           2 | Jane Smith      |          1 |
|           3 | Robert Johnson  |          1 |
|           4 | Sarah Brown     |          1 |
|           5 | David Lee       |          1 |
|           6 | Laura Hall      |          1 |
|           7 | Michael Davis   |          1 |
|           8 | Emma Wilson     |          1 |
|           9 | William Taylor  |          1 |
|          10 | Olivia Adams    |          1 |
+-------------+-----------------+------------+
```

13) Find the Maximum Order Amount for Each Customer.



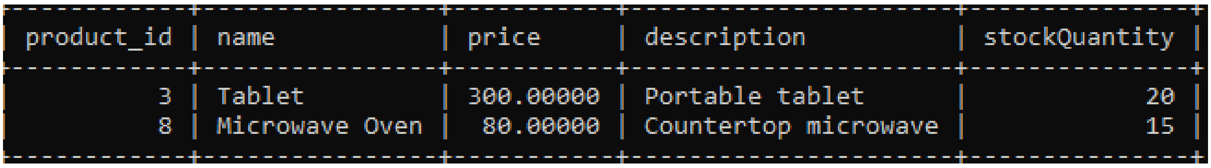
```
+-------------+-----------------+------------------+
| customer_id | customer_name   | max_order_amount |
+-------------+-----------------+------------------+
|           1 | John Doe        |       1200.00000 |
|           2 | Jane Smith      |        900.00000 |
|           3 | Robert Johnson  |        300.00000 |
|           4 | Sarah Brown     |        150.00000 |
|           5 | David Lee       |       1800.00000 |
|           6 | Laura Hall      |        400.00000 |
|           7 | Michael Davis   |        700.00000 |
|           8 | Emma Wilson     |        160.00000 |
|           9 | William Taylor  |        140.00000 |
|          10 | Olivia Adams    |       1400.00000 |
+-------------+-----------------+------------------+
```

14) Get Customers Who Placed Orders Totaling Over $1000



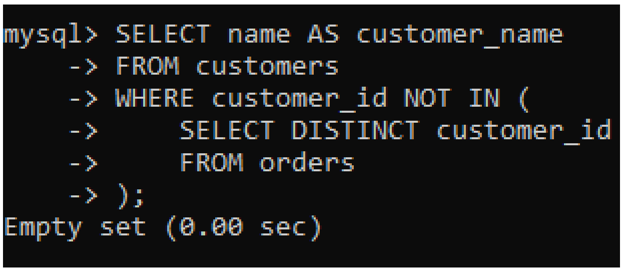
```
+---------------+
| customer_name |
+---------------+
| John Doe      |
| David Lee     |
| Olivia Adams  |
+---------------+
```

15) Subquery to Find Products Not in the Cart.



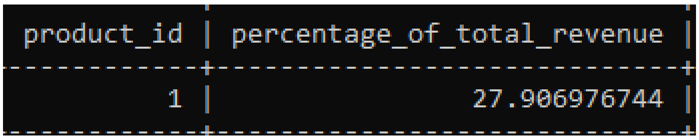
```
+------------+----------------+-----------+----------------------+---------------+
| product_id | name           | price     | description          | stockQuantity |
+------------+----------------+-----------+----------------------+---------------+
|          3 | Tablet         | 300.00000 | Portable tablet      |            20 |
|          8 | Microwave Oven |  80.00000 | Countertop microwave |            15 |
+------------+----------------+-----------+----------------------+---------------+
```

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



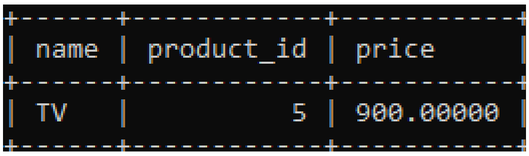
```
mysql> SELECT name AS customer_name
    -> FROM customers
    -> WHERE customer_id NOT IN (
    ->     SELECT DISTINCT customer_id
    ->     FROM orders
    -> );
Empty set (0.00 sec)
```

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



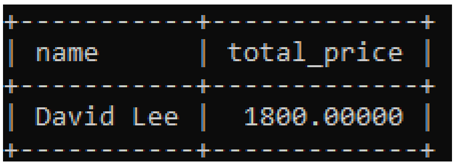
```
 product_id | percentage_of_total_revenue |
------------+-----------------------------+
          1 |                27.906976744 |
------------+-----------------------------+
```

18) Subquery to Find Products with Low Stock.



```
+------+------------+-----------+
| name | product_id | price     |
+------+------------+-----------+
| TV   |          5 | 900.00000 |
+------+------------+-----------+
```

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



```
+-----------+-------------+
| name      | total_price |
+-----------+-------------+
| David Lee |  1800.00000 |
+-----------+-------------+
```