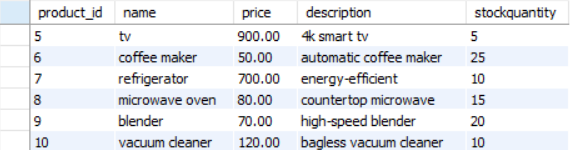
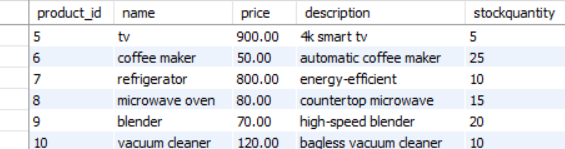
**Hexaware SQL Coding Assessment**

Date : 19-06-2025

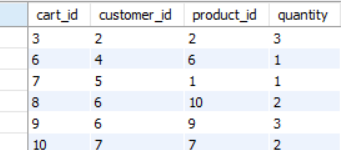
1. Update refrigerator product price to 800.

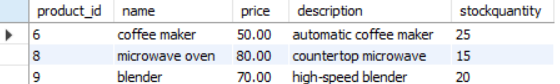
update products set price =800 where name = 'refrigerator';

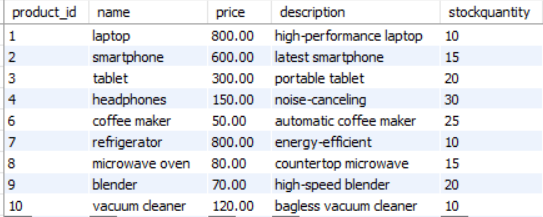
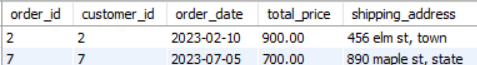
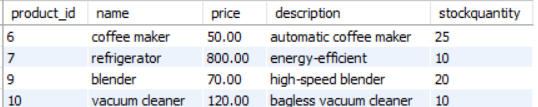
select \* from products;

1. Remove all cart items for a specific customer.  
     
    delete from cart where customer\_id =9 ;

select \* from cart;  
  
 

3. Retrieve Products Priced Below $100.  
  
 select \* from products where price < 100;  
  
 

4. Find Products with Stock Quantity Greater Than 5.   
  
 select \* from products where stockquantity > 5;  
  
   
  
5. Retrieve Orders with Total Amount Between $500 and $1000.   
  
 select \* from orders where total\_price between 500 and 1000;  
  
   
  
  
6. Find Products which name end with letter ‘r’.   
  
 select \* from products where name like '%r';  
  
 

7. Retrieve Cart Items for Customer 5.   
  
 select \* from cart where customer\_id = 5;

  
  
  
  
  
  
  
  
  
  
  
8. Find Customers Who Placed Orders in 2023.  
   
 select distinct c.customer\_id, c.name from customers c

join orders o on c.customer\_id = o.customer\_id

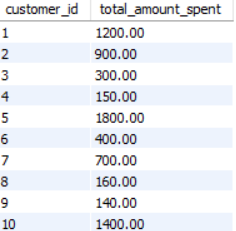
where year(o.order\_date) = 2023;

  
  
  
 9. Determine the Minimum Stock Quantity for Each Product Category.   
  
 select name, min(stockquantity) as min\_stock

from products group by name;

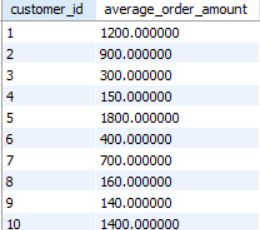
  
  
  
10. Calculate the Total Amount Spent by Each Customer.   
  
 select customer\_id, sum(total\_price) as total\_amount\_spent

from orders group by customer\_id;



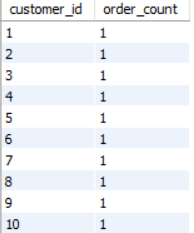
11. Find the Average Order Amount for Each Customer.  
  
 select customer\_id, avg(total\_price) as average\_order\_amount

from orders group by customer\_id;

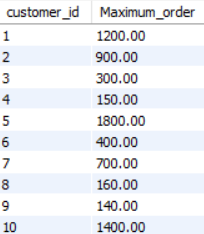


12. Count the Number of Orders Placed by Each Customer.   
  
 select customer\_id, count(\*) as order\_count

from orders group by customer\_id;

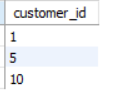
  
  
13. Find the Maximum Order Amount for Each Customer.  
  
 select customer\_id, max(total\_price) as Maximum\_order

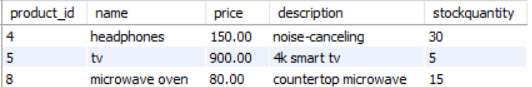
from orders group by customer\_id;



14. Get Customers Who Placed Orders Totaling Over $1000.   
  
 select customer\_id from orders group by customer\_id

having sum(total\_price) > 1000;

  
  
15. Subquery to Find Products Not in the Cart.   
  
 select \* from products where product\_id   
 not in (select distinct product\_id from cart);

  
  
16. Subquery to Find Customers Who Haven't Placed Orders.   
  
 select \* from customers where customer\_id

not in (select distinct customer\_id from orders);

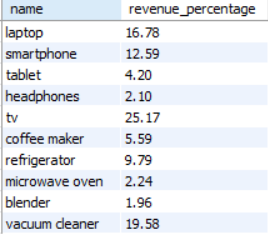
  
  
 After adding a new roe with customer\_id = 11,



17. Subquery to Calculate the Percentage of Total Revenue for a Product.  
  
 select p.name,

round(sum(o.total\_price) \* 100.0 / (select sum(total\_price) from orders), 2)

as revenue\_percentage

from orders o **join** products p on o.order\_id = p.product\_id  
 

18. Subquery to Find Products with Low Stock.   
  
 select \* from products where stockquantity < 10;



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

select distinct c.customer\_id, c.name from customers c

join orders o on c.customer\_id = o.customer\_id

where o.total\_price > 1000;

