Coding Assesment – MySql

1. Update refrigerator product price to 800.

Ans : update products set price = 800.00 where product_id = 7;

```
update products set price = 800.00 where product_id = 7;
select * from products;
```

Result Grid

product_id	name	price	description	stockQuantity
4	Headphones	150.00	Noise-canceling	30
5	TV	900.00	4K Smart TV	5
6	Coffee Maker	50.00	Automatic coffee maker	25
7	Refrigerator	800.00	Energy-efficient	10
8	Microwave Oven	80.00	Countertop microwave	15
9	Blender	70.00	High-speed blender	20

2. Remove all cart items for a specific customer.

Ans: delete from cart where customer_ID = 7

```
delete from cart where customer_ID = 7;
select * from cart;
```

Result Grid

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

3. Retrieve Products Priced Below $100.

Ans: select * from products where price < 100.00;

Query 1

```
select * from products where price < 100.00;
```

Result Grid

product_id	name	price	description	stockQuantity
6	Coffee Maker	50.00	Automatic coffee maker	25
8	Microwave Oven	80.00	Countertop microwave	15
9	Blender	70.00	High-speed blender	20
NULL	NULL	NULL	NULL	NULL

4. Find Products with Stock Quantity Greater Than 5.

Ans: select * from products where stockQuantity > 5;

Query 1

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

Result Grid

product_id	name	price	description	stockQuantity
1	Laptop	800.00	High-performance laptop	10
2	Smartphone	600.00	Latest smartphone	15
3	Tablet	300.00	Portable tablet	20
4	Headphones	150.00	Noise-canceling	30
6	Coffee Maker	50.00	Automatic coffee maker	25
7	Refrigerator	800.00	Energy-efficient	10

products 11

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

Ans: select * from orders where total_price between 500.00 and 1000.00;

Query 1

```
select * from orders where total_price between 500.00 and 1000.00;
```

Result Grid

order_id	customer_id	order_date	total_price	shipping_address
2	2	2023-02-10	900.00	NULL
7	7	2023-07-05	700.00	NULL
NULL	NULL	NULL	NULL	NULL

orders 13

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

Ans: select * from products where name like '%r';

Query 1

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

Result Grid

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

products 14

7. Retrieve Cart Items for Customer 5.

Ans: select * from cart where customer_ID = 5;

Query 1

```
select * from cart where customer_ID = 5;
```

Result Grid

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

cart 15

8. Find Customers Who Placed Orders in 2023.

Ans: select distinct c.* from customers c inner join orders o on c.customer_ID = o.customer_ID where year(o.order_Date) = 2023;

Query 1

```
select distinct c.* from customers c inner join orders o on c.customer_ID = o.customer_ID where year(o.order_Date) = 2023;
```

Result Grid

customer_id	Firstname	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

Result 21

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

Ans: select description, MIN(stockQuantity) as min_stock_quantity from products group by description;

Query 1

```
select description, MIN(stockQuantity) as min_stock_quantity from products group by description;
```

Result Grid

description	min_stock_quantity
High-performance laptop	10
Latest smartphone	15
Portable tablet	20
Noise-canceling	30
4K Smart TV	5
Automatic coffee maker	25

Result 24

10. Calculate the Total Amount Spent by Each Customer.

Ans: select c.customer_ID, c.firstName, c.last_Name, SUM(o.total_price) AS total_amount_spent from customers c join orders o on c.customer_ID = o.customer_ID group by c.customer_ID, c.firstName, c.last_Name;

Query 1

```
select c.customer_ID, c.firstName, c.last_Name, SUM(o.total_price) AS total_amount_spent from customers c join orders o on c.customer_ID = o.customer_ID group by c.customer_ID, c.firstName, c.last_Nam
```

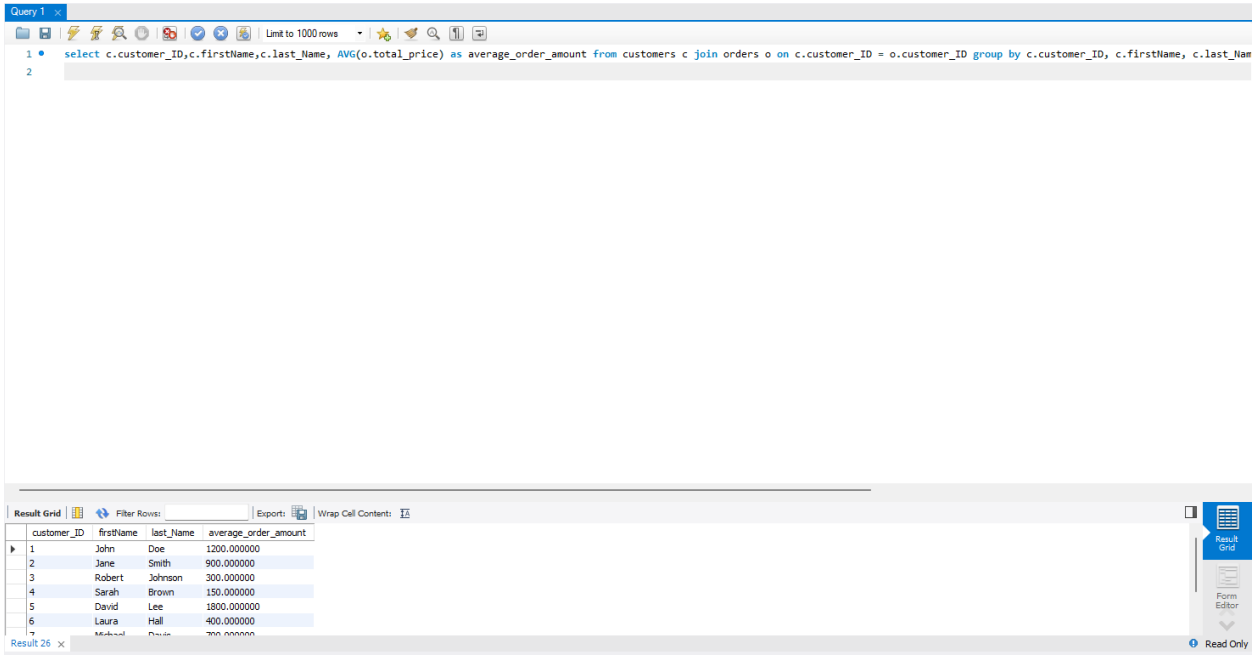
Result Grid

customer_ID	firstName	last_Name	total_amount_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

Result 25

11. Find the Average Order Amount for Each Customer.

Ans: select c.customer_ID,c.firstName,c.last_Name, AVG(o.total_price) as average_order_amount from customers c join orders o on c.customer_ID = o.customer_ID group by c.customer_ID, c.firstName, c.last_Name;

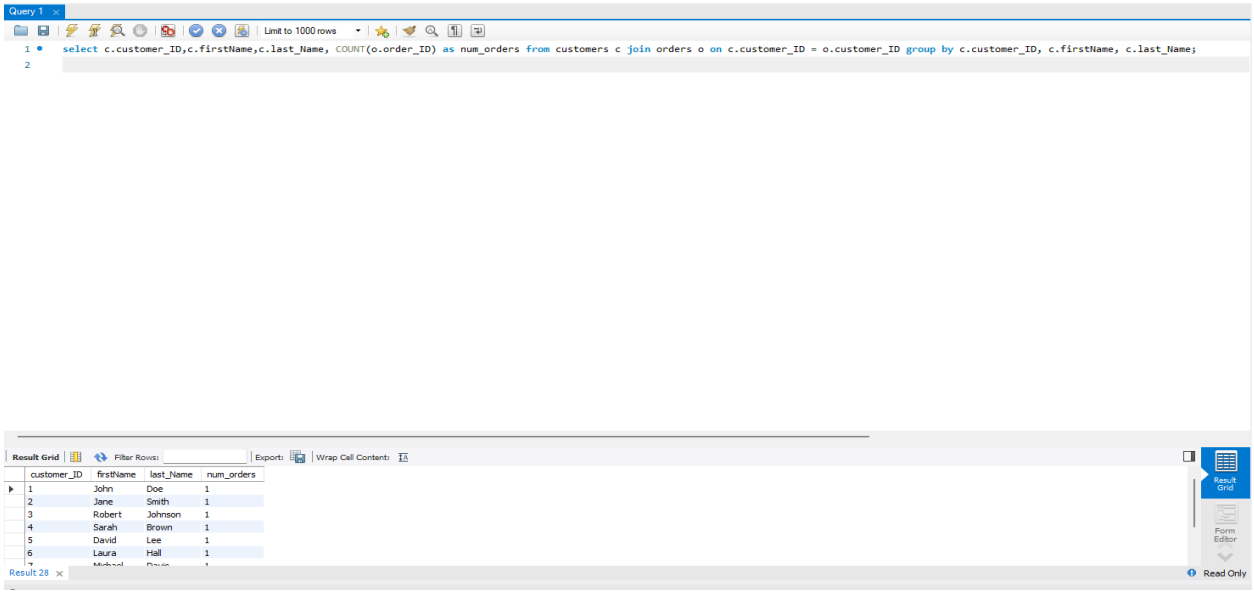


```
select c.customer_ID,c.firstName,c.last_Name, AVG(o.total_price) as average_order_amount from customers c join orders o on c.customer_ID = o.customer_ID group by c.customer_ID, c.firstName, c.last_Nam
```

customer_ID	firstName	last_Name	average_order_amount
1	John	Doe	1200.000000
2	Jane	Smith	900.000000
3	Robert	Johnson	300.000000
4	Sarah	Brown	150.000000
5	David	Lee	1800.000000
6	Laura	Hall	400.000000

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

Ans: select c.customer_ID,c.firstName,c.last_Name, COUNT(o.order_ID) as num_orders from customers c join orders o on c.customer_ID = o.customer_ID group by c.customer_ID, c.firstName, c.last_Name;

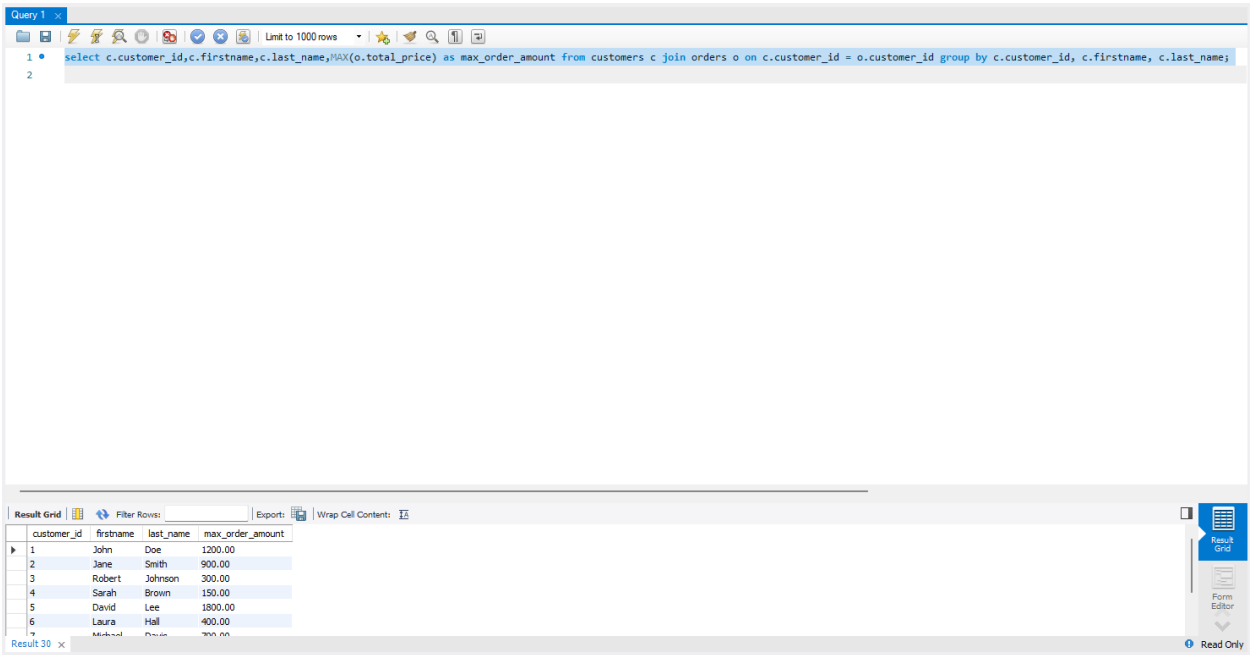


```
select c.customer_ID,c.firstName,c.last_Name, COUNT(o.order_ID) as num_orders from customers c join orders o on c.customer_ID = o.customer_ID group by c.customer_ID, c.firstName, c.last_Name;
```

customer_ID	firstName	last_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

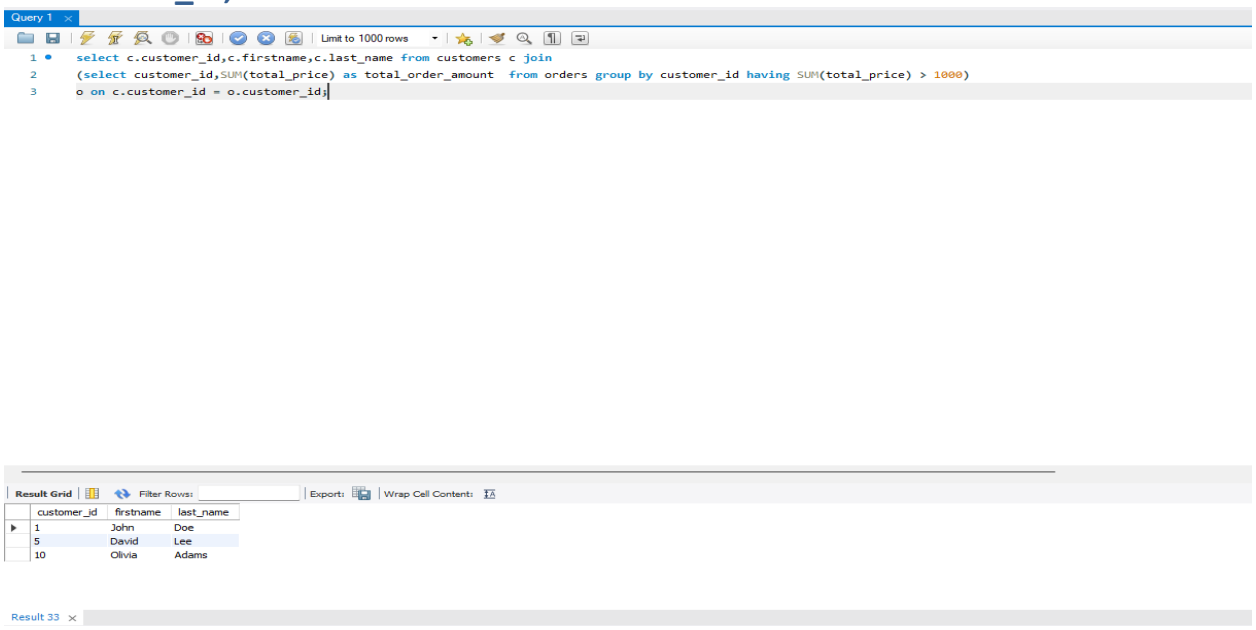
13. Find the Maximum Order Amount for Each Customer.

Ans: select c.customer_id,c.firstname,c.last_name,MAX(o.total_price) as max_order_amount from customers c join orders o on c.customer_id = o.customer_id group by c.customer_id, c.firstname, c.last_name;



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

Ans: select c.customer_id,c.firstname,c.last_name from customers c join (select customer_id,SUM(total_price) as total_order_amount from orders group by customer_id having SUM(total_price) > 1000) o on c.customer_id = o.customer_id;



15. Subquery to Find Products Not in the Cart.

Ans: select * from products where product_ID not in (select distinct product_ID from cart);

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

product_id	name	price	description	stockQuantity
7	Refrigerator	800.00	Energy-efficient	10
8	Microwave Oven	80.00	Countertop microwave	15
NULL	NULL	NULL	NULL	NULL

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

Ans: select * from customers where customer_id not in (select distinct customer_id from orders);

```
select * from customers where customer_id not in (
select distinct customer_id from orders);
```

customer_id	Firstname	last_name	email	address
NULL	NULL	NULL	NULL	NULL

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

Ans: SELECT p.product_id,p.name AS product_name, p.price AS unit_price, SUM(oi.quantity) AS total_quantity_sold,SUM(oi.quantity * p.price) AS total_revenue,

(SUM(oi.quantity * p.price) / total_orders.total_revenue) * 100 AS percentage_of_total_revenue

FROM products p JOIN order_items oi ON p.product_id = oi.product_id

JOIN (SELECT SUM(oi.quantity * p.price) AS total_revenue FROM order_items oi JOIN products p ON oi.product_id = p.product_id) AS total_orders

GROUP BY p.product_id, p.name, p.price, total_orders.total_revenue ORDER BY total_revenue DESC;

Query 1 order_items

Limit to 1000 rows

```
1 • SELECT p.product_id,p.name AS product_name, p.price AS unit_price, SUM(oi.quantity) AS total_quantity_sold,SUM(oi.quantity * p.price) AS total_revenue,
2   (SUM(oi.quantity * p.price) / total_orders.total_revenue) * 100 AS percentage_of_total_revenue
3   FROM products p JOIN order_items oi ON p.product_id = oi.product_id
4   JOIN (SELECT SUM(oi.quantity * p.price) AS total_revenue FROM order_items oi JOIN products p ON oi.product_id = p.product_id) AS total_orders
5   GROUP BY p.product_id, p.name, p.price, total_orders.total_revenue ORDER BY total_revenue DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_id	product_name	unit_price	total_quantity_sold	total_revenue	percentage_of_total_revenue
2	Smartphone	600.00	5	3000.00	34.883721
1	Laptop	800.00	3	2400.00	27.906977
5	TV	900.00	2	1800.00	20.930233
4	Headphones	150.00	4	600.00	6.976744
3	Tablet	300.00	1	300.00	3.488372
10	Vacuum Cleaner	120.00	2	240.00	2.790698

Result 46

18. Subquery to Find Products with Low Stock.

Ans : select p.product_id,p.name AS product_name,p.stockQuantity

from products p where p.stockQuantity < (select AVG(stockQuantity) from products)order by p.stockQuantity ASC;

```
select p.product_id,p.name AS product_name,p.stockQuantity
from products p where p.stockQuantity < (select AVG(stockQuantity) from products)
order by p.stockQuantity ASC;
```

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

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

Ans: SELECT customer_id, firstname, last_name FROM customers WHERE customer_id IN (SELECT customer_id FROM orders GROUP BY customer_id HAVING AVG(total_price) > (SELECT AVG(total_price) FROM orders));

```
SELECT customer_id, firstname, last_name FROM customers WHERE customer_id IN (
    SELECT customer_id FROM orders GROUP BY customer_id HAVING AVG(total_price) > (
        SELECT AVG(total_price) FROM orders)
);
```

customer_id	firstname	last_name
1	John	Doe
2	Jane	Smith
5	David	Lee
10	Olivia	Adams