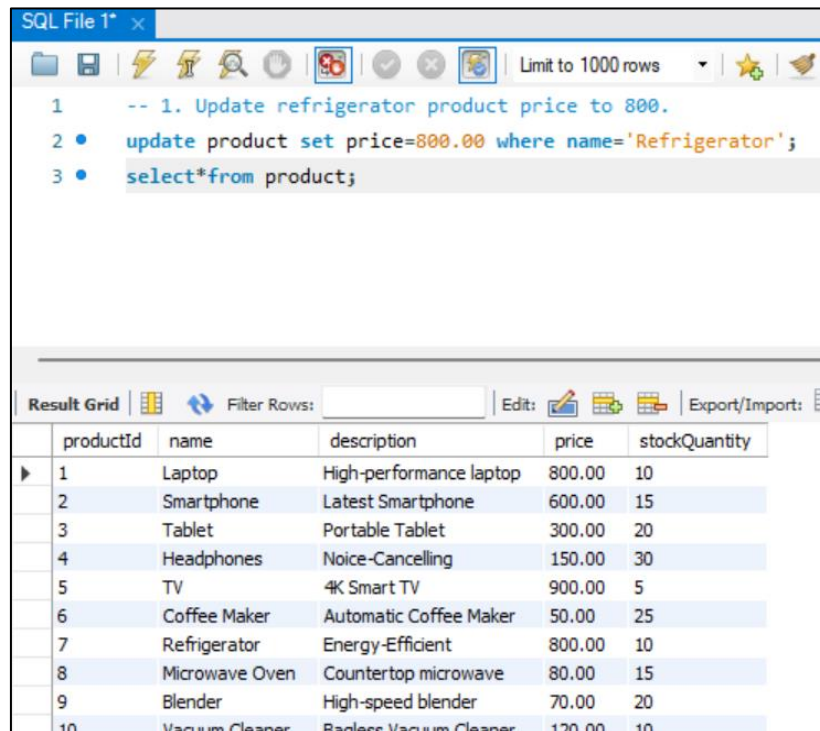


SQL - Coding Challenge – Ecom

1. Update refrigerator product price to 800.

update product set price=800.00 where name='Refrigerator';



The screenshot shows an SQL IDE window titled "SQL File 1* x". The code editor contains the following SQL statements:

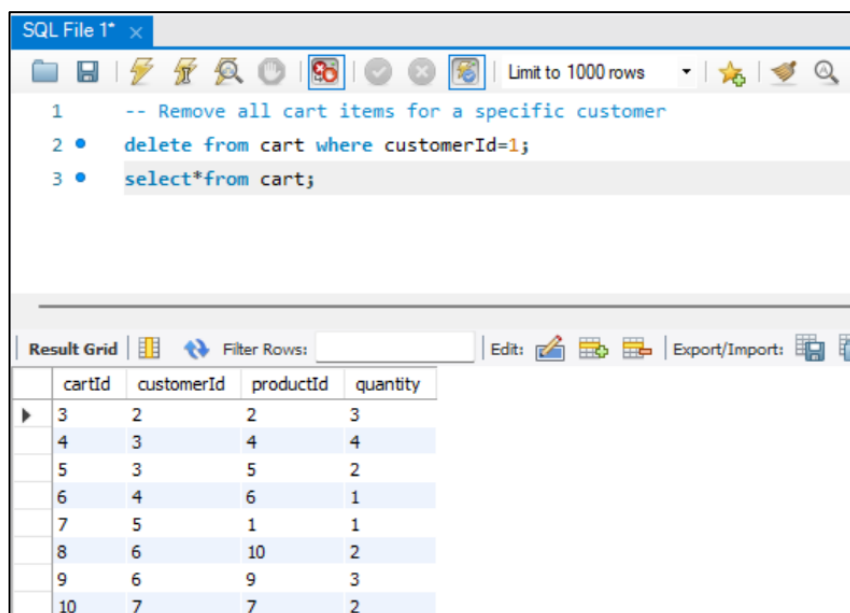
```
1 -- 1. Update refrigerator product price to 800.
2 • update product set price=800.00 where name='Refrigerator';
3 • select * from product;
```

Below the code editor is the "Result Grid" showing the output of the SELECT statement. The grid has columns: productId, name, description, price, and stockQuantity. The data is as follows:

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	Noice-Cancelling	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

2. Remove all cart items for a specific customer.

delete from cart where customerId=1;



The screenshot shows an SQL IDE window titled "SQL File 1* x". The code editor contains the following SQL statements:

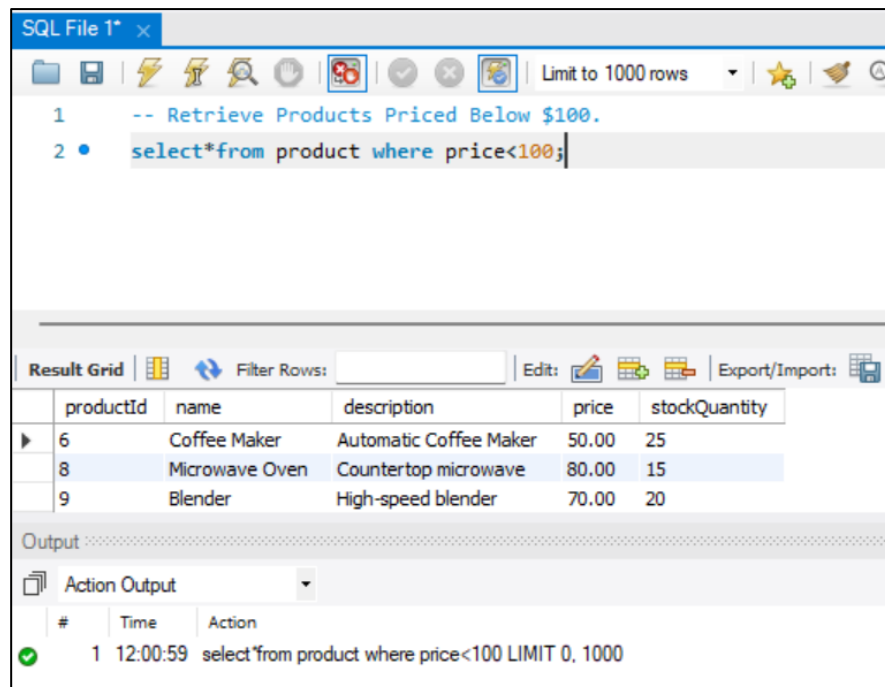
```
1 -- Remove all cart items for a specific customer
2 • delete from cart where customerId=1;
3 • select * from cart;
```

Below the code editor is the "Result Grid" showing the output of the SELECT statement. The grid has columns: cartId, customerId, productId, and quantity. The data is as follows:

cartId	customerId	productId	quantity
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

3. Retrieve Products Priced Below \$100.

`select*from product where price<100;`



SQL File 1* x

Limit to 1000 rows

```
1 -- Retrieve Products Priced Below $100.
2 • select*from product where price<100;
```

Result Grid

	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

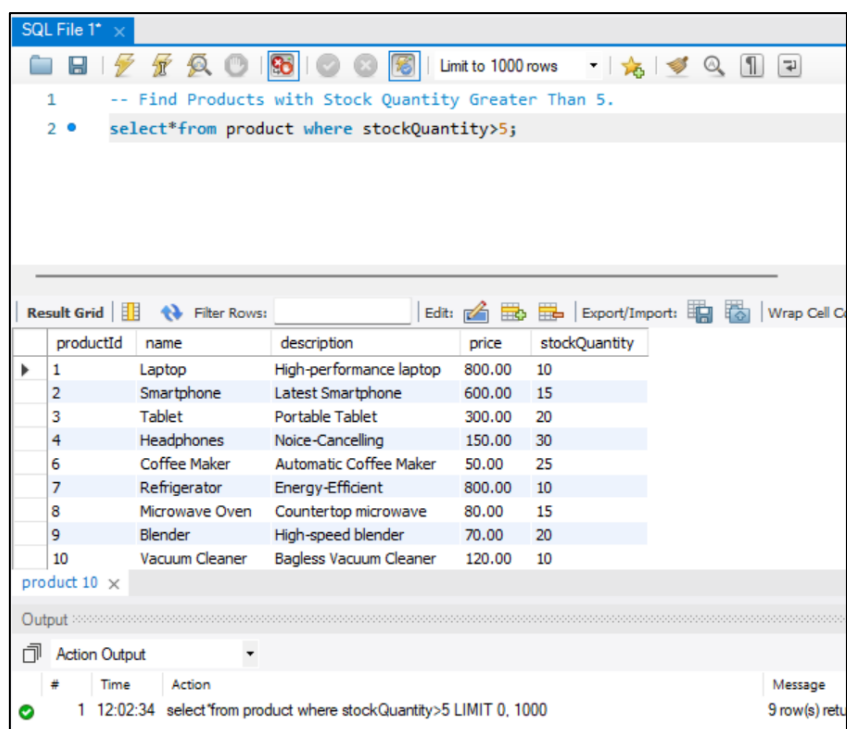
Output

Action Output

#	Time	Action
✓ 1	12:00:59	select*from product where price<100 LIMIT 0, 1000

4. Find Products with Stock Quantity Greater Than 5.

`select*from product where stockQuantity>5;`



SQL File 1* x

Limit to 1000 rows

```
1 -- Find Products with Stock Quantity Greater Than 5.
2 • select*from product where stockQuantity>5;
```

Result Grid

	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	Noice-Cancelling	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

product 10 x

Output

Action Output

#	Time	Action	Message
✓ 1	12:02:34	select*from product where stockQuantity>5 LIMIT 0, 1000	9 row(s) returned

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

`select*from orders where totalAmount between 500 and 1000;`

SQL File 1* x

Limit to 1000 rows

```
1 -- Retrieve Orders with Total Amount Between $500 and $1000
2 • select*from orders where totalAmount between 500 and 1000;
```

Result Grid

	orderId	customerId	orderDate	totalAmount
▶	2	2	2023-02-10	900.00
	7	7	2023-07-05	700.00

Output

Action Output

#	Time	Action
✓ 1	12:05:21	select*from orders where totalAmount between 500 and 1000 LIMIT 0, 1000

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

`select*from product where name like '%r';`

SQL File 1* x

Limit to 1000 rows

```
1 -- Find Products which name end with letter 'r'
2 • select*from product where name like '%r';
```

Result Grid

	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

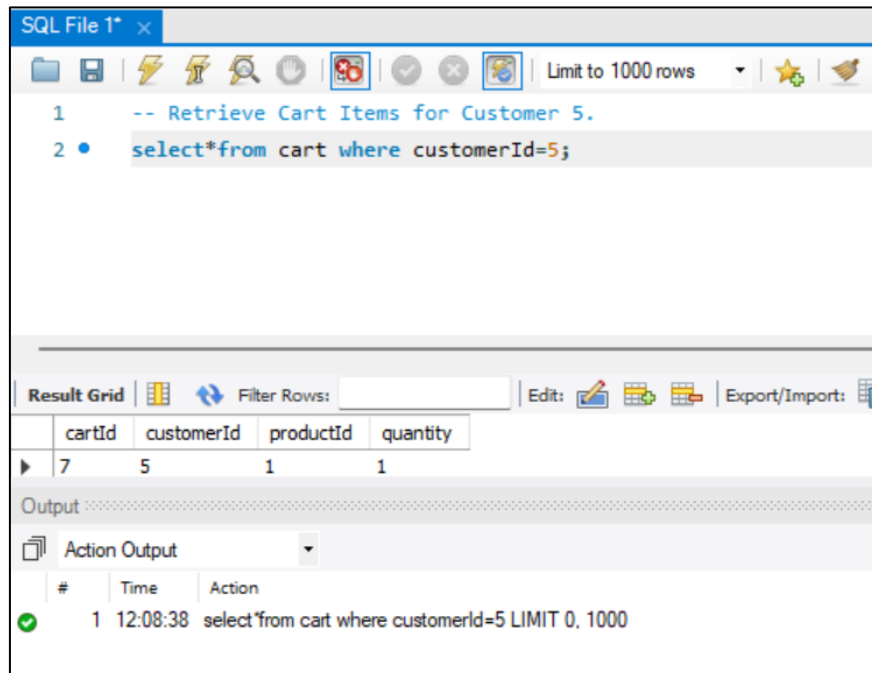
Output

Action Output

#	Time	Action
✓ 1	12:06:46	select*from product where name like '%r' LIMIT 0, 1000

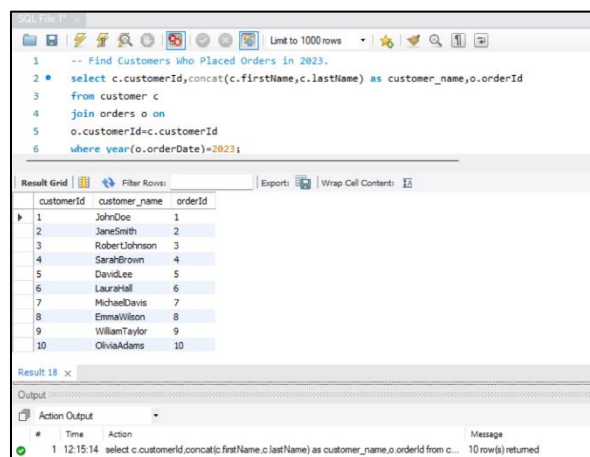
7. Retrieve Cart Items for Customer 5.

`select*from cart where customerId=5;`



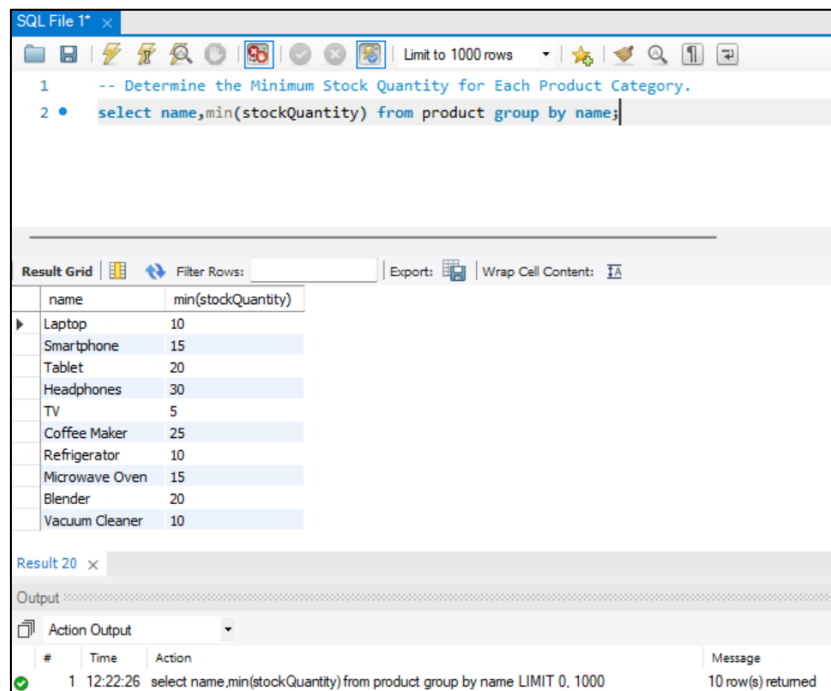
8. Find Customers Who Placed Orders in 2023

`select c.customerId,concat(c.firstName,c.lastName) as customer_name,o.orderId`
`from customer c`
`join orders o on`
`o.customerId=c.customerId`
`where year(o.orderDate)=2023;`



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

select name,min(stockQuantity) from product group by name;



The screenshot shows a SQL IDE window titled "SQL File 1*" with a toolbar and a "Limit to 1000 rows" dropdown. The SQL editor contains two lines of code: a comment and a query to find the minimum stock quantity for each product category. Below the editor is a "Result Grid" showing the results of the query. The grid has two columns: "name" and "min(stockQuantity)". The results are listed in a table with 10 rows. At the bottom, there is an "Output" section with a dropdown menu set to "Action Output". Below that is a table with columns "#", "Time", "Action", and "Message". The first row shows a successful execution of the query, returning 10 rows.

```
1 -- Determine the Minimum Stock Quantity for Each Product Category.
2 select name,min(stockQuantity) from product group by name;
```

name	min(stockQuantity)
Laptop	10
Smartphone	15
Tablet	20
Headphones	30
TV	5
Coffee Maker	25
Refrigerator	10
Microwave Oven	15
Blender	20
Vacuum Cleaner	10

Result 20 x

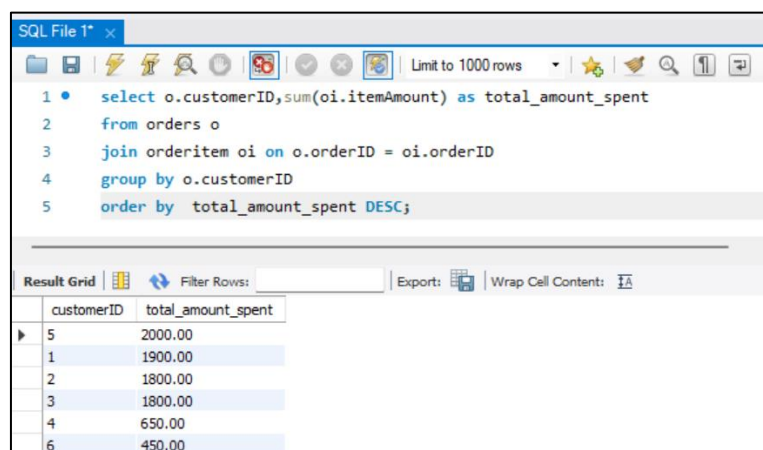
Output

Action Output

#	Time	Action	Message
1	12:22:26	select name,min(stockQuantity) from product group by name LIMIT 0, 1000	10 row(s) returned

10. Calculate the Total Amount Spent by Each Customer.

select o.customerID,sum(oi.itemAmount) as total_amount_spent
from orders o
left join orderitem oi on o.orderID = oi.orderID
group by o.customerID
order by total_amount_spent DESC;



The screenshot shows a SQL IDE window titled "SQL File 1*" with a toolbar and a "Limit to 1000 rows" dropdown. The SQL editor contains a query to calculate the total amount spent by each customer. Below the editor is a "Result Grid" showing the results of the query. The grid has two columns: "customerID" and "total_amount_spent". The results are listed in a table with 6 rows, ordered by total amount spent in descending order. At the bottom, there is an "Output" section with a dropdown menu set to "Action Output". Below that is a table with columns "#", "Time", "Action", and "Message". The first row shows a successful execution of the query.

```
1 select o.customerID,sum(oi.itemAmount) as total_amount_spent
2 from orders o
3 join orderitem oi on o.orderID = oi.orderID
4 group by o.customerID
5 order by total_amount_spent DESC;
```

customerID	total_amount_spent
5	2000.00
1	1900.00
2	1800.00
3	1800.00
4	650.00
6	450.00

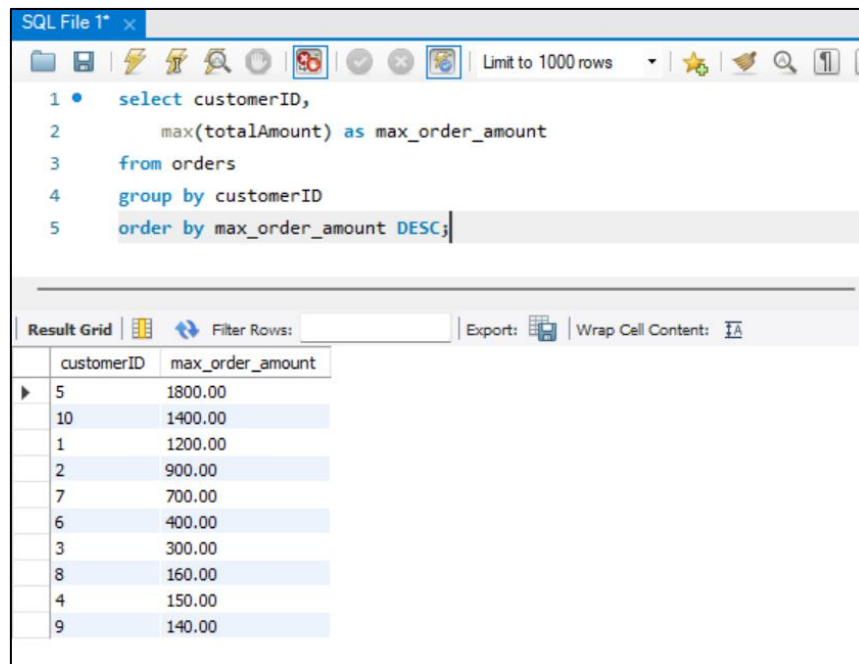
Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

customerID total_amount_spent

5	2000.00
1	1900.00
2	1800.00
3	1800.00
4	650.00
6	450.00

11. Find the Average Order Amount for Each Customer.

```
select customerID,  
max(totalAmount) as max_order_amount  
from orders  
group by customerID  
order by max_order_amount DESC;
```



The screenshot shows an SQL IDE window titled "SQL File 1*" with a toolbar at the top. The query editor contains the following SQL code:

```
1 • select customerID,  
2     max(totalAmount) as max_order_amount  
3 from orders  
4 group by customerID  
5 order by max_order_amount DESC;
```

Below the query editor is a "Result Grid" section with a toolbar. It displays the results of the query in a table with two columns: "customerID" and "max_order_amount". The results are ordered by "max_order_amount" in descending order.

customerID	max_order_amount
5	1800.00
10	1400.00
1	1200.00
2	900.00
7	700.00
6	400.00
3	300.00
8	160.00
4	150.00
9	140.00

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

```
select c.customerId,concat(c.firstName,c.lastName) as  
customer_name,count(o.orderId) as noof_orders_placed  
from customer c  
join orders o on o.customerId=c.customerId  
group by customerid  
order by noof_orders_placed;
```

SQL File 1*

```

1  -- Count the Number of Orders Placed by Each Customer.
2  • select c.customerId,concat(c.firstname,c.lastName) as customer_name,count(o.orderId) as noof_orders_placed
3  from customer c
4  join orders o on o.customerId=c.customerId
5  group by customerId
6  order by noof_orders_placed;

```

Result Grid

	customerId	customer_name	noof_orders_placed
1	JohnDoe	1	
2	JaneSmith	1	
3	RobertJohnson	1	
4	SarahBrown	1	
5	DavidLee	1	
6	LauraHall	1	
7	MichaelDavis	1	
8	EmmaWilson	1	
9	WilliamTaylor	1	
10	OliviaAdams	1	

Result 23

Output

Action Output

#	Time	Action	Message
1	12:37:28	select c.customerId,concat(c.firstname,c.lastName) as customer_name,count(o.orderId) ...	10 row(s) returned

13. Find the Maximum Order Amount for Each Customer

```

select o.customerId,max(o.totalAmount) as max_order-amount
from orders o
group by o.customerId
order by max_order_amount DESC;

```

SQL File 1*

```

1  -- Find the Maximum Order Amount for Each Customer.
2  • select o.customerID,max(o.totalAmount) as max_order_amount
3  from orders o
4  group by o.customerID
5  ORDER BY max_order_amount DESC ,customerId ASC;

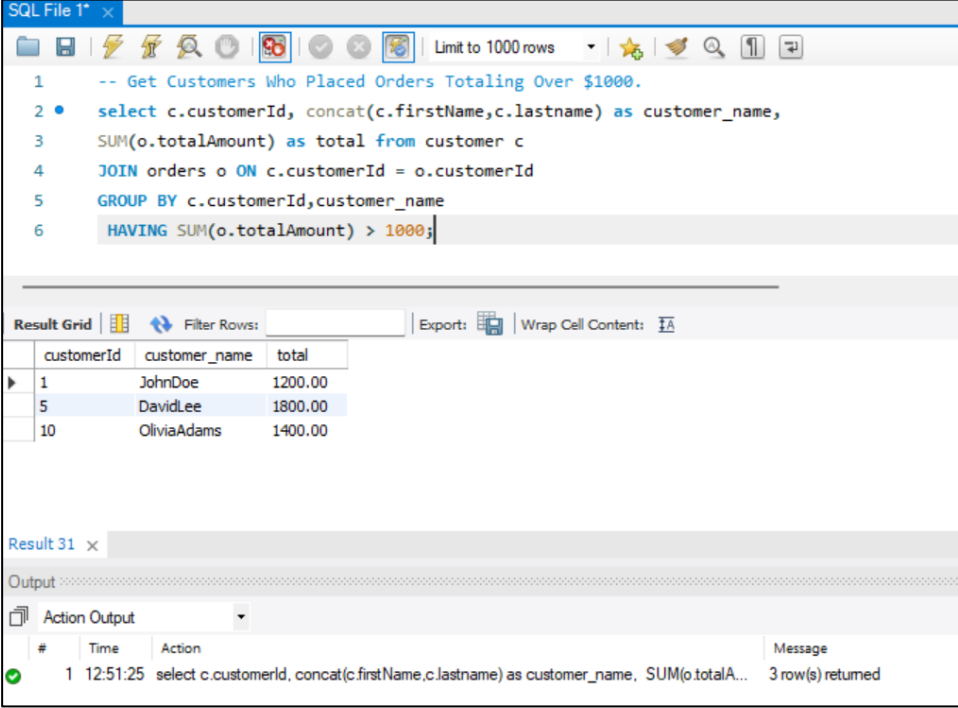
```

Result Grid

	customerID	max_order_amount
5	1800.00	
10	1400.00	
1	1200.00	
2	900.00	
7	700.00	
6	400.00	
3	300.00	
8	160.00	
4	150.00	
9	140.00	

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

```
select c.customerId,concat(c.firstName,c.lastName) as customer_name,  
sum(o.totalAmount) as total from customer c  
join orders o on c.customerId = o.customerId  
group by c.customerId,customer_name  
having sum(o.totalAmount)>1000;
```



The screenshot shows a SQL IDE window titled "SQL File 1*" with a toolbar and a query editor. The query is as follows:

```
1 -- Get Customers Who Placed Orders Totaling Over $1000.  
2 • select c.customerId, concat(c.firstName,c.lastName) as customer_name,  
3 SUM(o.totalAmount) as total from customer c  
4 JOIN orders o ON c.customerId = o.customerId  
5 GROUP BY c.customerId,customer_name  
6 HAVING SUM(o.totalAmount) > 1000;
```

Below the query editor is a "Result Grid" section with a table showing the results of the query. The table has four columns: "customerId", "customer_name", and "total". The results are as follows:

customerId	customer_name	total
1	JohnDoe	1200.00
5	DavidLee	1800.00
10	OliviaAdams	1400.00

At the bottom of the IDE, there is an "Output" section with a dropdown menu set to "Action Output". It shows a single message:

#	Time	Action	Message
1	12:51:25	select c.customerId, concat(c.firstName,c.lastName) as customer_name, SUM(o.totalA...	3 row(s) returned

15. Subquery to Find Products Not in the Cart.

```
select productid,name,price  
from product  
where productid not in(  
select distinct productId from cart);
```


SQL File 1* x

Limit to 1000 rows

```

1  -- Subquery to Find Products Not in the Cart.
2  • SELECT productId, name, price
3    FROM product
4    WHERE productId NOT IN (
5      SELECT DISTINCT productId
6      FROM cart);

```

Result Grid

productId	name	price
3	Tablet	300.00
8	Microwave Oven	80.00
NULL	NULL	NULL

product 33 x

Output

Action Output

#	Time	Action	Message
1	12:55:30	SELECT productId, name, price FROM product WHERE productId NOT IN (SELECT ...	2 row(s) returned

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

```

select p.productId, name as product_name, sum(oi.itemAmount) as
product_revenue, (sum(oi.itemAmount) / (select sum(itemAmount) from
orderitem)) * 100 as revenue_percentage
from product p
join orderitem oi on p.productId = oi.productId
group by p.productId, p.name
order by revenue_percentage DESC;

```

SQL File 1* x

Limit to 1000 rows

```

1 • select p.productId, p.name as product_name,
2    sum(oi.itemAmount) as product_revenue,
3    (SUM(oi.itemAmount) / (select sum(itemAmount) from orderitem)) * 100 as revenue_percentage
4    from product p
5    join orderitem oi on p.productId = oi.productId
6    group by p.productId, p.name
7    order by revenue_percentage DESC;

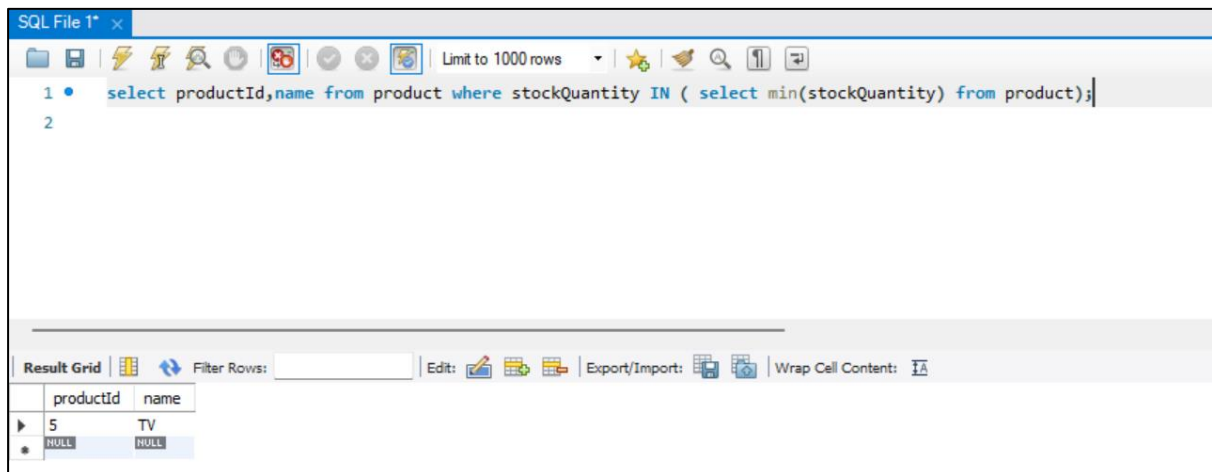
```

Result Grid

productId	product_name	product_revenue	revenue_percentage
2	Smartphone	3000.00	34.883721
1	Laptop	2400.00	27.906977
5	TV	1800.00	20.930233
4	Headphones	600.00	6.976744
3	Tablet	300.00	3.488372
10	Vacuum Cleaner	240.00	2.790698
9	Blender	210.00	2.441860
6	Coffee Maker	50.00	0.581395

18. Subquery to Find Products with Low Stock.

select productId,name from product where stockQuantity in(select min(stockQuantity) from product);



The screenshot shows a SQL IDE window titled "SQL File 1*" with a toolbar and a "Limit to 1000 rows" dropdown. The query editor contains the following SQL statement:

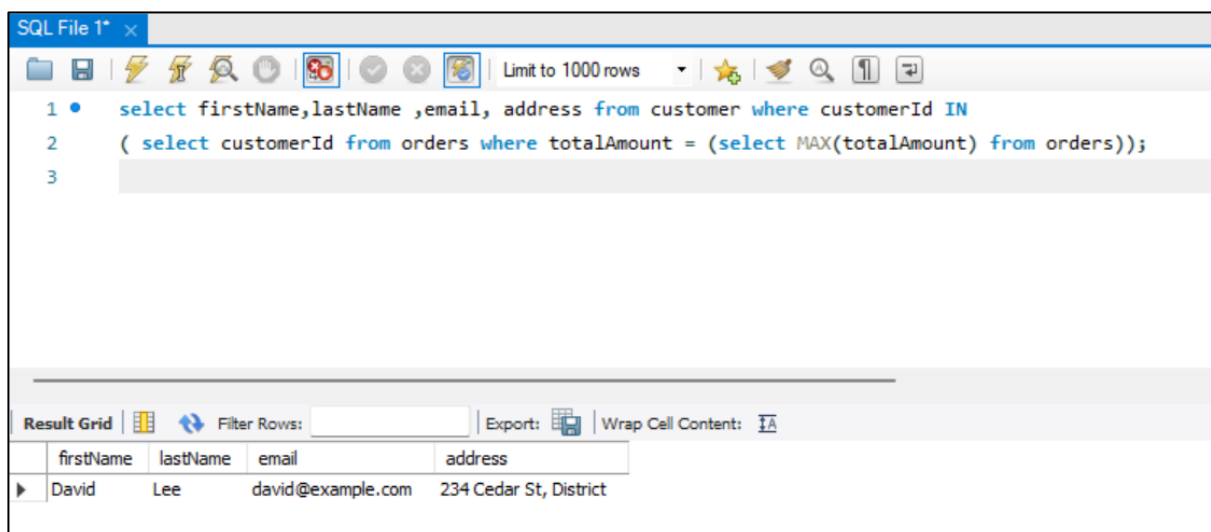
```
1 • select productId,name from product where stockQuantity IN ( select min(stockQuantity) from product);
2
```

Below the query editor is a "Result Grid" section with a "Filter Rows:" input field and buttons for "Edit:", "Export/Import:", and "Wrap Cell Content:". The result grid displays the following data:

productId	name
5	TV
NULL	NULL

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

select firstName,lastName,email,address from customer where customerId in
(select customerId from orders where totalAmount=(select max(totalAmount) from orders));



The screenshot shows a SQL IDE window titled "SQL File 1*" with a toolbar and a "Limit to 1000 rows" dropdown. The query editor contains the following SQL statement:

```
1 • select firstName,lastName ,email, address from customer where customerId IN
2 ( select customerId from orders where totalAmount = (select MAX(totalAmount) from orders));
3
```

Below the query editor is a "Result Grid" section with a "Filter Rows:" input field and buttons for "Export:" and "Wrap Cell Content:". The result grid displays the following data:

firstName	lastName	email	address
David	Lee	david@example.com	234 Cedar St, District