

1. `Select *, (Select customer_id from orders WHERE orders.id = order_details.order_id) AS customer_id from order_details;`

The screenshot shows a database management tool interface. The top tab is 'orders', and the 'order_details' tab is selected. The SQL editor contains the query: `1 • Select *, (Select customer_id from orders WHERE orders.id = order_details.order_id) AS customer_id from order_details;` The results are displayed in a 'Result Grid' with columns: id, order_id, product_id, quantity, and customer_id. The grid shows 28 rows of data. The interface includes a 'Limit to 1000 rows' dropdown, a 'Filter Rows' search bar, and an 'Export' button.

	id	order_id	product_id	quantity	customer_id
1	10248	11	12	90	
2	10248	42	10	90	
3	10248	72	5	90	
4	10249	14	9	81	
5	10249	51	40	81	
6	10250	41	10	34	
7	10250	51	35	34	
8	10250	65	15	34	
9	10251	22	6	84	
10	10251	57	15	84	
11	10251	65	20	84	
12	10252	20	40	76	
13	10252	33	25	76	
14	10252	60	40	76	
15	10253	31	20	34	
16	10253	39	42	34	
17	10253	49	40	34	
18	10254	24	15	14	
19	10254	55	21	14	
20	10254	74	21	14	
21	10255	2	20	68	
22	10255	16	35	68	
23	10255	36	25	68	
24	10255	69	30	68	
25	10256	53	15	88	
26	10256	77	12	88	
Result 28					

2.

`SELECT *`

`FROM order_details`

`WHERE order_details.order_id IN (`

`SELECT orders.id`

`FROM orders`

`WHERE orders.shipper_id = 3`

`);`

The screenshot shows a database IDE with a SQL query editor and a result grid. The query is as follows:

```

1 • SELECT *
2 FROM order_details
3 WHERE order_details.order_id IN (
4     SELECT orders.id
5     FROM orders
6     WHERE orders.shipper_id = 3
7 );

```

The result grid displays the following data:

	id	order_id	product_id	quantity
1	1	10248	11	12
2	2	10248	42	10
3	3	10248	72	5
4	4	10249	14	9
5	5	10249	51	40
6	6	10250	41	10
7	7	10250	51	35
8	8	10250	65	15
9	9	10251	22	6
10	10	10251	57	15
11	11	10251	65	20
12	12	10252	20	40
13	13	10252	33	25
14	14	10252	60	40
15	15	10253	31	20
16	16	10253	39	42
17	17	10253	49	40
18	18	10254	24	15
19	19	10254	55	21
20	20	10254	74	21
21	21	10255	2	20
22	22	10255	16	35
23	23	10255	36	25
24	24	10255	59	30
25	25	10256	53	15
26	26	10256	77	12

The result grid also includes a search bar, an export button, and a status bar at the bottom indicating 'order_details 29'.

3.

```

SELECT order_id, avg(quantity) as average_quantity
FROM (
select order_id, quantity
from order_details
WHERE quantity > 10
) as filter

```

Group by order_id;

The screenshot shows a database IDE with a SQL editor and a results grid. The SQL editor contains the following query:

```
1 • SELECT order_id, avg(quantity) as average_quantity
2 FROM (
3   select order_id, quantity
4   from order_details
5   WHERE quantity > 10
6 ) as filter
7 Group by order_id;
```

The results grid displays the output of the query, showing the order_id and the average quantity for each order. The results are as follows:

order_id	average_quantity
10250	25.0000
10251	17.5000
10252	35.0000
10253	34.0000
10254	19.0000
10255	27.5000
10256	13.5000
10257	20.0000
10258	57.5000
10260	25.5000
10261	20.0000
10262	13.5000
10263	46.0000
10264	30.0000
10265	25.0000
10266	12.0000
10267	45.0000
10269	40.0000
10270	27.5000
10271	24.0000
10272	32.0000
10273	30.4000
10274	20.0000
10275	12.0000
10276	15.0000
10277	16.0000
10278	18.6667
10279	15.0000

4.

```
With temp as (
  select order_id, quantity
  from order_details
  WHERE quantity > 10
)
SELECT order_id, avg(quantity) as average_quantity
FROM temp
Group by order_id;
```

orders	orders	order_details
Limit to 1000 rows		
<pre> 5 WHERE quantity > 10 6) as filter 7 Group by order_id; 8 9 10 With temp as (11 select order_id, quantity 12 from order_details 13 WHERE quantity > 10 14) 15 SELECT order_id, avg(quantity) as average_quantity 16 FROM temp 17 Group by order_id; 18 </pre>		
100%	20:13	
Result Grid	Filter Rows: Search	Export:
order_id	average_quantity	
10248	12.0000	
10249	40.0000	
10250	25.0000	
10251	17.5000	
10252	35.0000	
10253	34.0000	
10254	19.0000	
10255	27.5000	
10256	13.5000	
10257	20.0000	
10258	57.5000	
10260	25.5000	
10261	20.0000	
10262	13.5000	
10263	46.0000	
10264	30.0000	
10265	25.0000	
10266	12.0000	
10267	45.0000	
10269	40.0000	
10270	27.5000	
Result 45		

5.

DROP FUNCTION IF EXISTS divide_numbers;

DELIMITER //

CREATE FUNCTION divide_numbers(x FLOAT,y FLOAT) RETURNS FLOAT

no sql

BEGIN

DECLARE result FLOAT;

IF y = 0 THEN

RETURN NULL;

ELSE

SET result = x / y;

RETURN result;

END IF;

END//

SELECT order_id, quantity, divide_numbers(quantity, 2) AS divided_quantity

FROM order_details

DELIMITER;

The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search. The main window displays SQL code for creating and dropping a function, and a query to select order details with the function applied. The left sidebar shows a tree view of database objects, including tables, views, and stored procedures. The bottom section shows the 'Result Grid' with columns for order_id, quantity, and divided_quantity, and the 'Action Output' section showing execution logs.

```
1
2 • DROP FUNCTION IF EXISTS divide_numbers;
3 DELIMITER //
4 • CREATE FUNCTION divide_numbers(x FLOAT, y FLOAT) RETURNS FLOAT
5 no sql
6 BEGIN
7     DECLARE result FLOAT;
8     IF y = 0 THEN
9         RETURN NULL;
10    ELSE
11        SET result = x / y;
12        RETURN result;
13    END IF;
14 END//
15
```

order_id	quantity	divided_quant...
10248	12	6
10248	10	5
10248	5	2.5
10249	9	4.5
10249	40	20
10250	10	5
10250	35	17.5
10250	15	7.5
.....		

Time	Response
14:40:...	C 0 row(s) affected
14:40:...	S 518 row(s) returned
14:44:...	D 0 row(s) affected
14:44:...	C 0 row(s) affected
14:44:...	S 518 row(s) returned

