WELCOME TO

SQL PIZZA SHOP

WHERE EVERY QUERY DELIVERS A SLICE OF INSIGHTS!

ORDER YOUR DATA NOW

PROJECT OVERVIEW

This project showcases my SQL skills through data analysis on a pizza sales dataset. It includes querying, data aggregation, and insights on sales performance, customer preferences, and revenue distribution.

OBJECTIVES

- To analyze pizza sales data using SQL.
- To extract meaningful insights such as top-selling pizzas, revenue trends, and order patterns.
- To demonstrate proficiency in SQL queries, joins, subqueries, and window functions.

DATASET USED

NAME:

- PIZZA SALES DATASET

DESCRIPTION:

- THIS DATASET CONTAINS INFORMATION ABOUT PIZZA ORDERS, INCLUDING DETAILS ON ORDER DATE, PIZZA TYPE, SIZE, QUANTITY, AND PRICE. IT HELPS ANALYZE SALES TRENDS, CUSTOMER PREFERENCES, AND REVENUE DISTRIBUTION.

KEY TABLES:

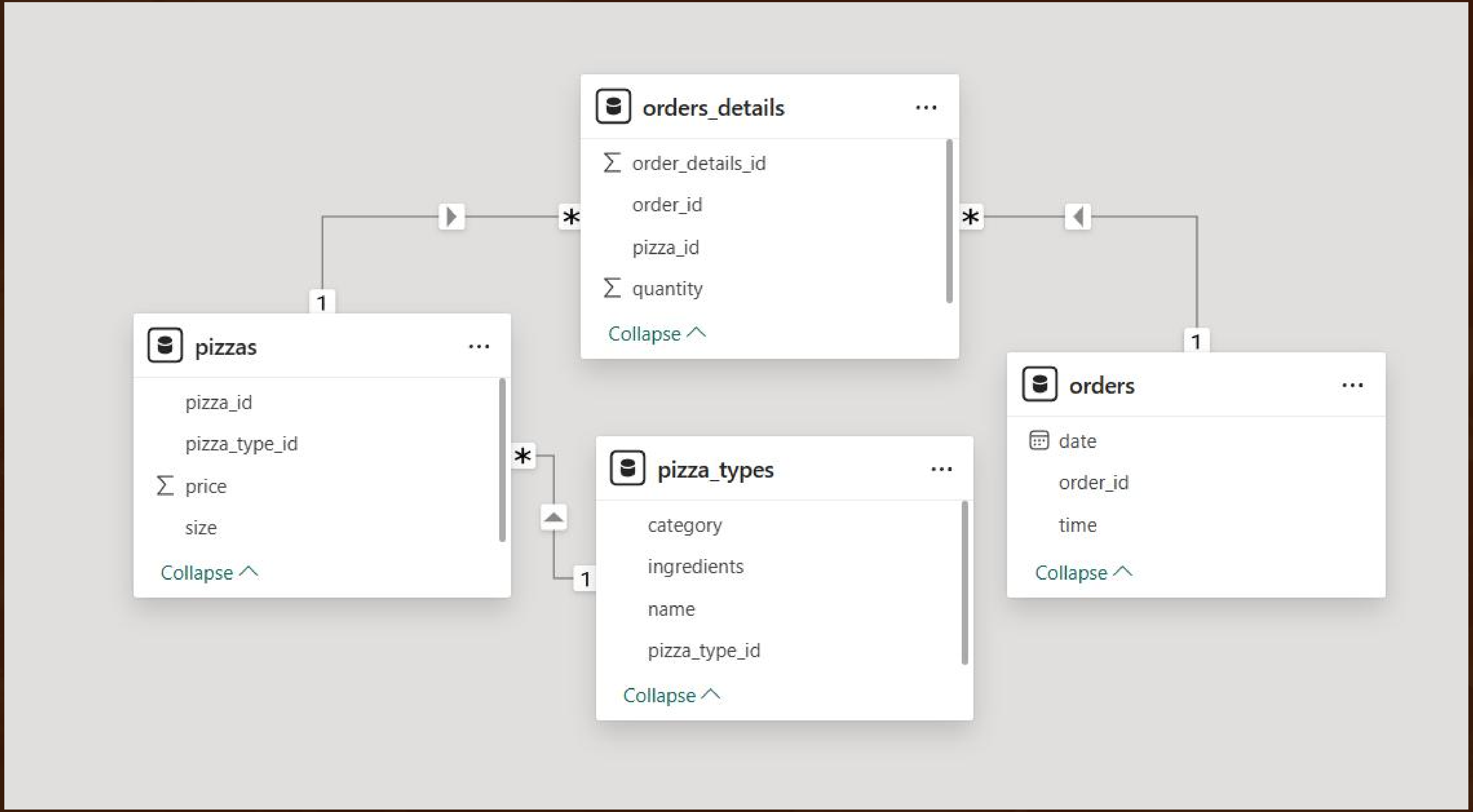
- ORDERS: CONTAINS ORDER IDS, ORDER DATES, AND TIMESTAMPS.
- ORDER DETAILS: LINKS EACH ORDER TO PIZZAS WITH QUANTITY AND PRICE INFORMATION.
- PIZZAS: PROVIDES DETAILS ABOUT EACH PIZZA, INCLUDING SIZE , PRICE AND TYPE
- PIZZA TYPES: LISTS DIFFERENT CATEGORIES OF PIZZAS, SUCH AS CLASSIC, VEGGIE, OR CHICKEN.

TOOLS USED

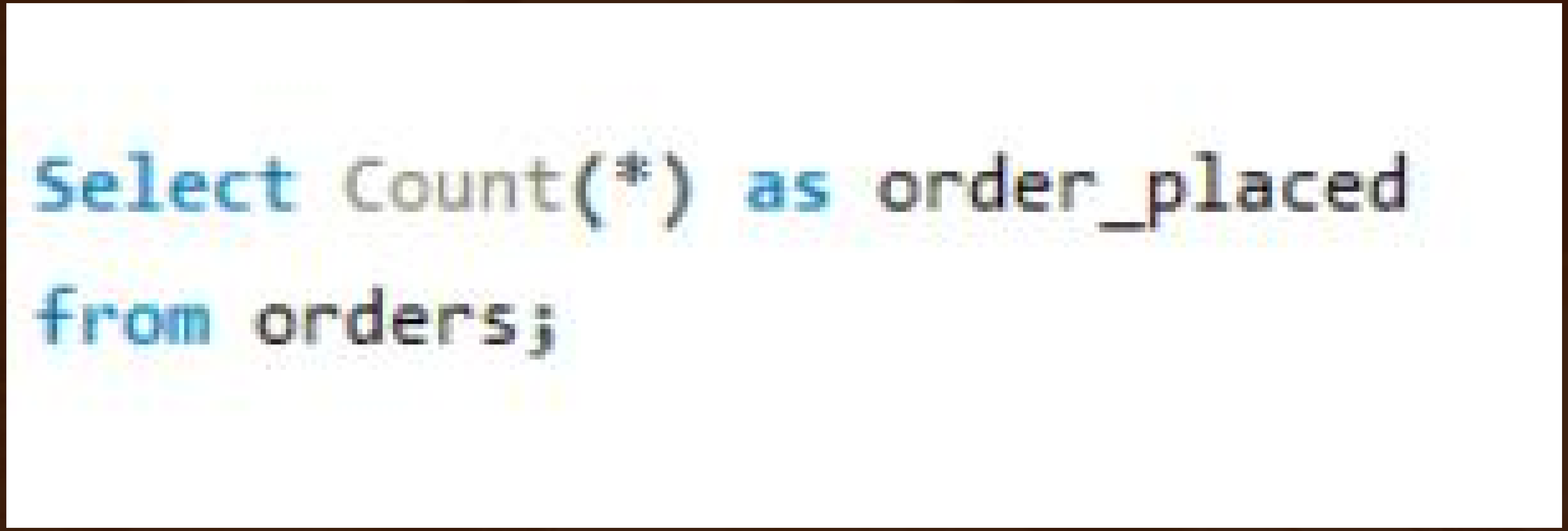
SQL DATABASE:

- MYSQL (USED FOR DATA EXTRACTION, TRANSFORMATION, AND QUERYING).

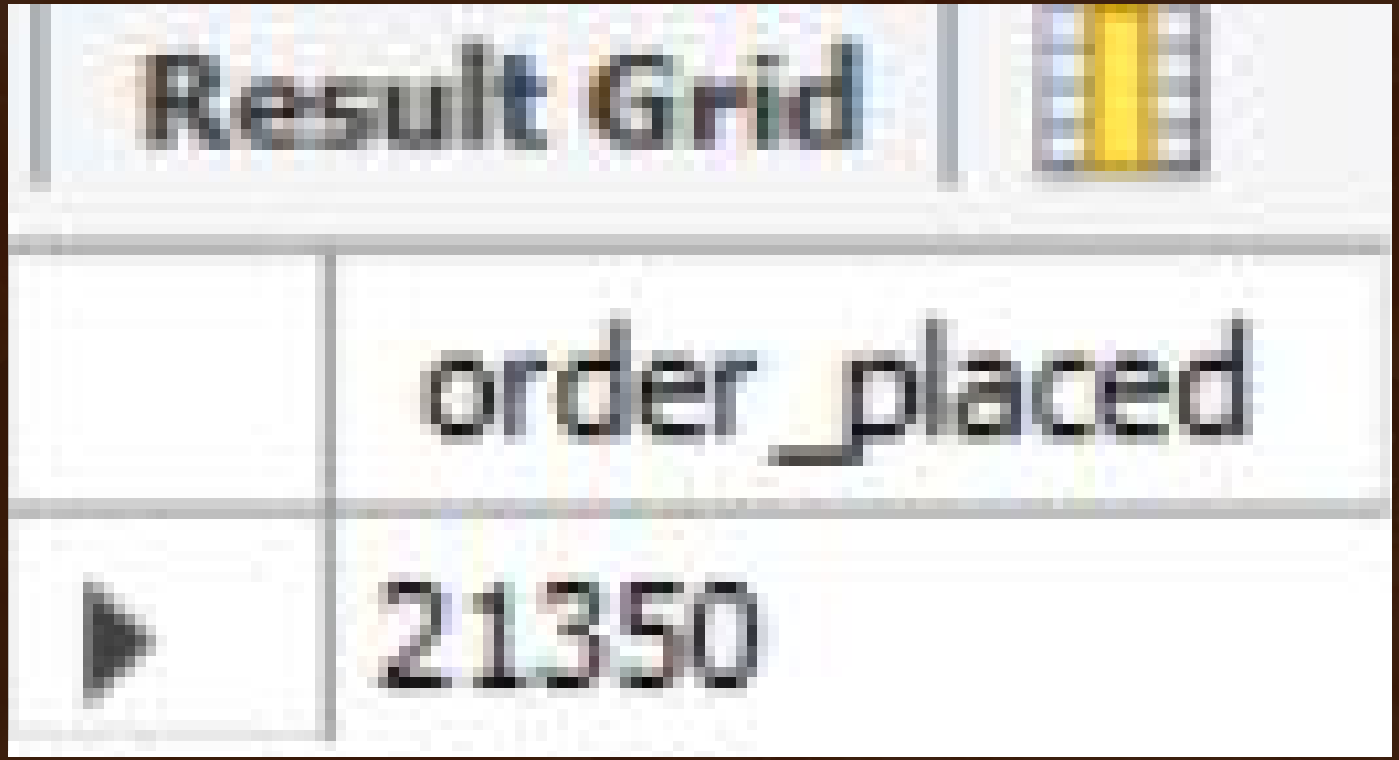
SCHEMA


orders_details
order_details_id
order_id
pizza_id
quantity
Collapse
pizzas
pizza_id
pizza_type_id
price
size
Collapse
pizza_types
category
ingredients
name
pizza_type_id
Collapse
orders
date
order_id
time
Collapse
1
*
1
*
*
1

1.RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

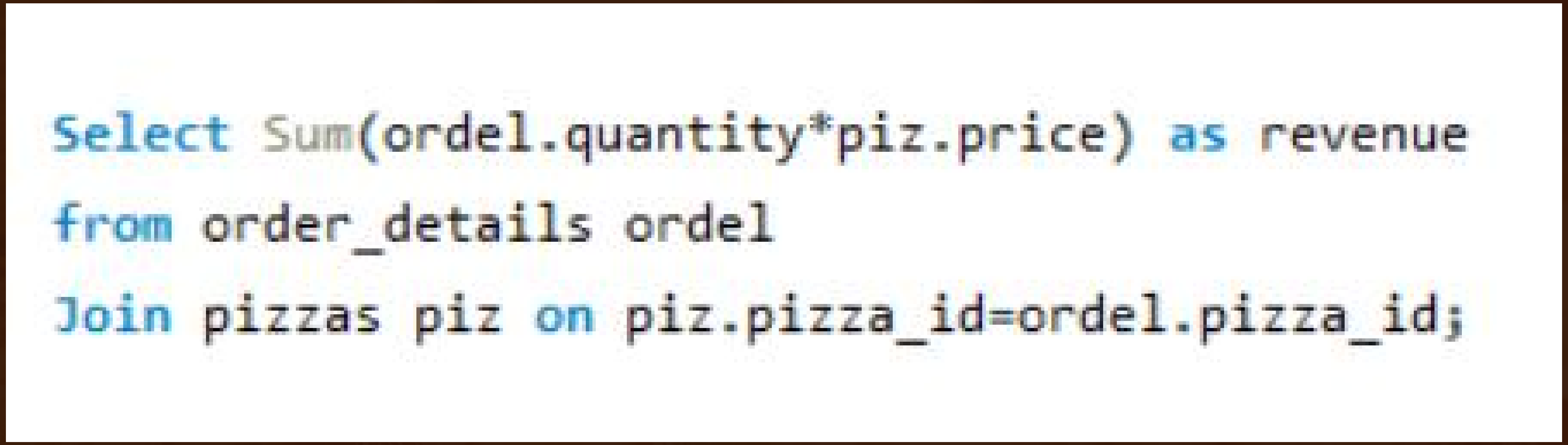


```sql
Select Count(*) as order_placed
from orders;
```

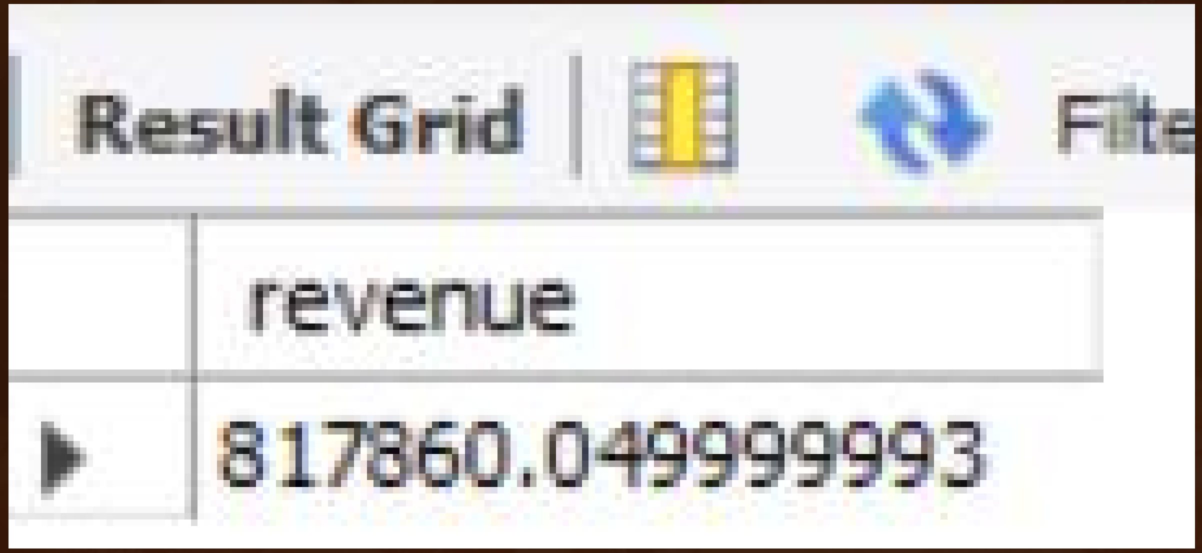


```
order_placed
21350
```

2. CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

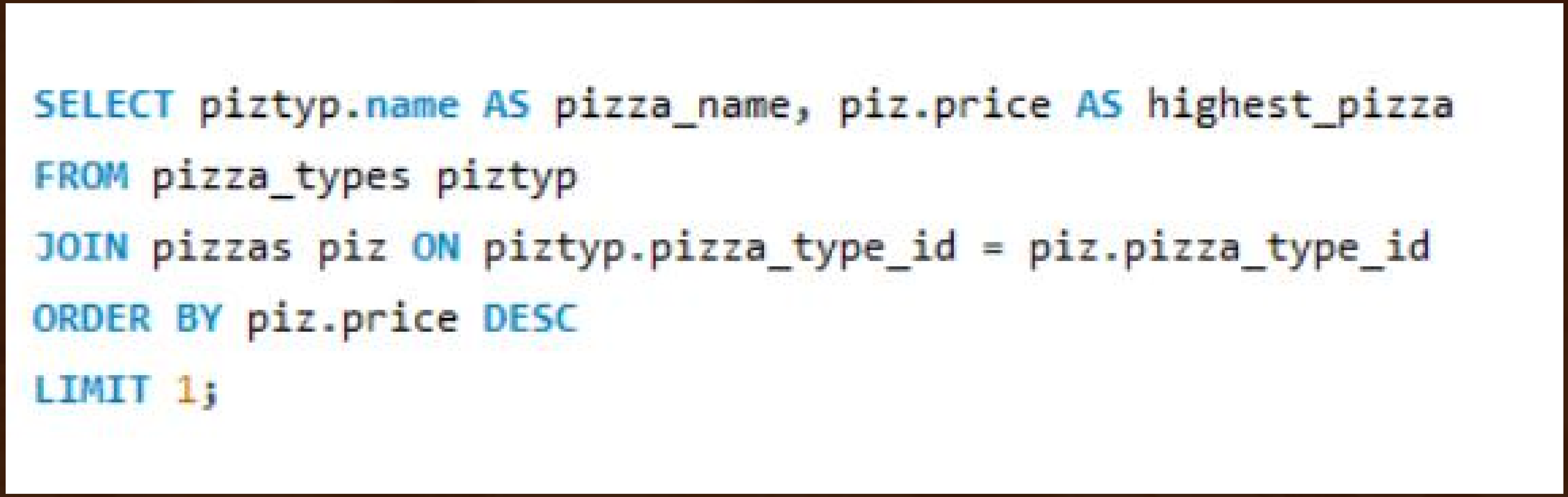


```sql
Select Sum(ordel.quantity*piz.price) as revenue
from order_details ordel
Join pizzas piz on piz.pizza_id=ordel.pizza_id;
```

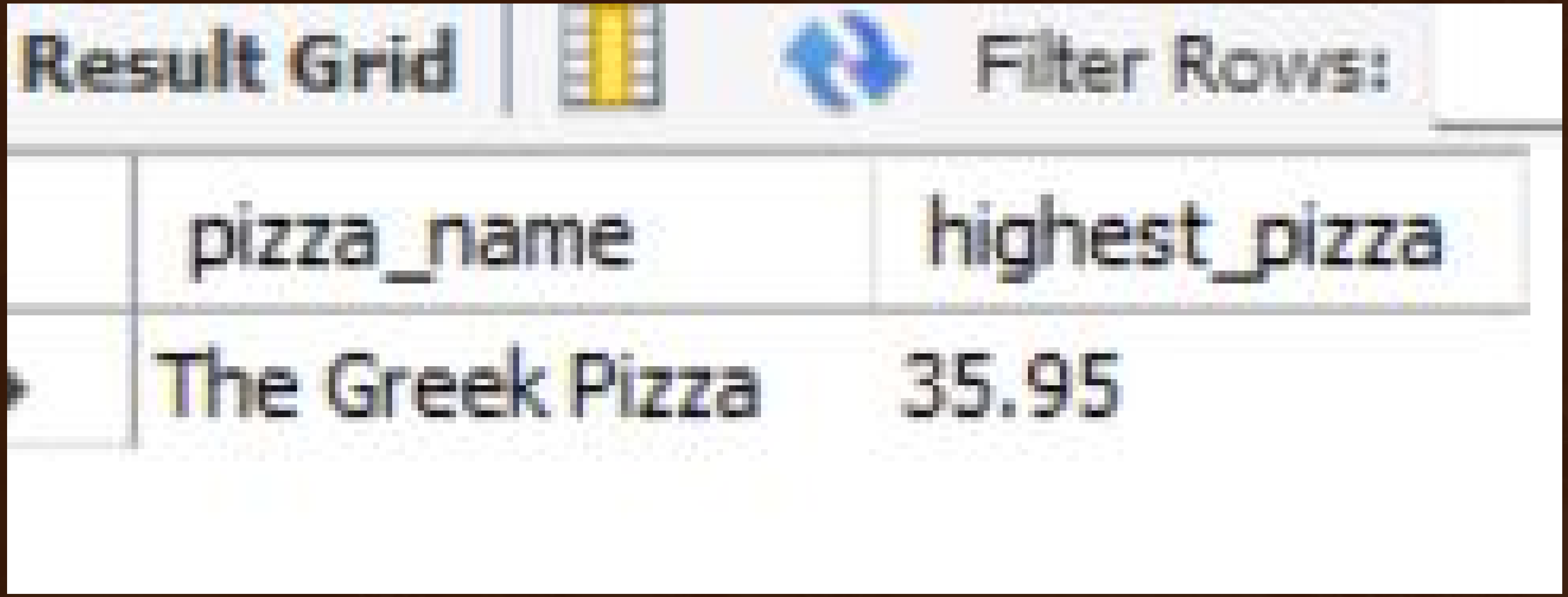


```
revenue
817860.049999993
```

3. IDENTIFY THE HIGHEST-PRICED PIZZA.

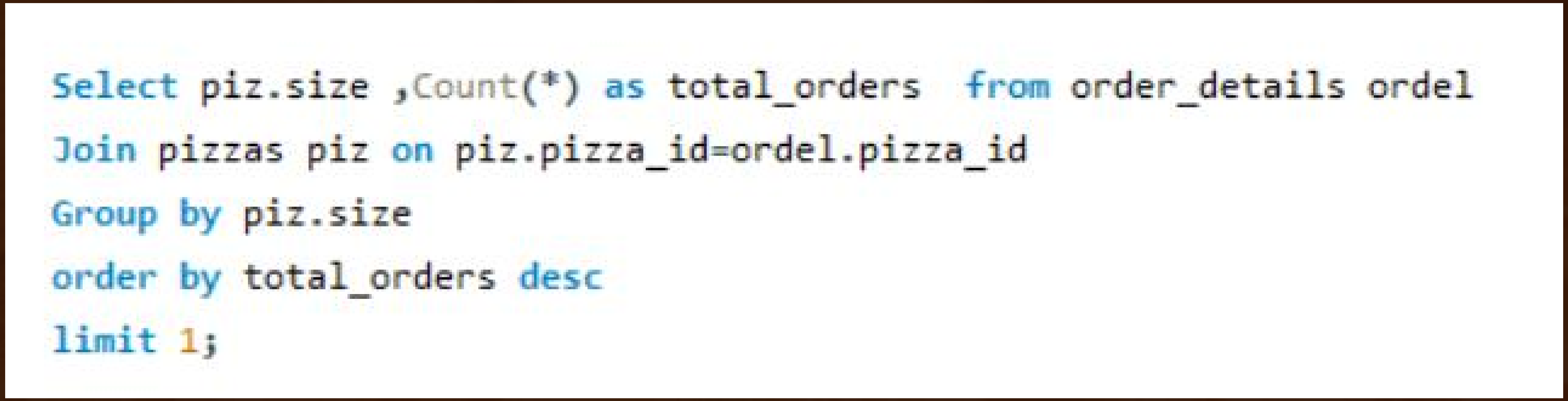


```sql
SELECT piztyp.name AS pizza_name, piz.price AS highest_pizza
FROM pizza_types piztyp
JOIN pizzas piz ON piztyp.pizza_type_id = piz.pizza_type_id
ORDER BY piz.price DESC
LIMIT 1;
```

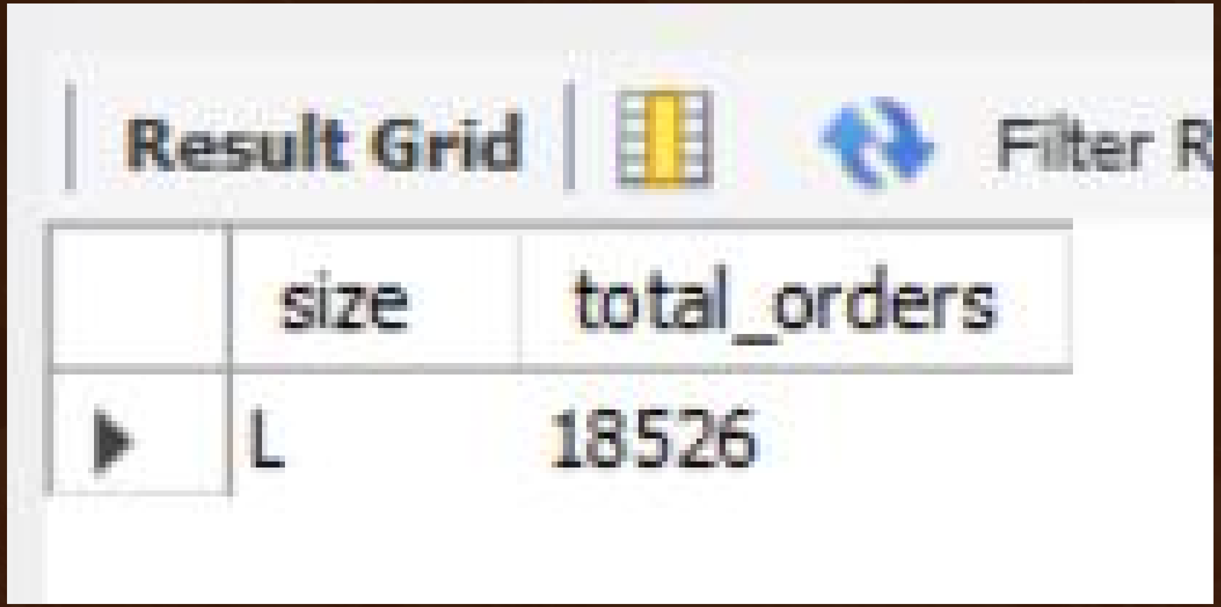


pizza_name	highest_pizza
The Greek Pizza	35.95

4. IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

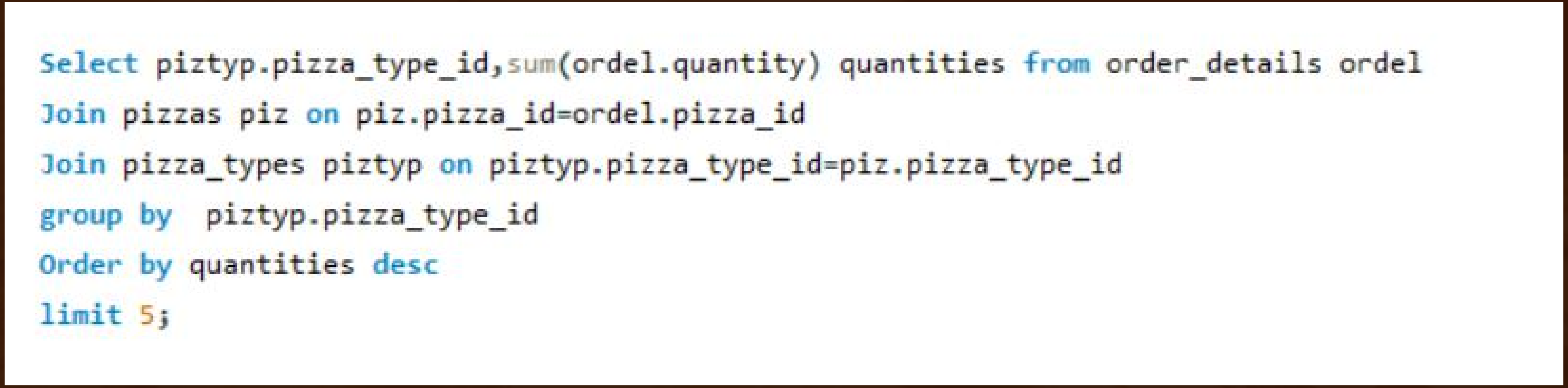


```
Select piz.size ,Count(*) as total_orders  from order_details ordel
Join pizzas piz on piz.pizza_id=ordel.pizza_id
Group by piz.size
order by total_orders desc
limit 1;
```

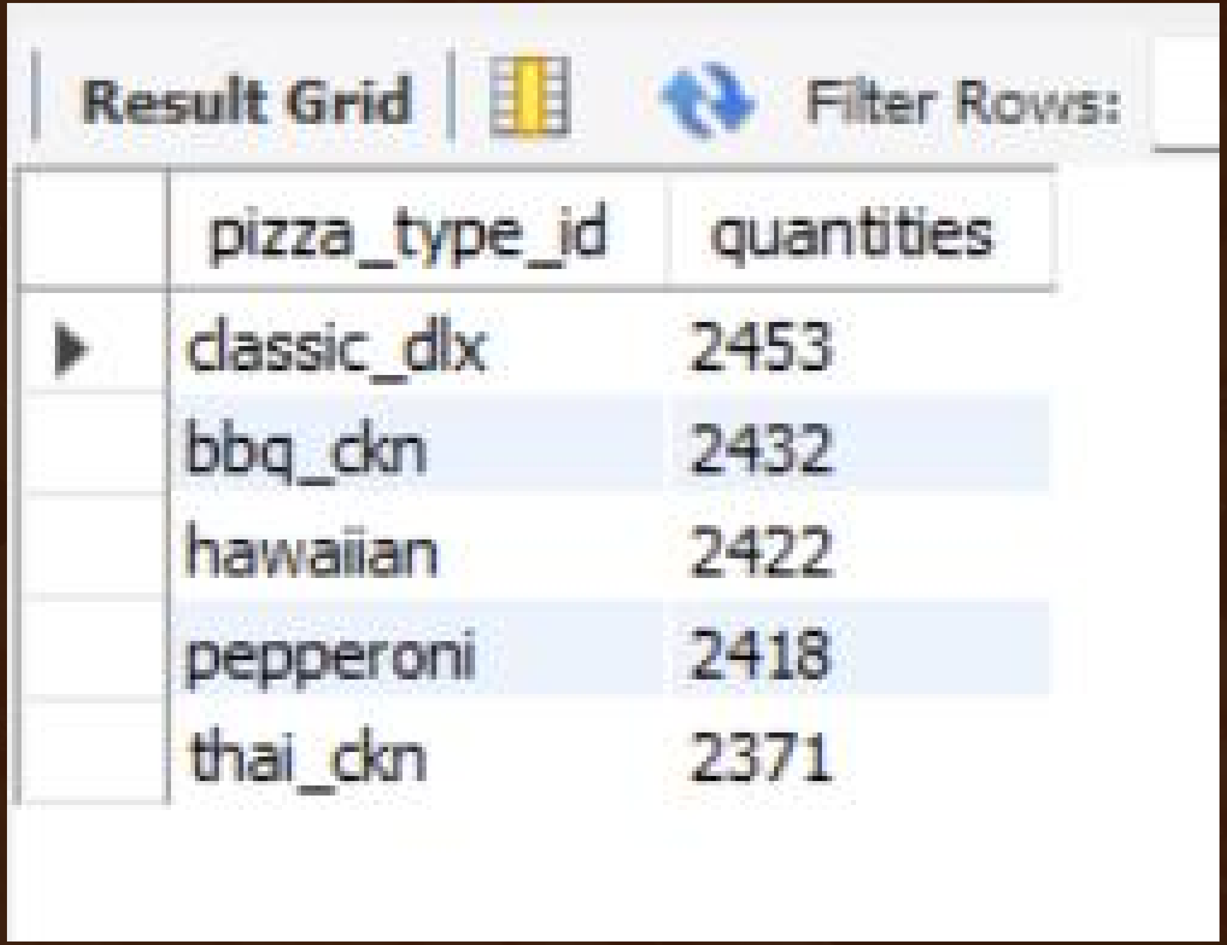


```
  size   total_orders
▶ L      18526
```

5. LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.

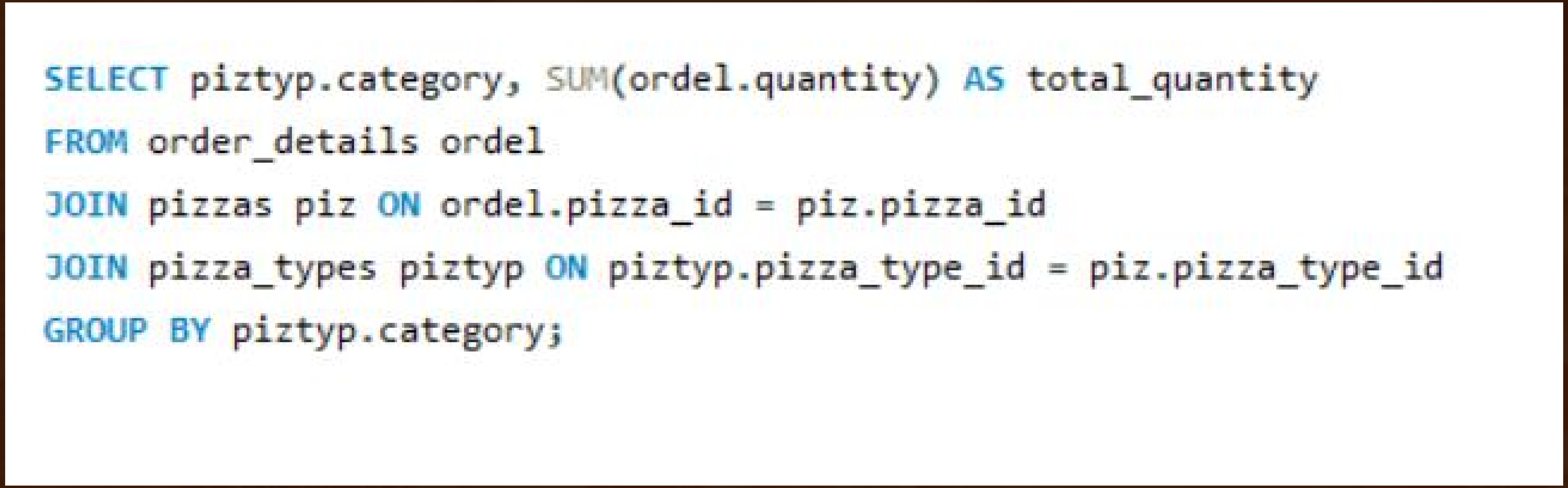


```
Select piztyp.pizza_type_id,sum(ordel.quantity) quantities from order_details ordel
Join pizzas piz on piz.pizza_id=ordel.pizza_id
Join pizza_types piztyp on piztyp.pizza_type_id=piz.pizza_type_id
group by  piztyp.pizza_type_id
Order by quantities desc
limit 5;
```

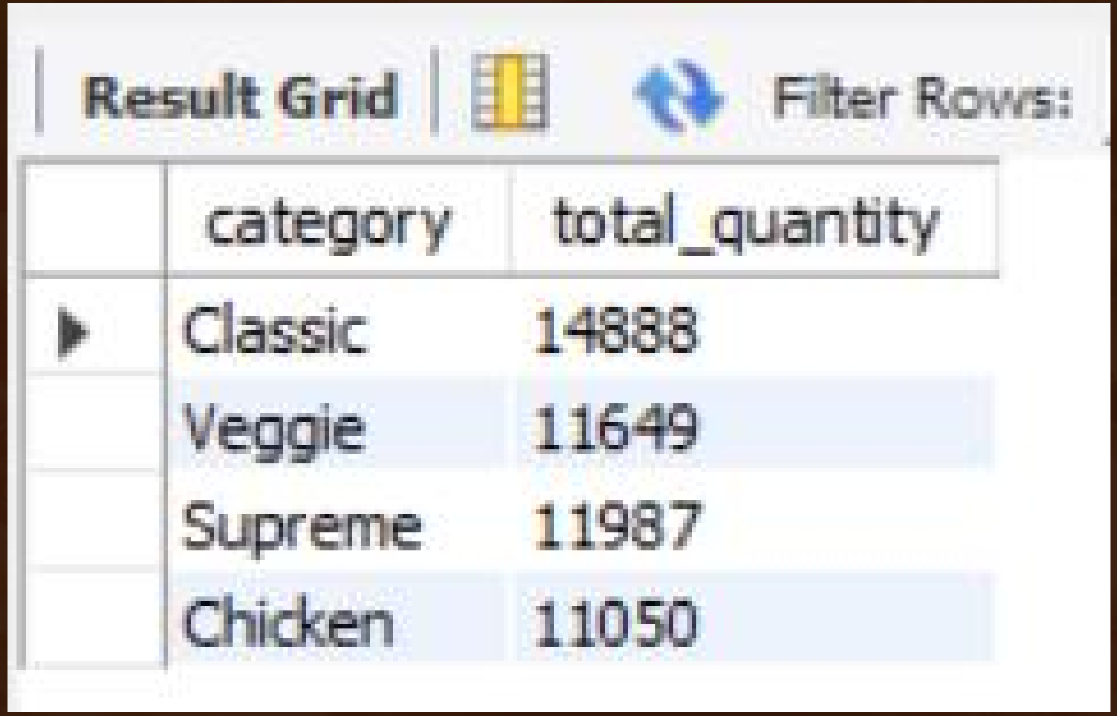


pizza_type_id	quantities
classic_dlx	2453
bbq_ckn	2432
hawaiian	2422
pepperoni	2418
thai_ckn	2371

6. JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

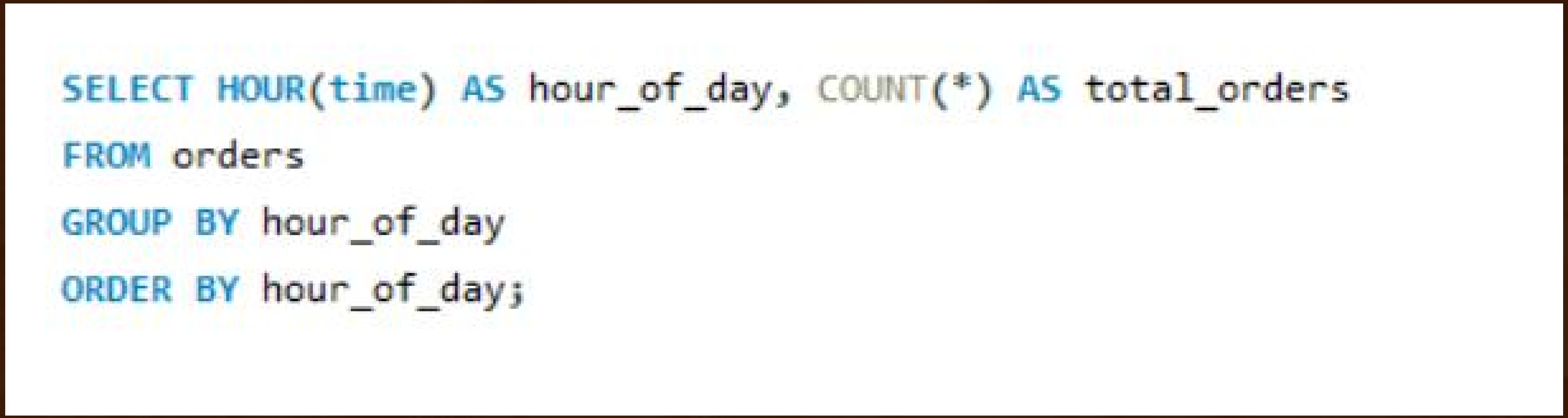


```sql
SELECT piztyp.category, SUM(ordel.quantity) AS total_quantity
FROM order_details ordel
JOIN pizzas piz ON ordel.pizza_id = piz.pizza_id
JOIN pizza_types piztyp ON piztyp.pizza_type_id = piz.pizza_type_id
GROUP BY piztyp.category;
```

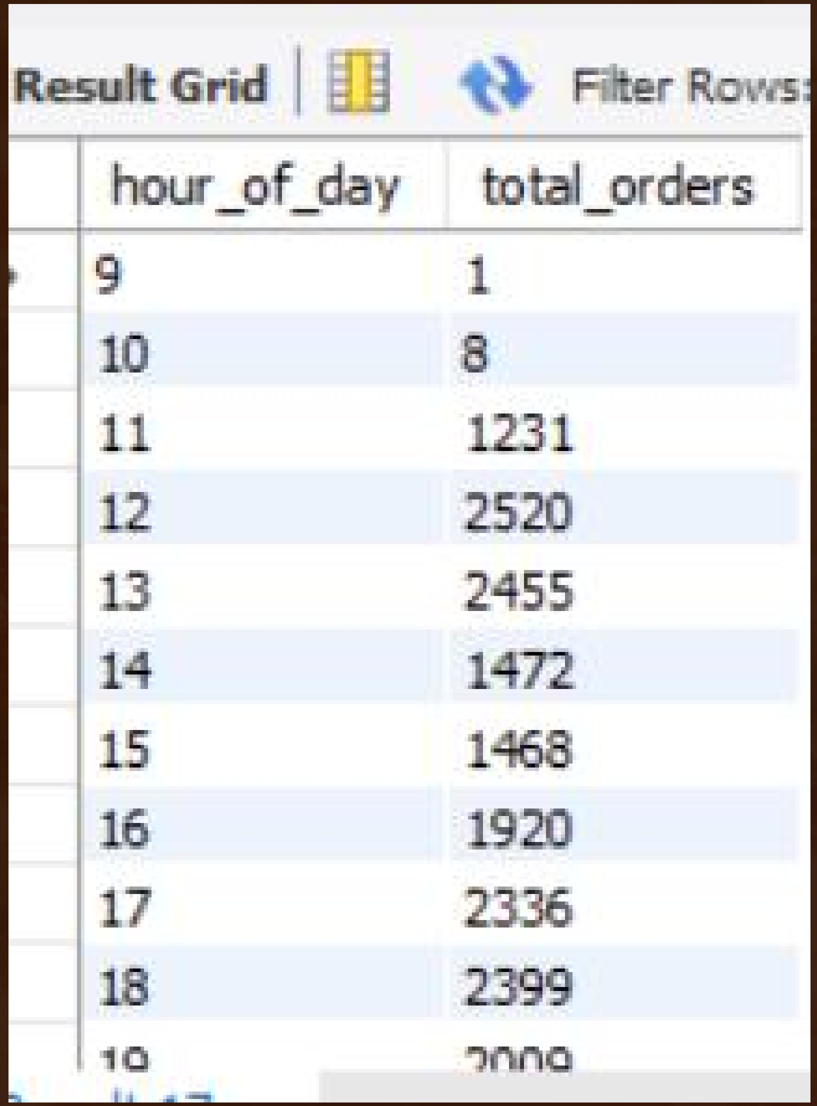


category	total_quantity
Classic	14888
Veggie	11649
Supreme	11987
Chicken	11050

7. DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

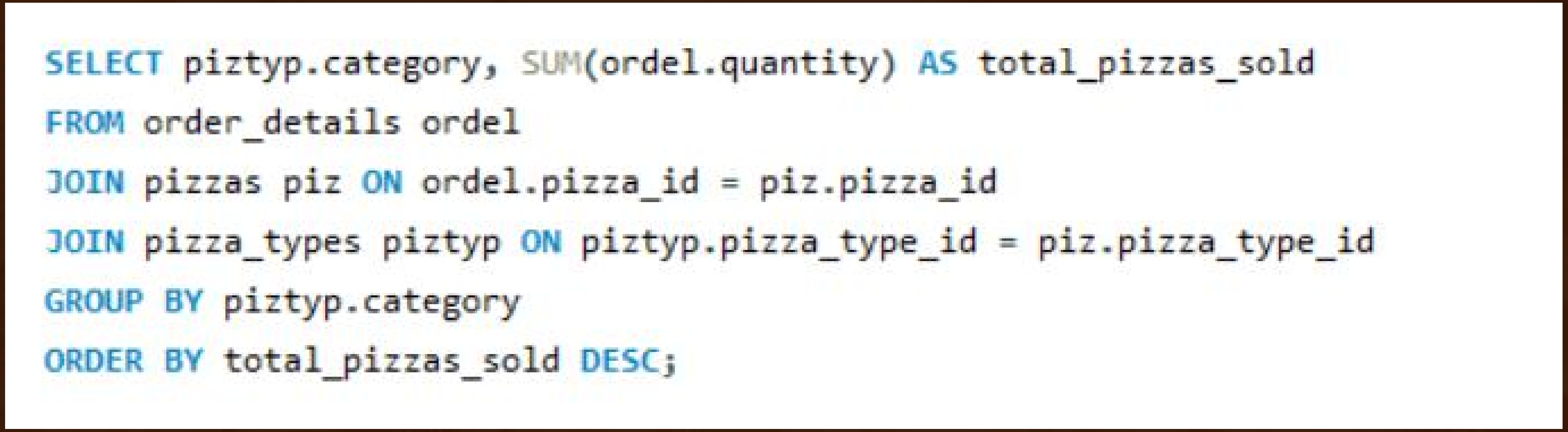


```sql
SELECT HOUR(time) AS hour_of_day, COUNT(*) AS total_orders
FROM orders
GROUP BY hour_of_day
ORDER BY hour_of_day;
```

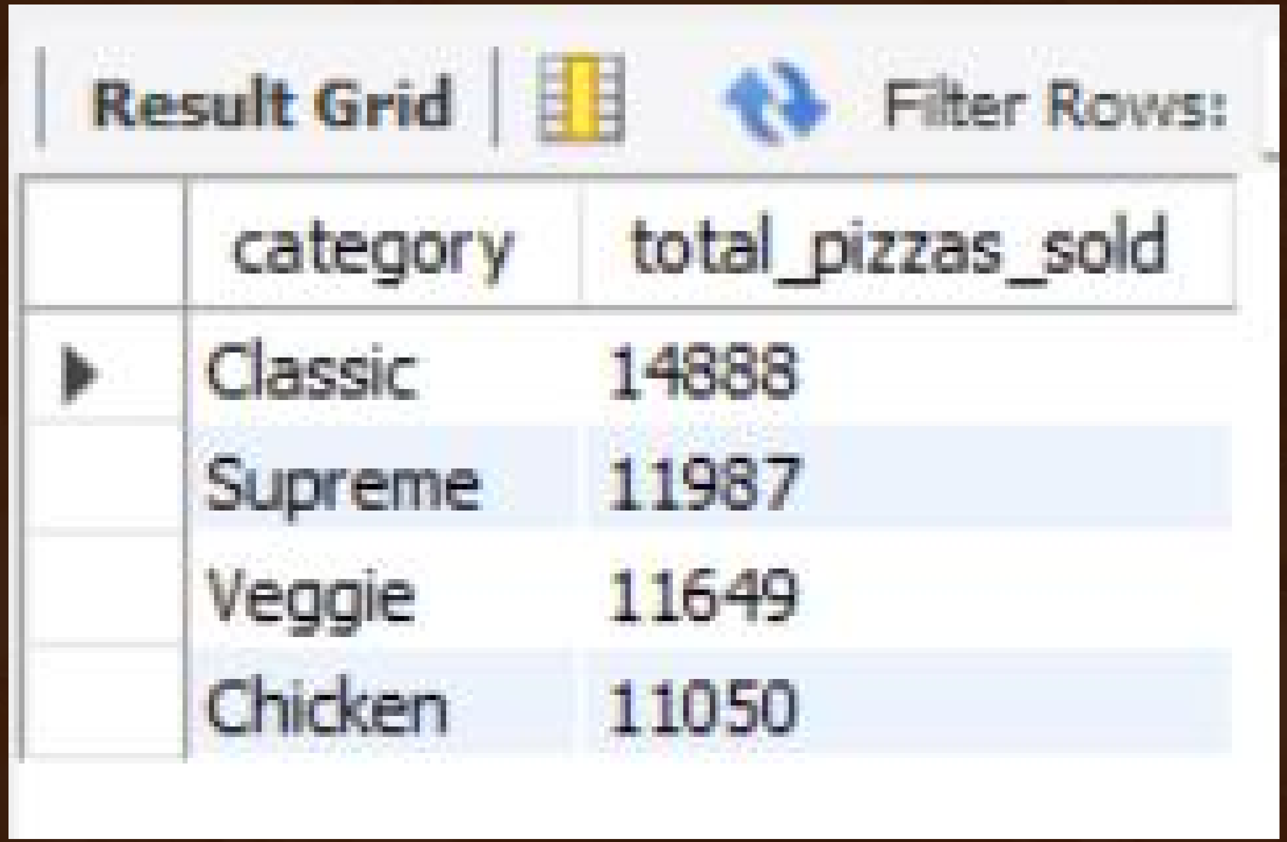


hour_of_day	total_orders
9	1
10	8
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009

8. JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

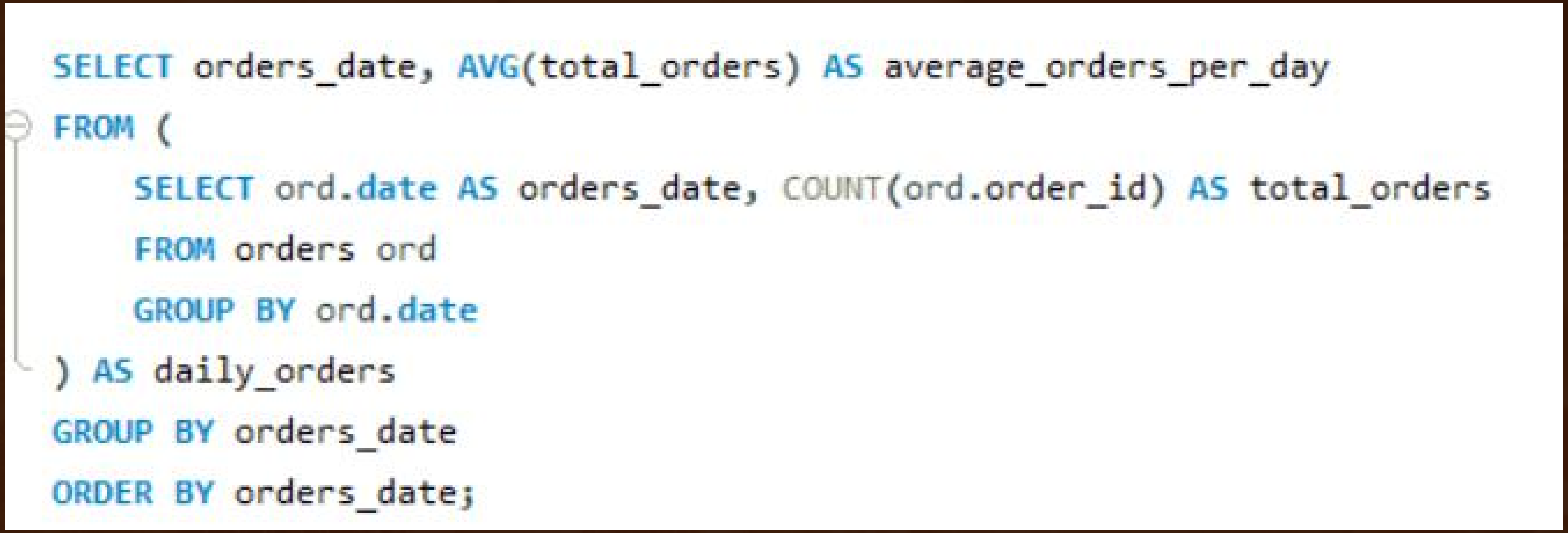


```
SELECT piztyp.category, SUM(ordel.quantity) AS total_pizzas_sold
FROM order_details ordel
JOIN pizzas piz ON ordel.pizza_id = piz.pizza_id
JOIN pizza_types piztyp ON piztyp.pizza_type_id = piz.pizza_type_id
GROUP BY piztyp.category
ORDER BY total_pizzas_sold DESC;
```

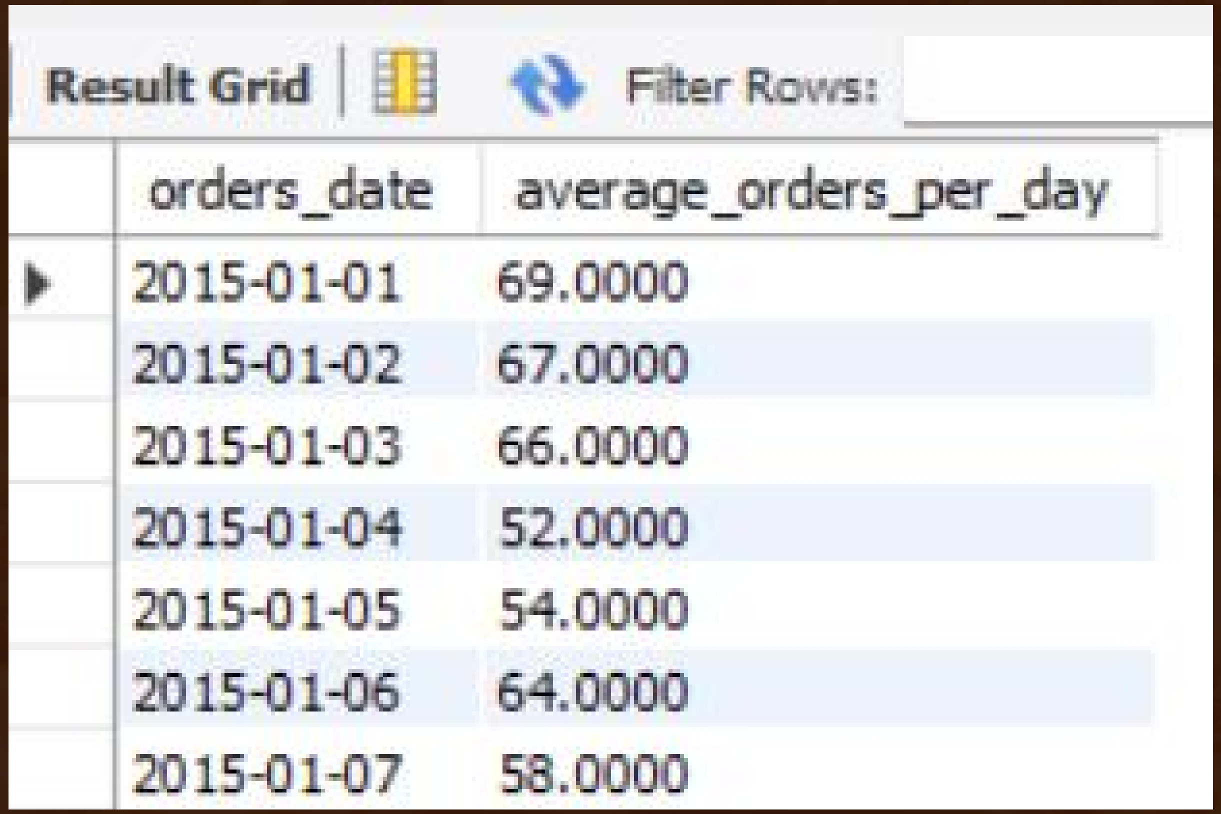


category	total_pizzas_sold
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

9. GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.



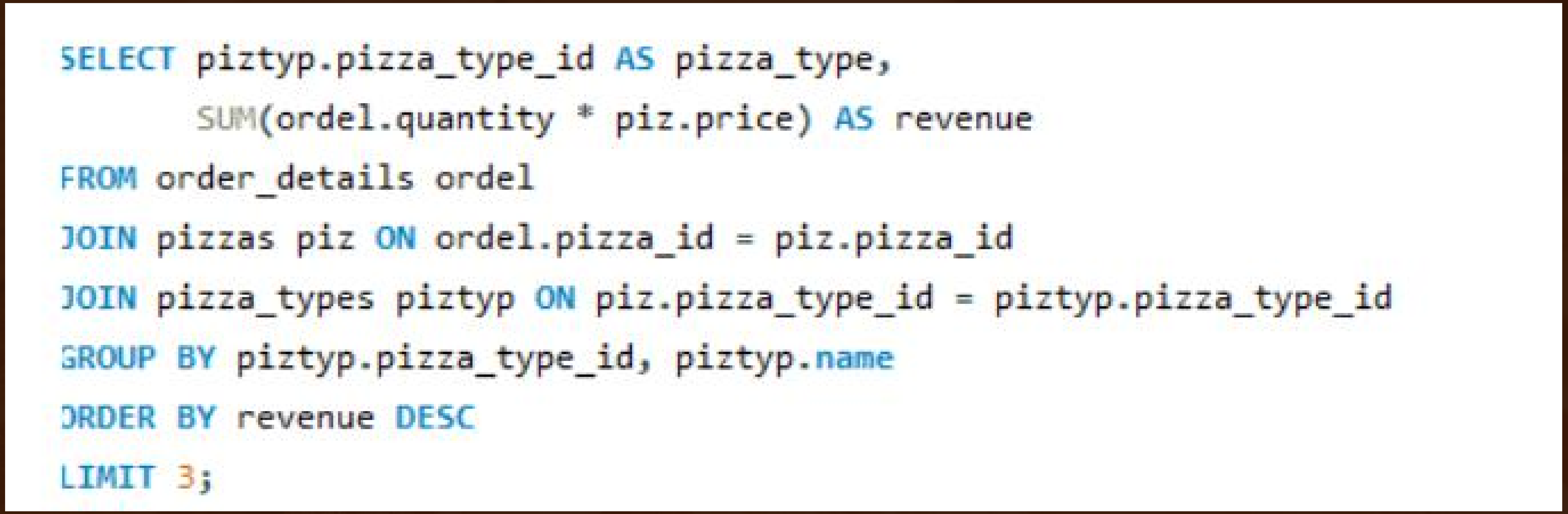
```
SELECT orders_date, AVG(total_orders) AS average_orders_per_day
FROM (
    SELECT ord.date AS orders_date, COUNT(ord.order_id) AS total_orders
    FROM orders ord
    GROUP BY ord.date
) AS daily_orders
GROUP BY orders_date
ORDER BY orders_date;
```



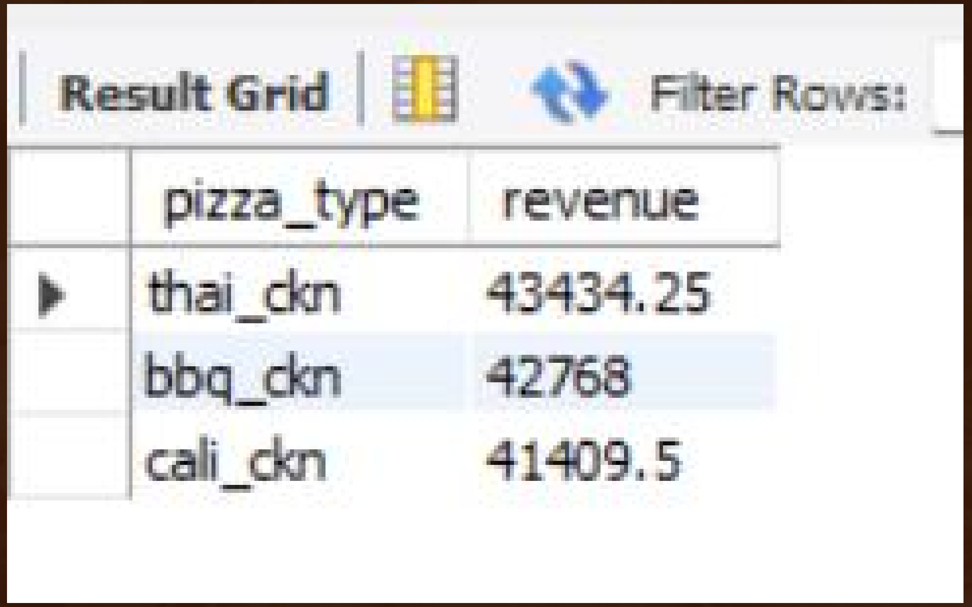
Result Grid | Filter Rows:

orders_date	average_orders_per_day
2015-01-01	69.0000
2015-01-02	67.0000
2015-01-03	66.0000
2015-01-04	52.0000
2015-01-05	54.0000
2015-01-06	64.0000
2015-01-07	58.0000

10. DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.



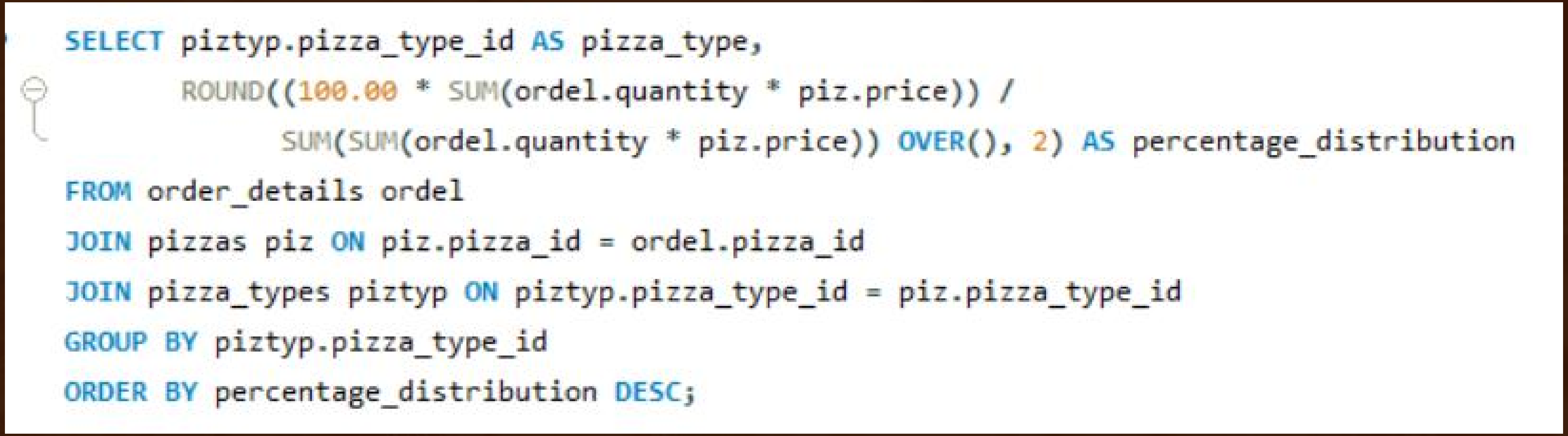
```sql
SELECT piztyp.pizza_type_id AS pizza_type,
       SUM(ordel.quantity * piz.price) AS revenue
FROM order_details ordel
JOIN pizzas piz ON ordel.pizza_id = piz.pizza_id
JOIN pizza_types piztyp ON piz.pizza_type_id = piztyp.pizza_type_id
GROUP BY piztyp.pizza_type_id, piztyp.name
ORDER BY revenue DESC
LIMIT 3;
```



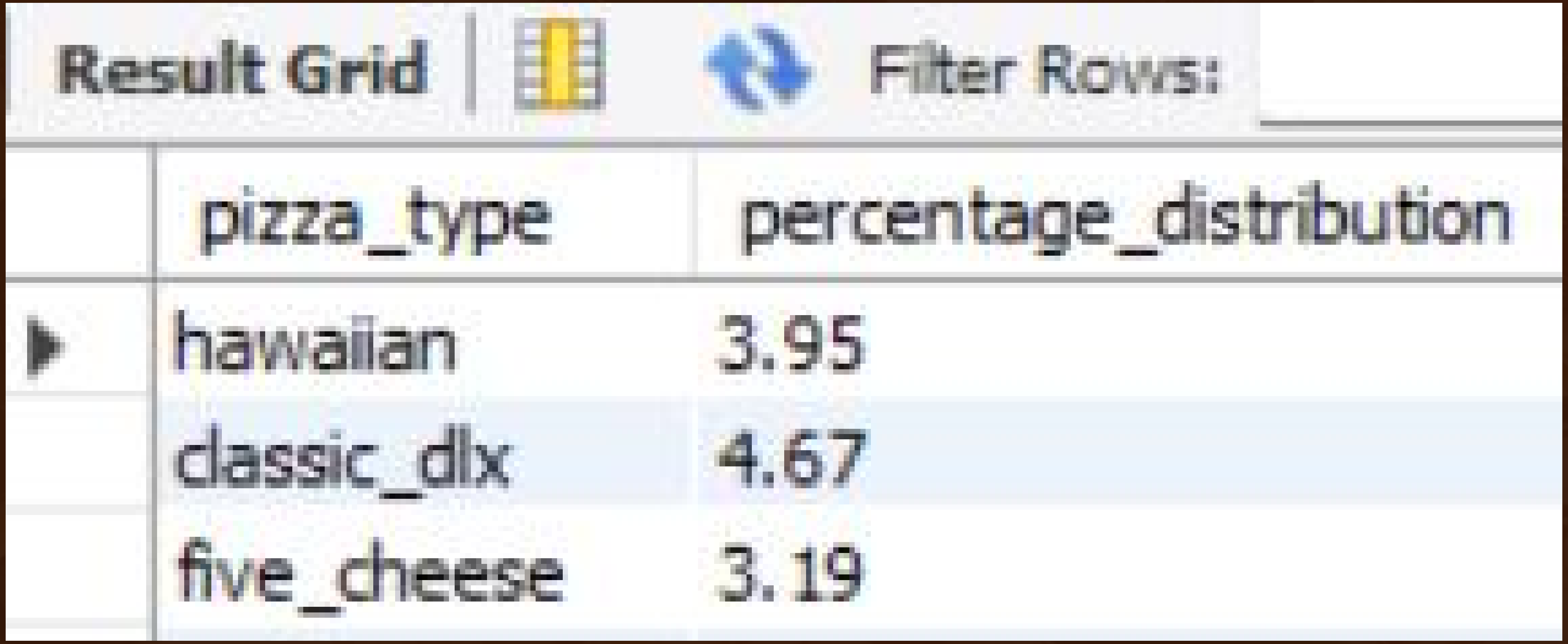
Result Grid | Filter Rows:

pizza_type	revenue
thai_ckn	43434.25
bbq_ckn	42768
cali_ckn	41409.5

11. CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.



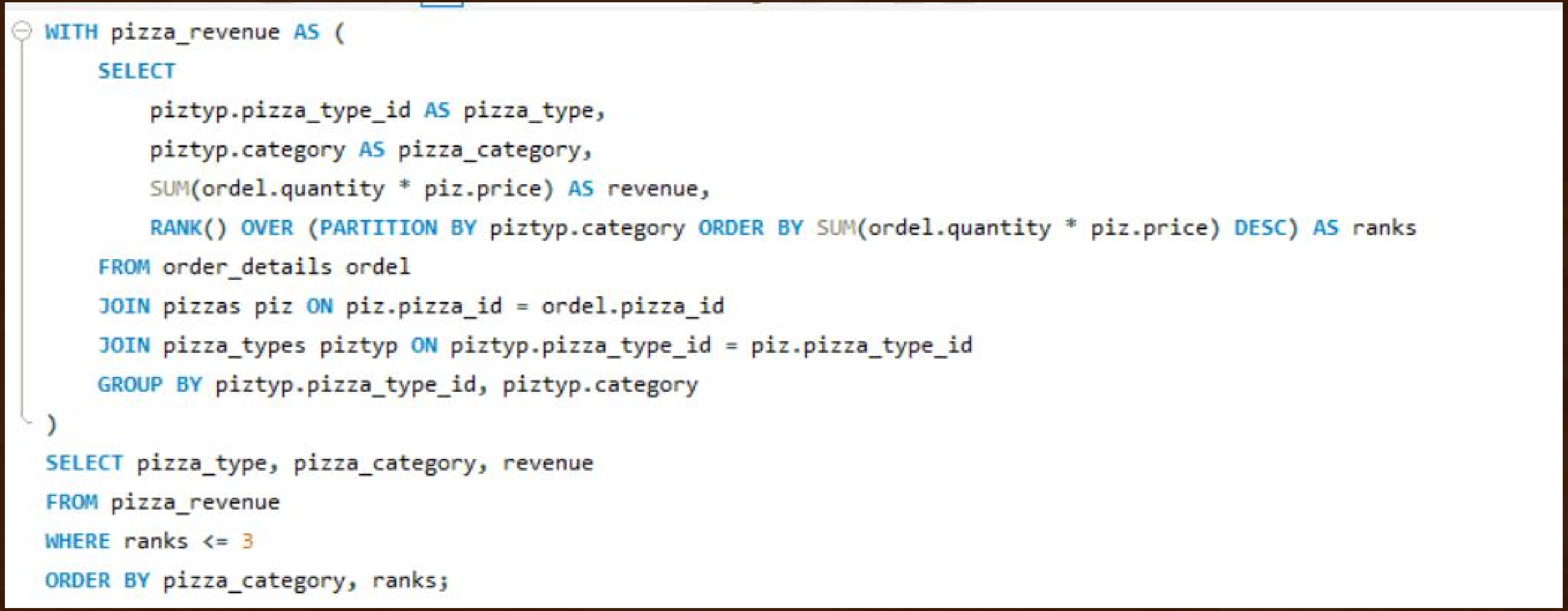
```
SELECT piztyp.pizza_type_id AS pizza_type,
       ROUND((100.00 * SUM(ordel.quantity * piz.price)) /
             SUM(SUM(ordel.quantity * piz.price)) OVER(), 2) AS percentage_distribution
FROM order_details ordel
JOIN pizzas piz ON piz.pizza_id = ordel.pizza_id
JOIN pizza_types piztyp ON piztyp.pizza_type_id = piz.pizza_type_id
GROUP BY piztyp.pizza_type_id
ORDER BY percentage_distribution DESC;
```



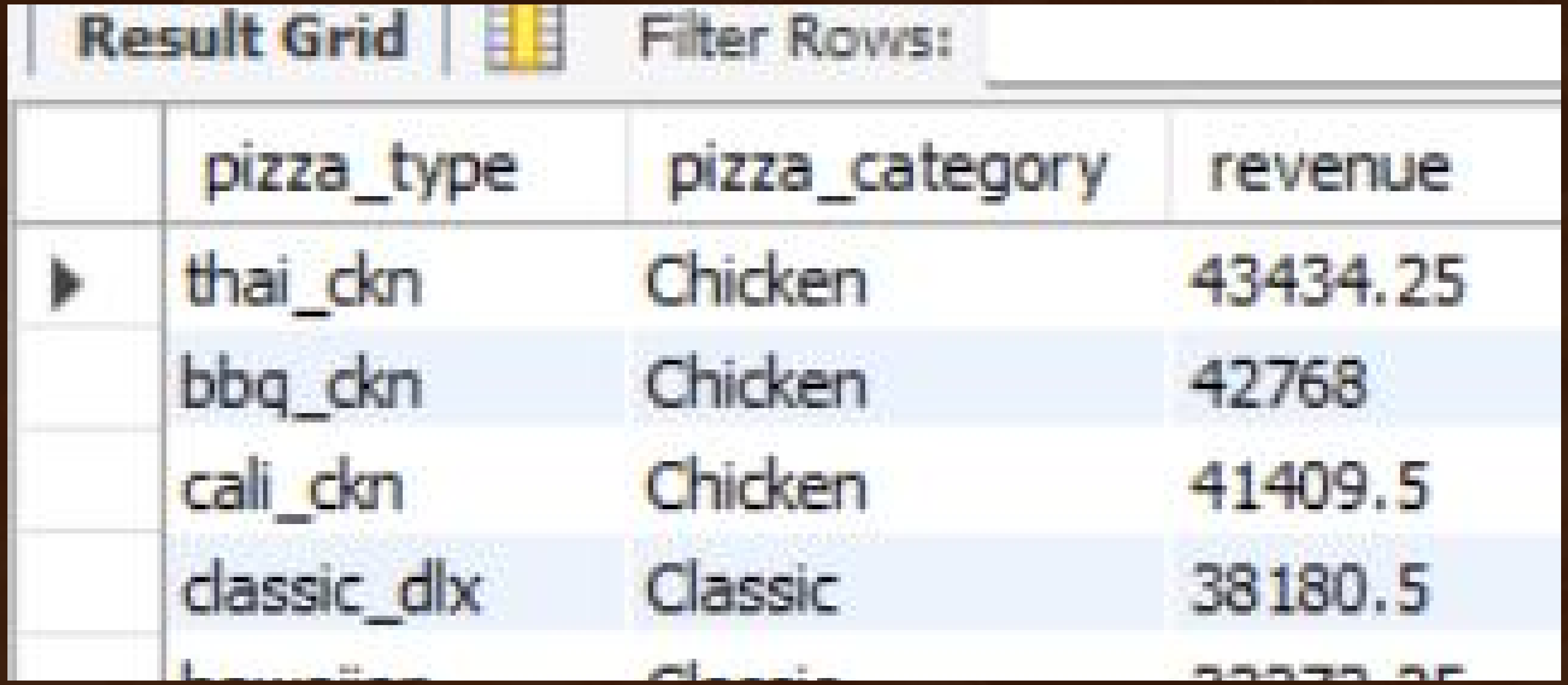
Result Grid | Filter Rows:

pizza_type	percentage_distribution
hawaiian	3.95
classic_dlx	4.67
five_cheese	3.19

12. DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.



```sql
WITH pizza_revenue AS (
    SELECT
        piztyp.pizza_type_id AS pizza_type,
        piztyp.category AS pizza_category,
        SUM(ordel.quantity * piz.price) AS revenue,
        RANK() OVER (PARTITION BY piztyp.category ORDER BY SUM(ordel.quantity * piz.price) DESC) AS ranks
    FROM order_details ordel
    JOIN pizzas piz ON piz.pizza_id = ordel.pizza_id
    JOIN pizza_types piztyp ON piztyp.pizza_type_id = piz.pizza_type_id
    GROUP BY piztyp.pizza_type_id, piztyp.category
)
SELECT pizza_type, pizza_category, revenue
FROM pizza_revenue
WHERE ranks <= 3
ORDER BY pizza_category, ranks;
```



```
   pizza_type    pizza_category   revenue
▶  thai_ckn      Chicken          43434.25
   bbq_ckn       Chicken          42768
   cali_ckn      Chicken          41409.5
   classic_dlx   Classic          38180.5
```

TRIED THE SQL PIZZA INSIGHTS PROJECT? YOUR FEEDBACK MATTERS! 🍕

👉 WHAT WORKED WELL? WHAT CAN BE IMPROVED?

💡 DROP YOUR THOUGHTS IN THE COMMENTS!

THANK YOU FOR ATTENTION