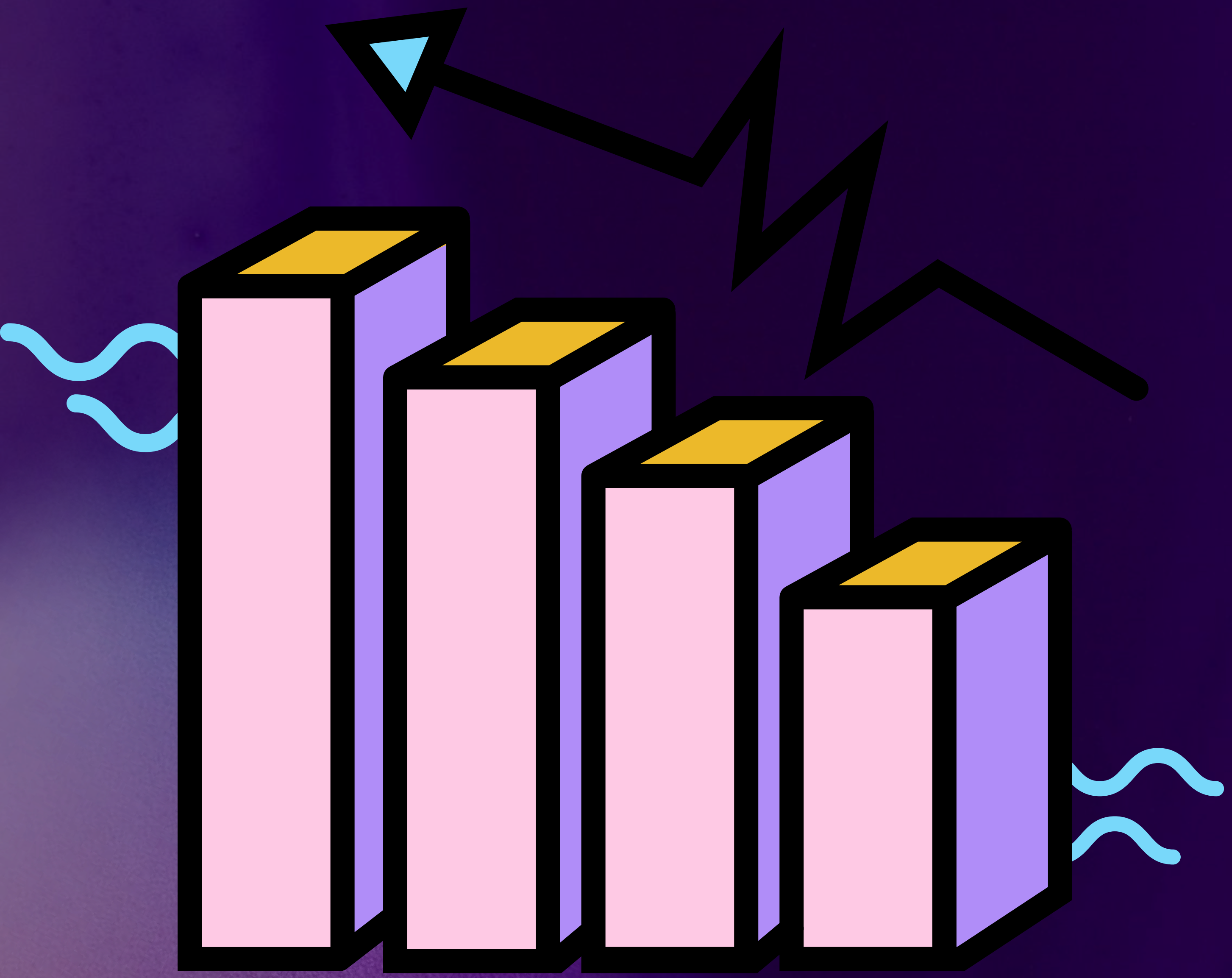


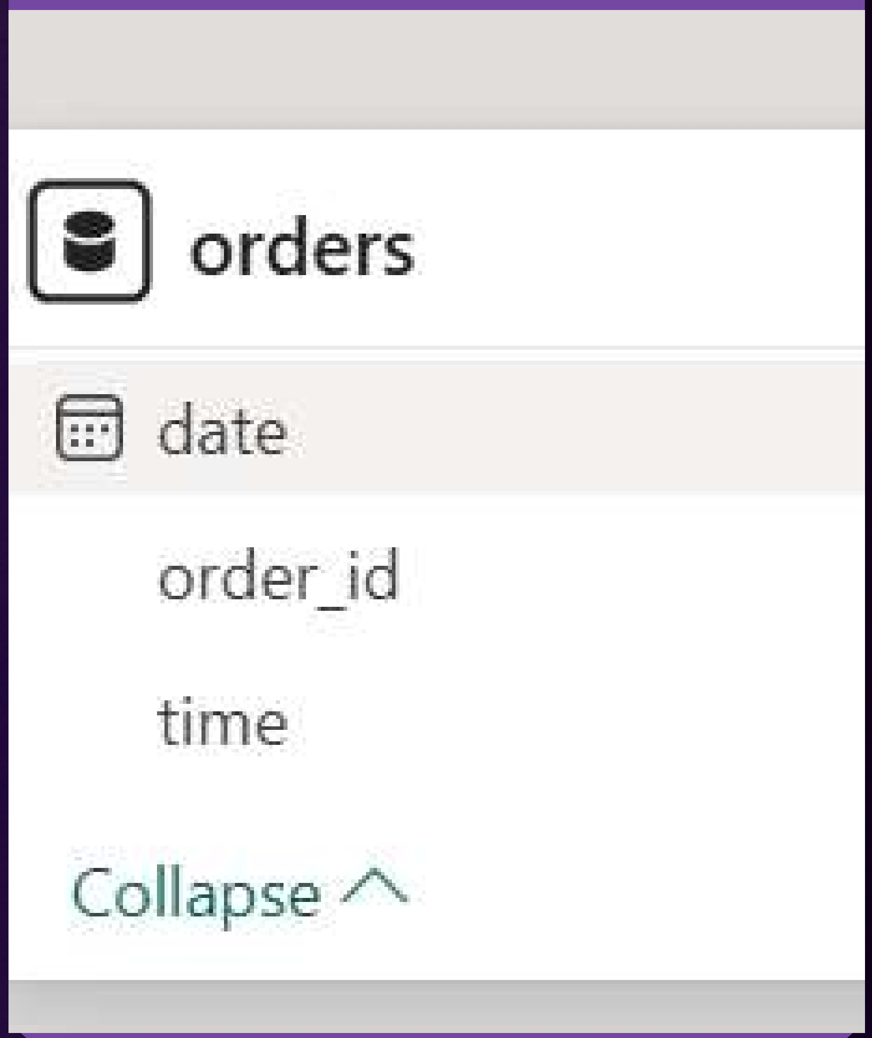
PIZZA SALES ANALYSIS USING SQL

BY HEMLATA SAKLA

Hi, I’m Hemlata Sakla. For this project,
I worked on a pizza sales dataset using SQL. I explored the data through different SQL queries to find out things like how much revenue the business made, which pizza types were the most popular, how sales varied by category, and how the revenue grew over time.
It was a fun, hands-on way to apply SQL to a real-world business scenario and understand how data can help in making better decisions.

Data Schemas

Orders

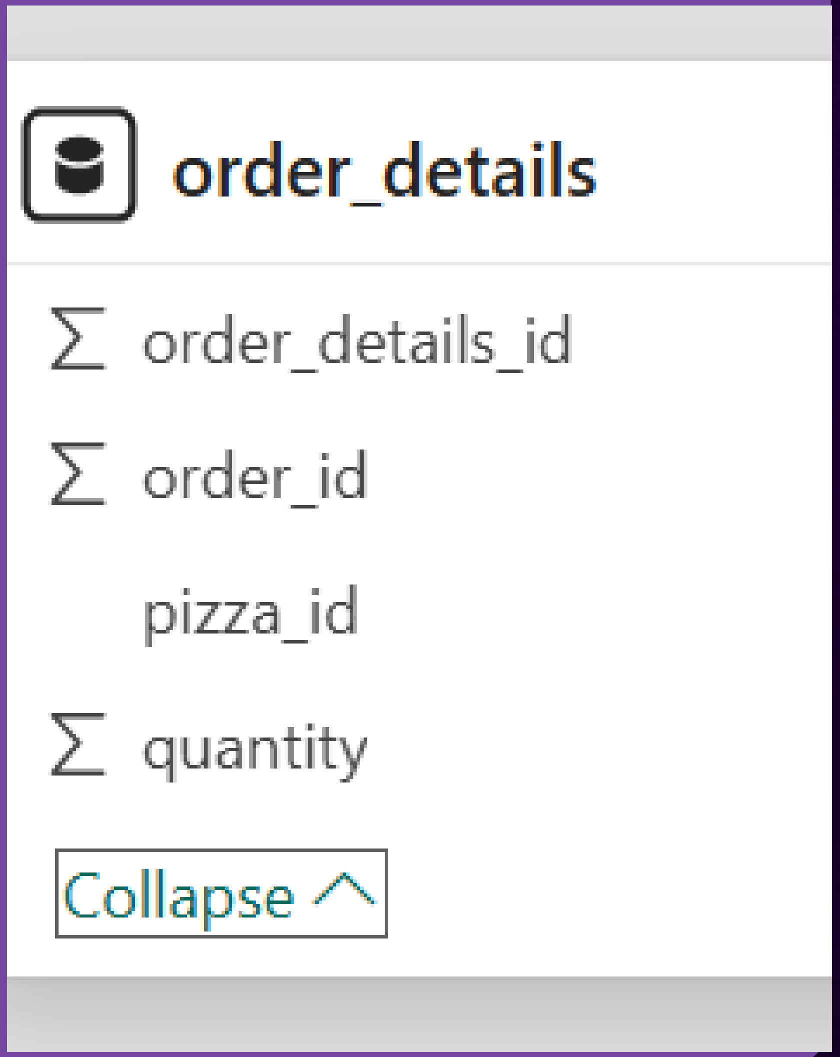


orders

- date
- order_id
- time

Collapse

Orders_details

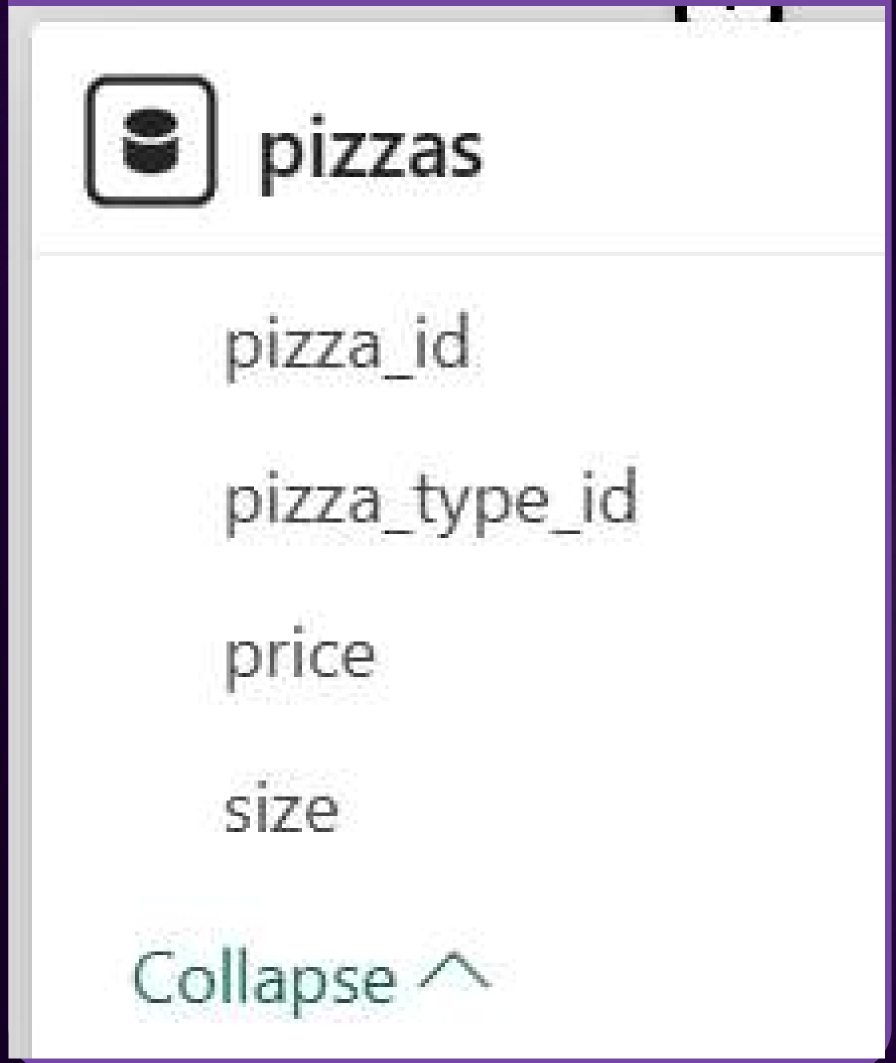


order_details

- ∑ order_details_id
- ∑ order_id
- pizza_id
- ∑ quantity

Collapse

Pizzas

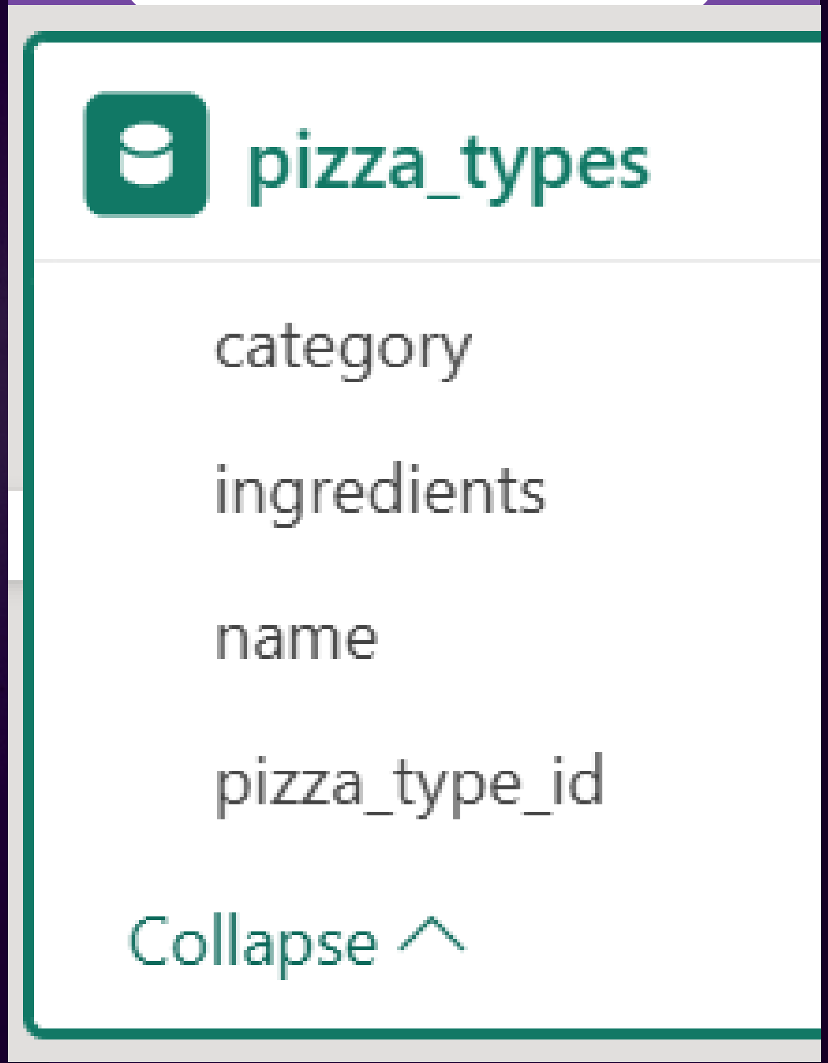


pizzas

- pizza_id
- pizza_type_id
- price
- size

Collapse

Pizza_types



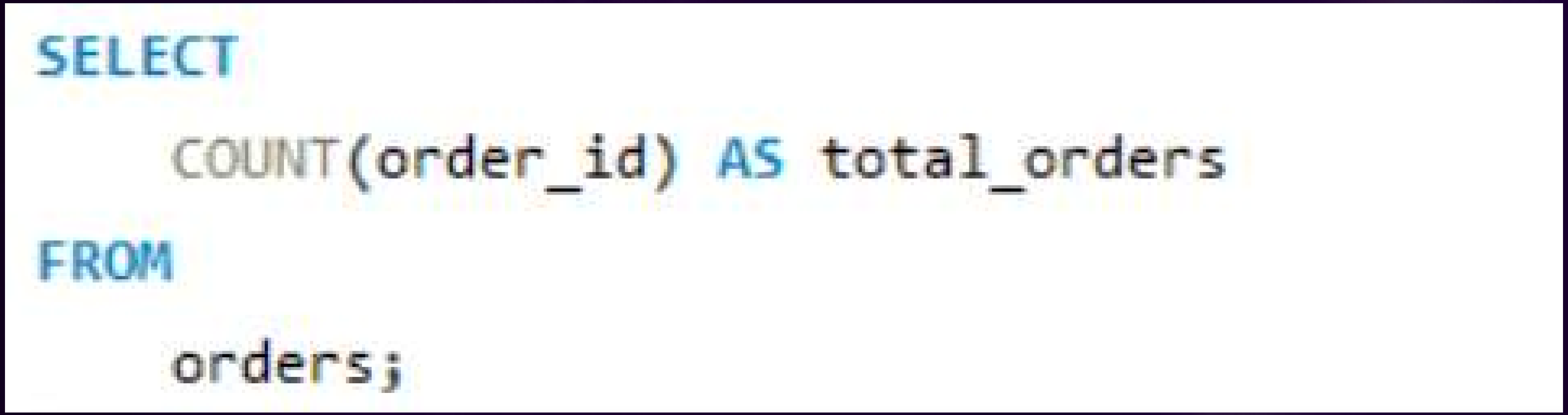
pizza_types

- category
- ingredients
- name
- pizza_type_id

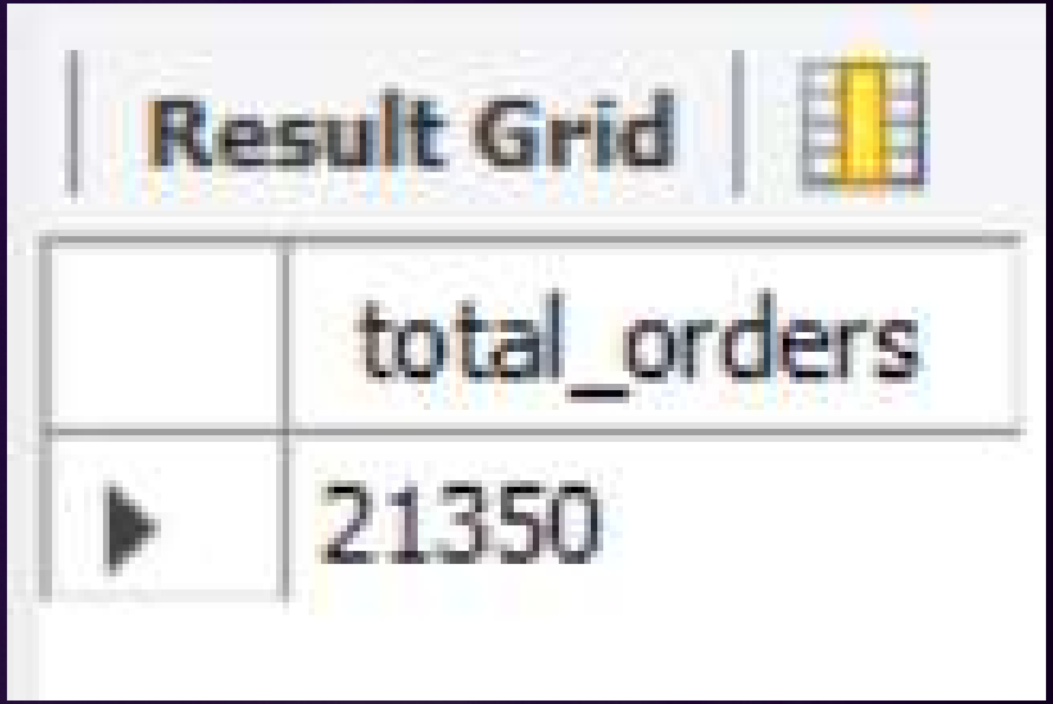
Collapse

Retrieve the total number of orders placed.

Input



```
SELECT
    COUNT(order_id) AS total_orders
FROM
    orders;
```

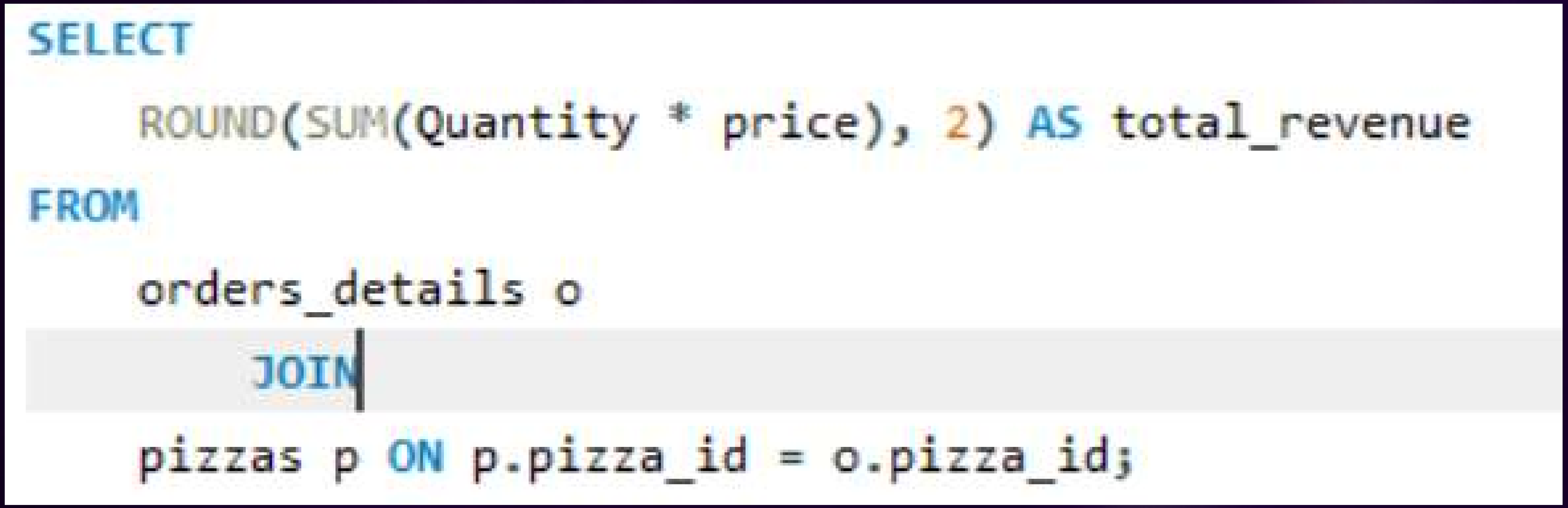


```
total_orders
21350
```

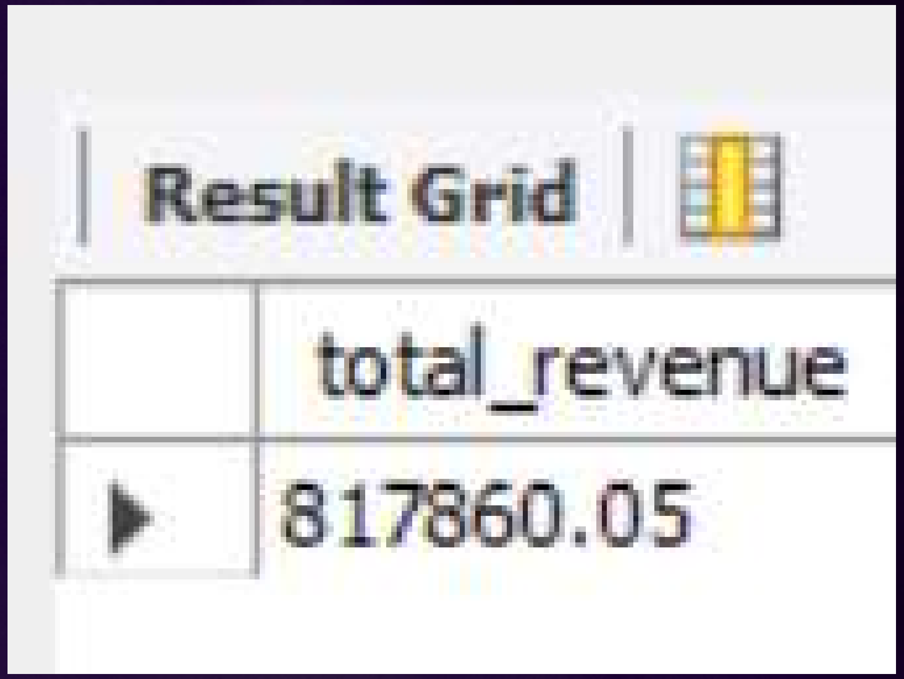
Output

Calculate the total revenue generated from pizza sales.

Input



```
SELECT
    ROUND(SUM(Quantity * price), 2) AS total_revenue
FROM
    orders_details o
        JOIN
    pizzas p ON p.pizza_id = o.pizza_id;
```

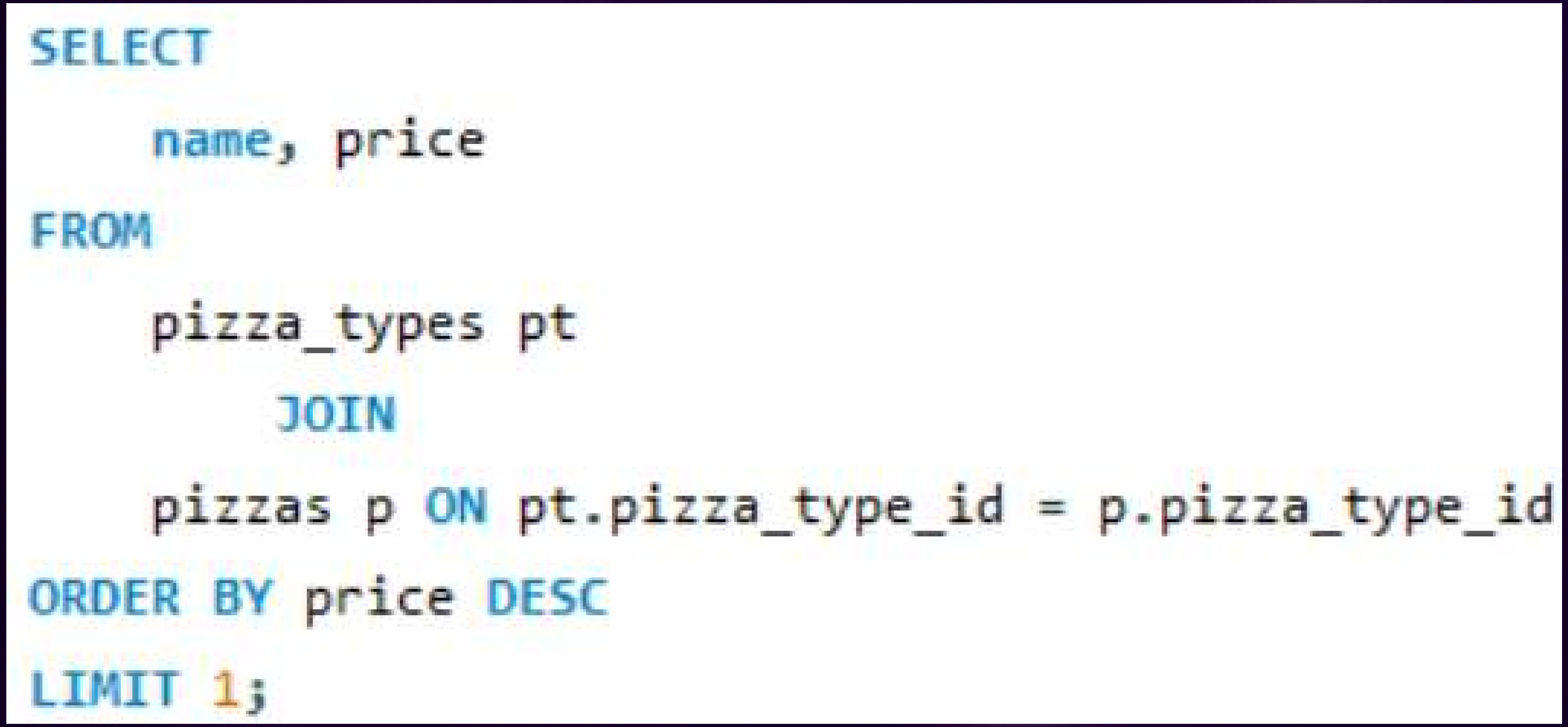


```
total_revenue
817860.05
```

Output

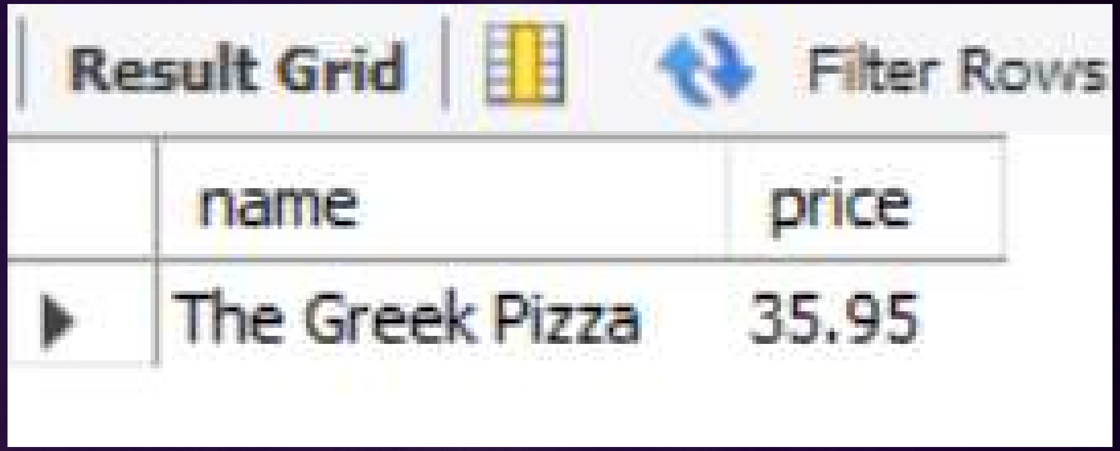
Identify the highest-priced pizza.

Input



```
SELECT
    name, price
FROM
    pizza_types pt
        JOIN
    pizzas p ON pt.pizza_type_id = p.pizza_type_id
ORDER BY price DESC
LIMIT 1;
```

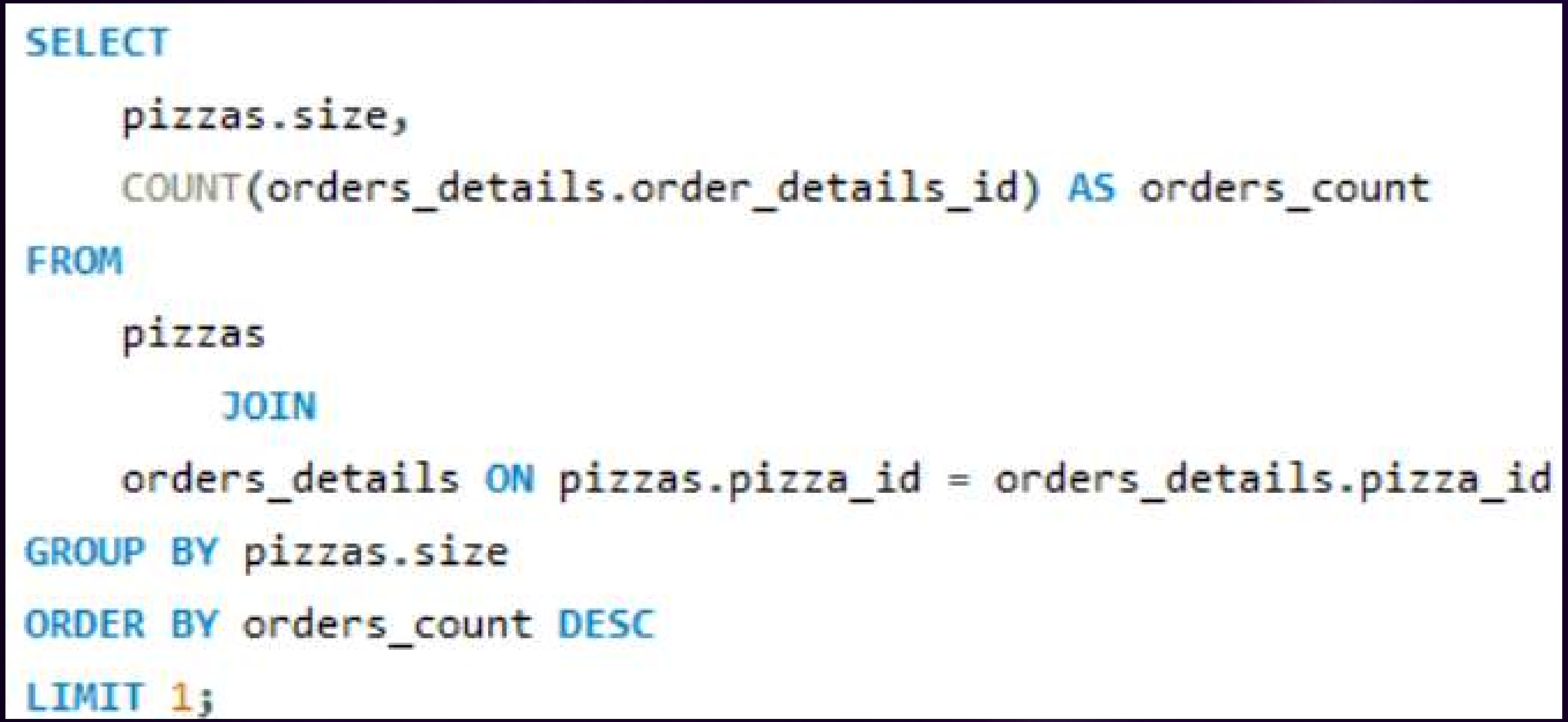
Output



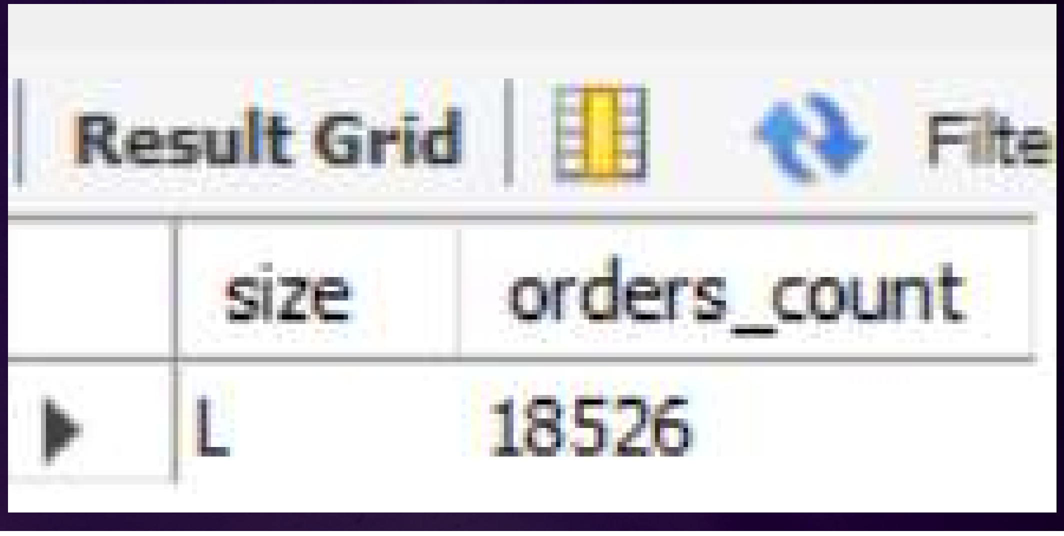
```
  name              price
▶ The Greek Pizza   35.95
```

Identify the most common pizza size ordered.

Input



```
SELECT
    pizzas.size,
    COUNT(orders_details.order_details_id) AS orders_count
FROM
    pizzas
        JOIN
    orders_details ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY pizzas.size
ORDER BY orders_count DESC
LIMIT 1;
```

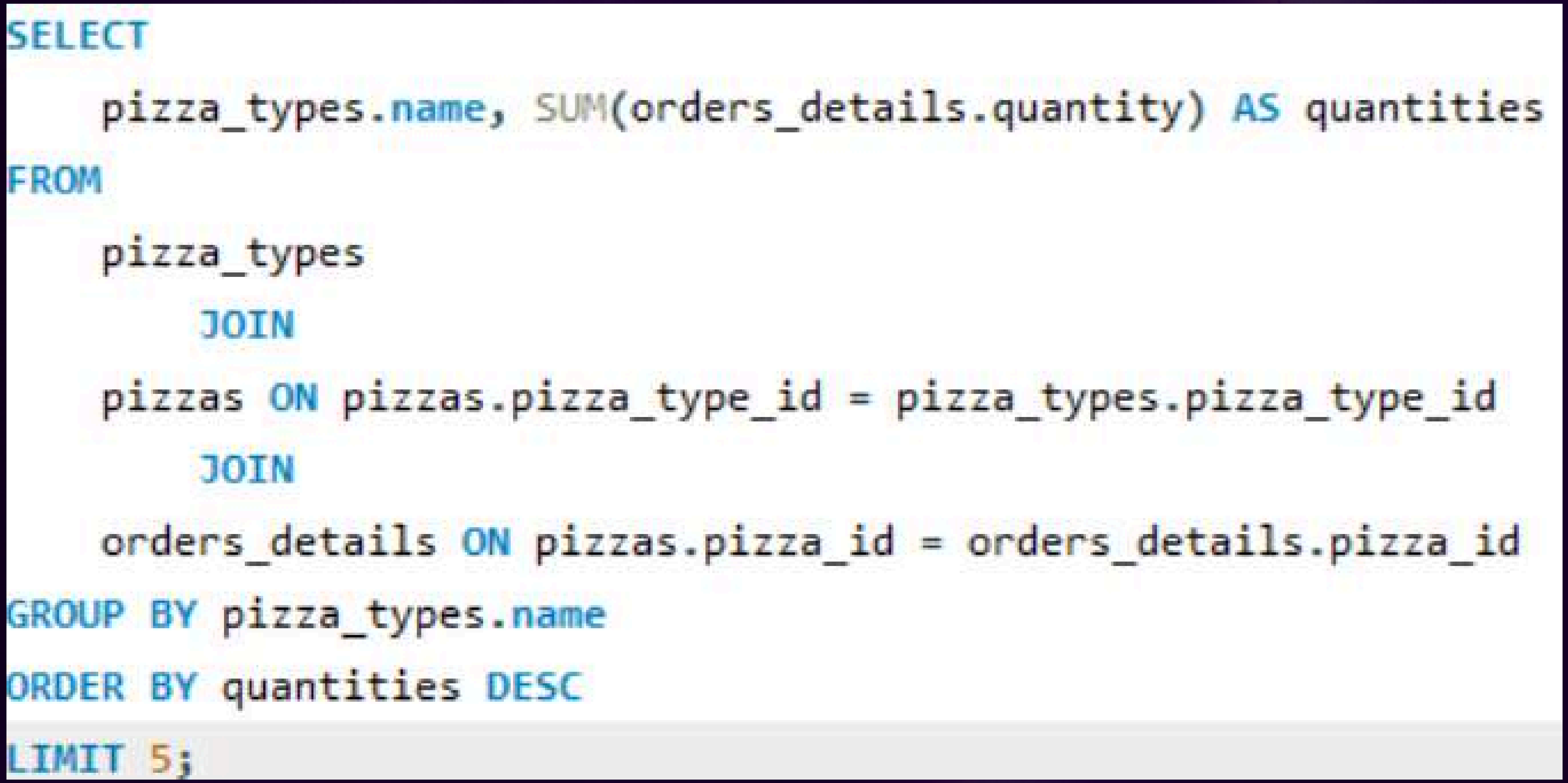


```
size   orders_count
L      18526
```

Output

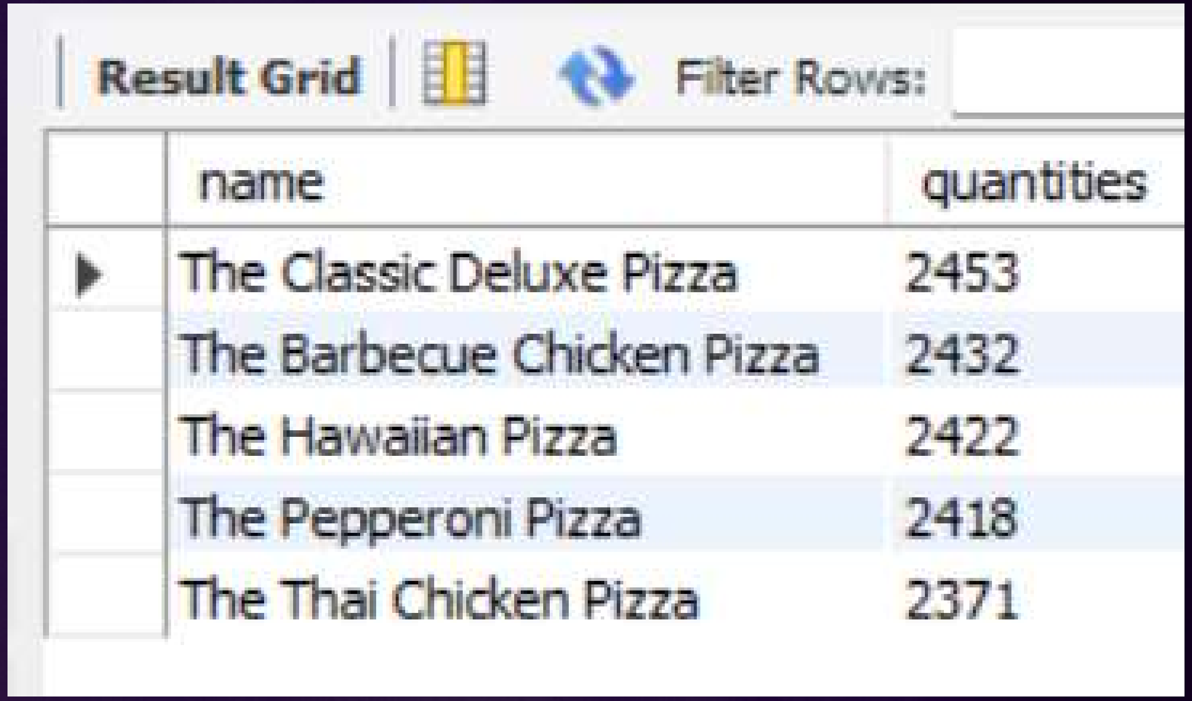
List the top 5 most ordered pizza types along with their quantities.

Input



```
SELECT
    pizza_types.name, SUM(orders_details.quantity) AS quantities
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
        JOIN
    orders_details ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY pizza_types.name
ORDER BY quantities DESC
LIMIT 5;
```

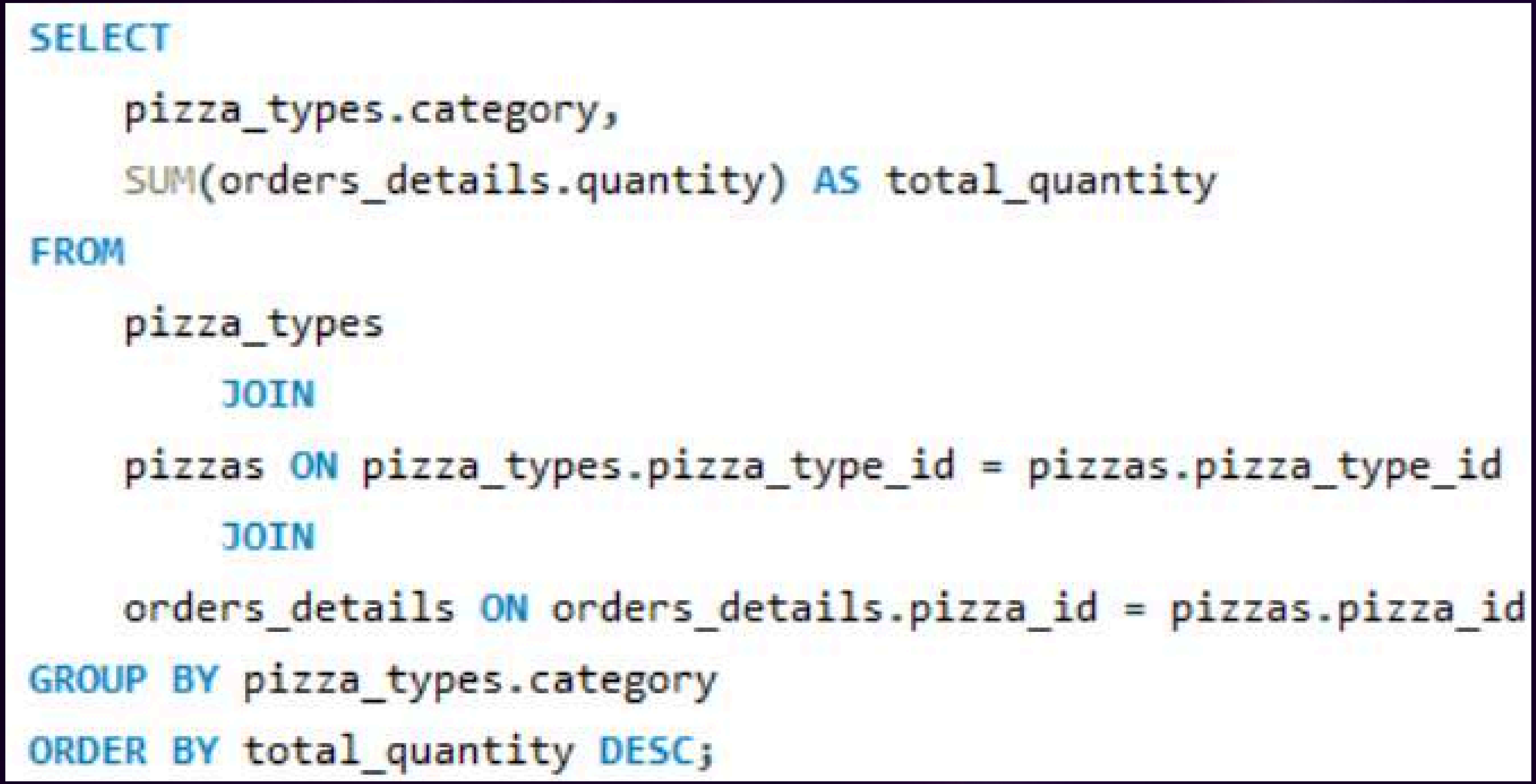
Output

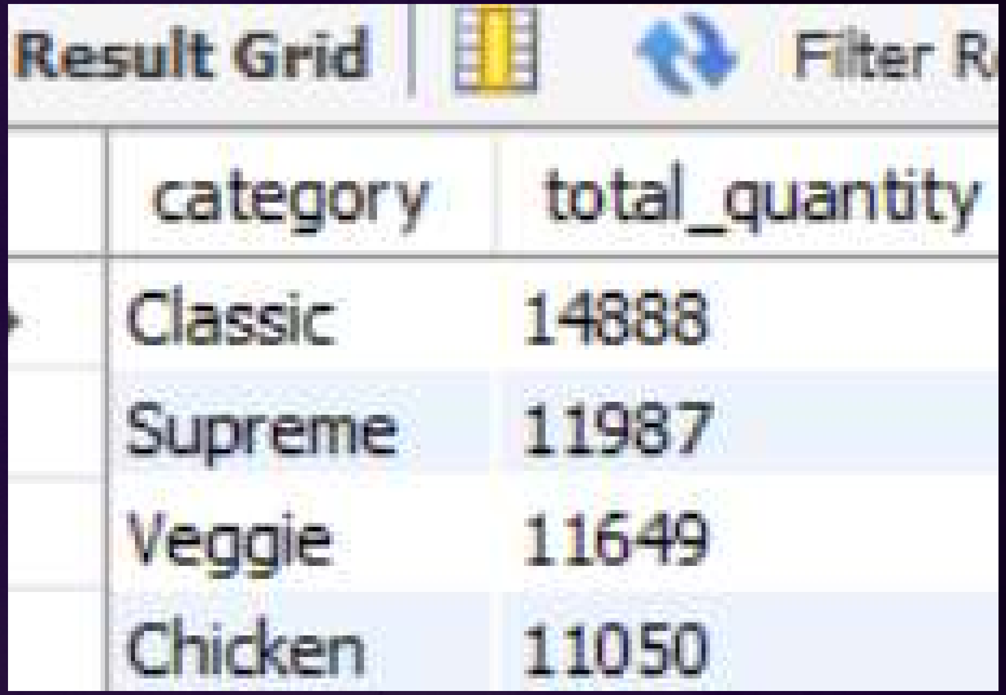


name	quantities
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

Join the necessary tables to find the total quantity of each pizza category ordered.

Input


```
SELECT
    pizza_types.category,
    SUM(orders_details.quantity) AS total_quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY total_quantity DESC;
```

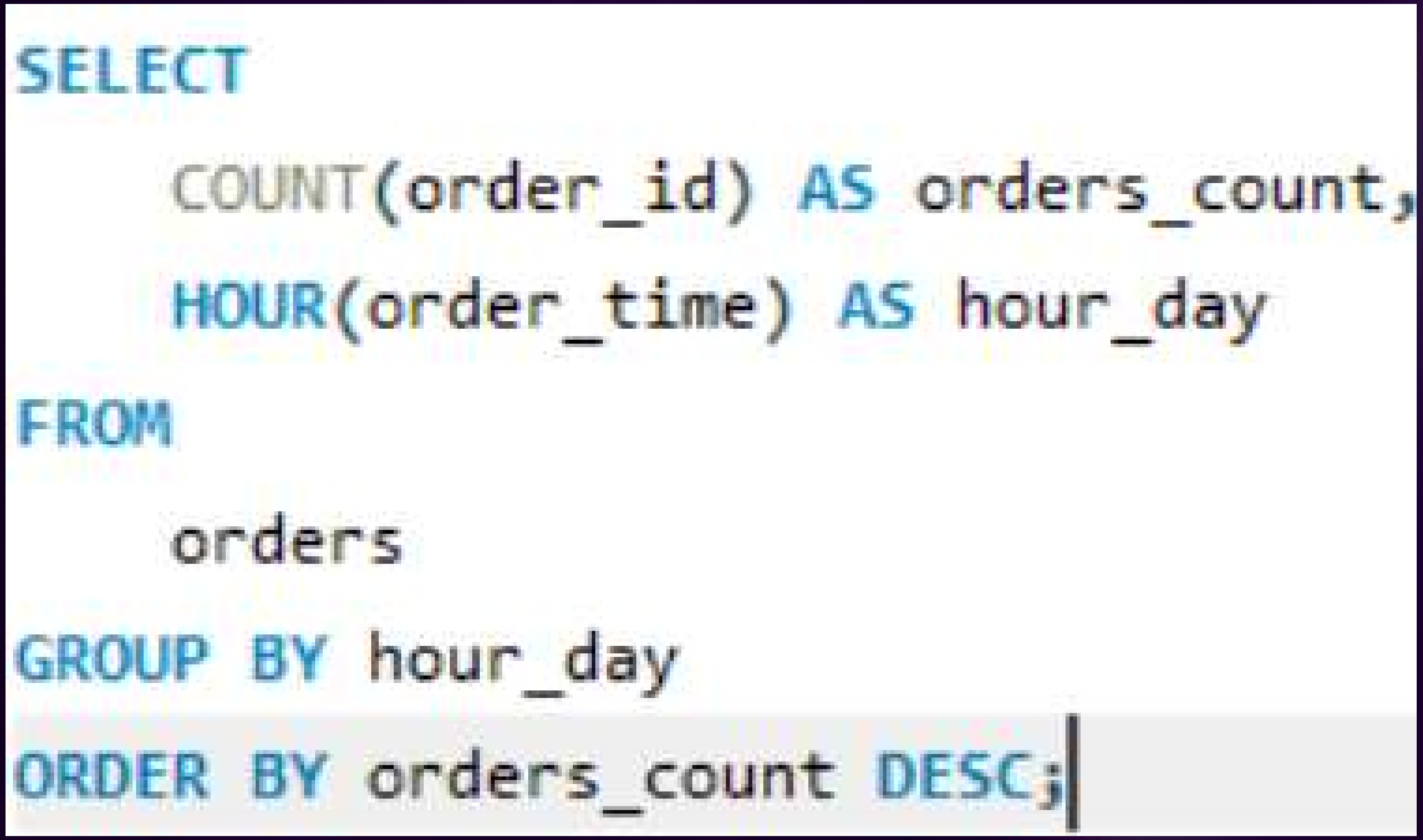


category	total_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Output

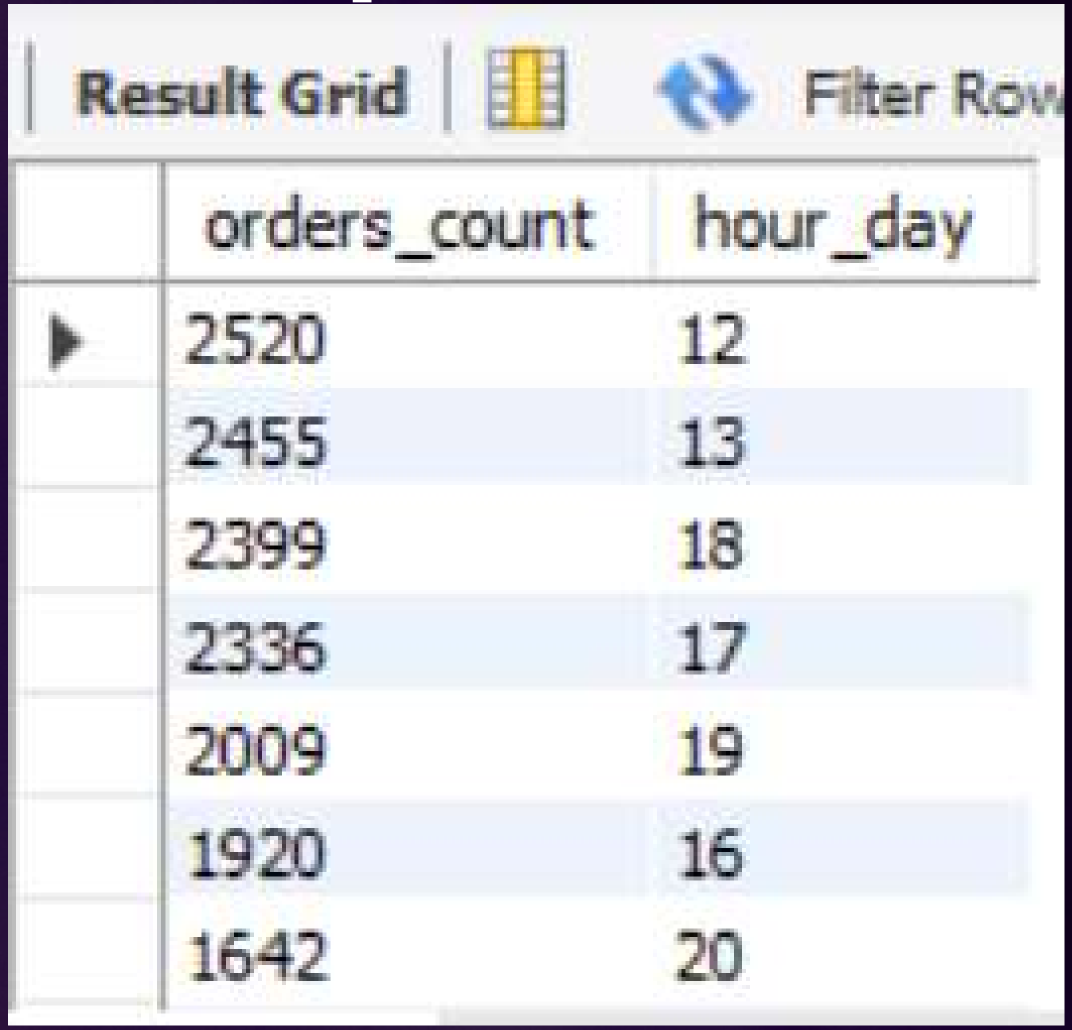
Determine the distribution of orders by hour of the day.

Input



```
SELECT
    COUNT(order_id) AS orders_count,
    HOUR(order_time) AS hour_day
FROM
    orders
GROUP BY hour_day
ORDER BY orders_count DESC;
```

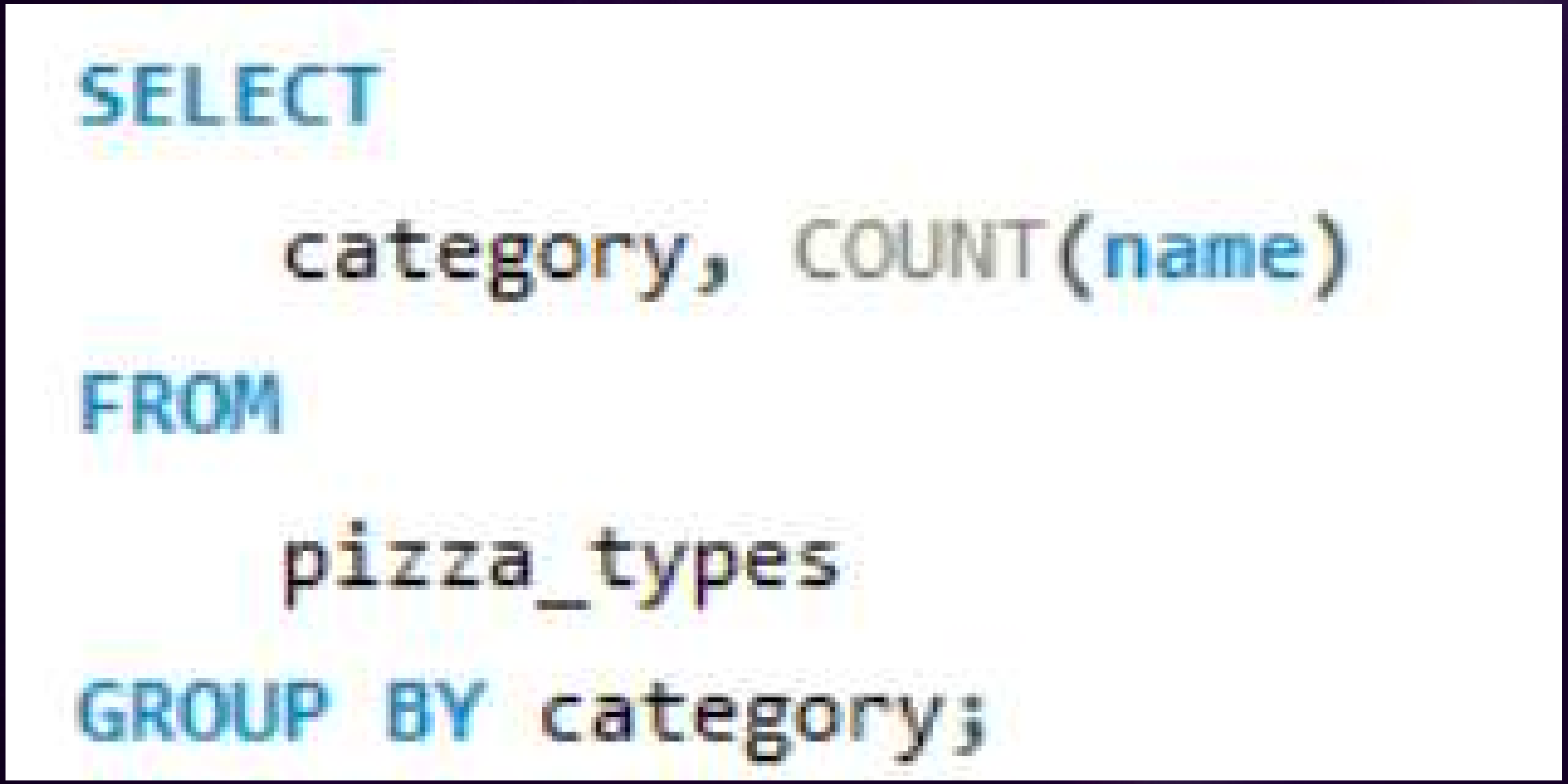
Output



orders_count	hour_day
2520	12
2455	13
2399	18
2336	17
2009	19
1920	16
1642	20

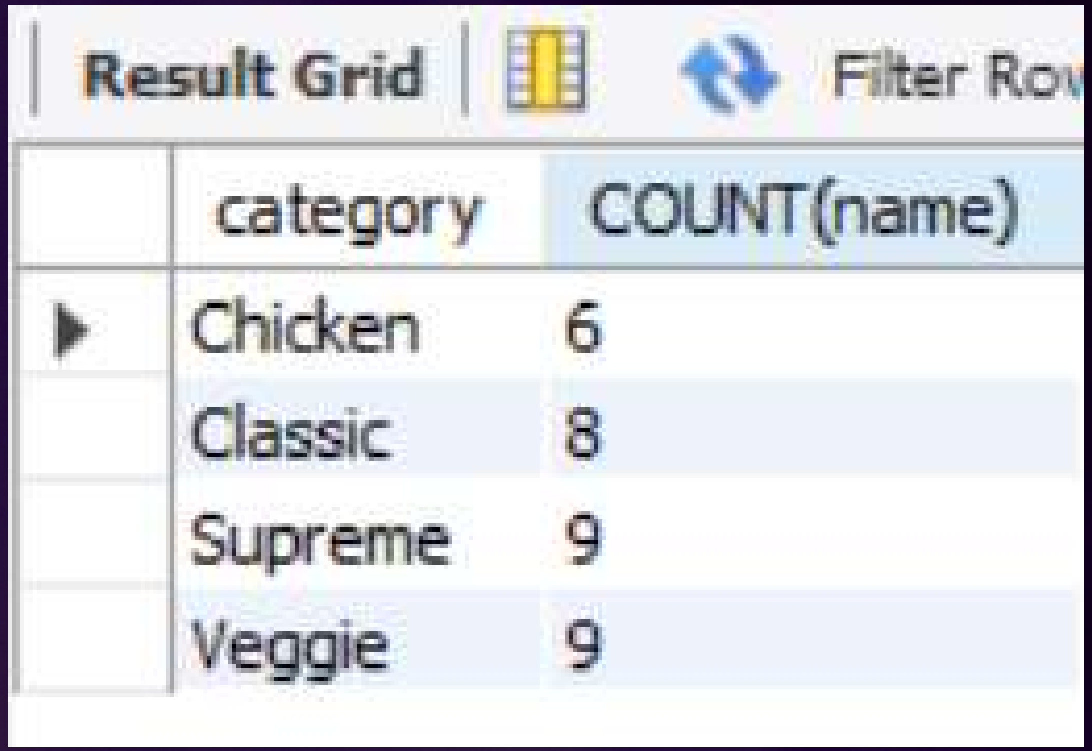
Join relevant tables to find the category-wise distribution of pizzas.

Input



```
SELECT
    category, COUNT(name)
FROM
    pizza_types
GROUP BY category;
```

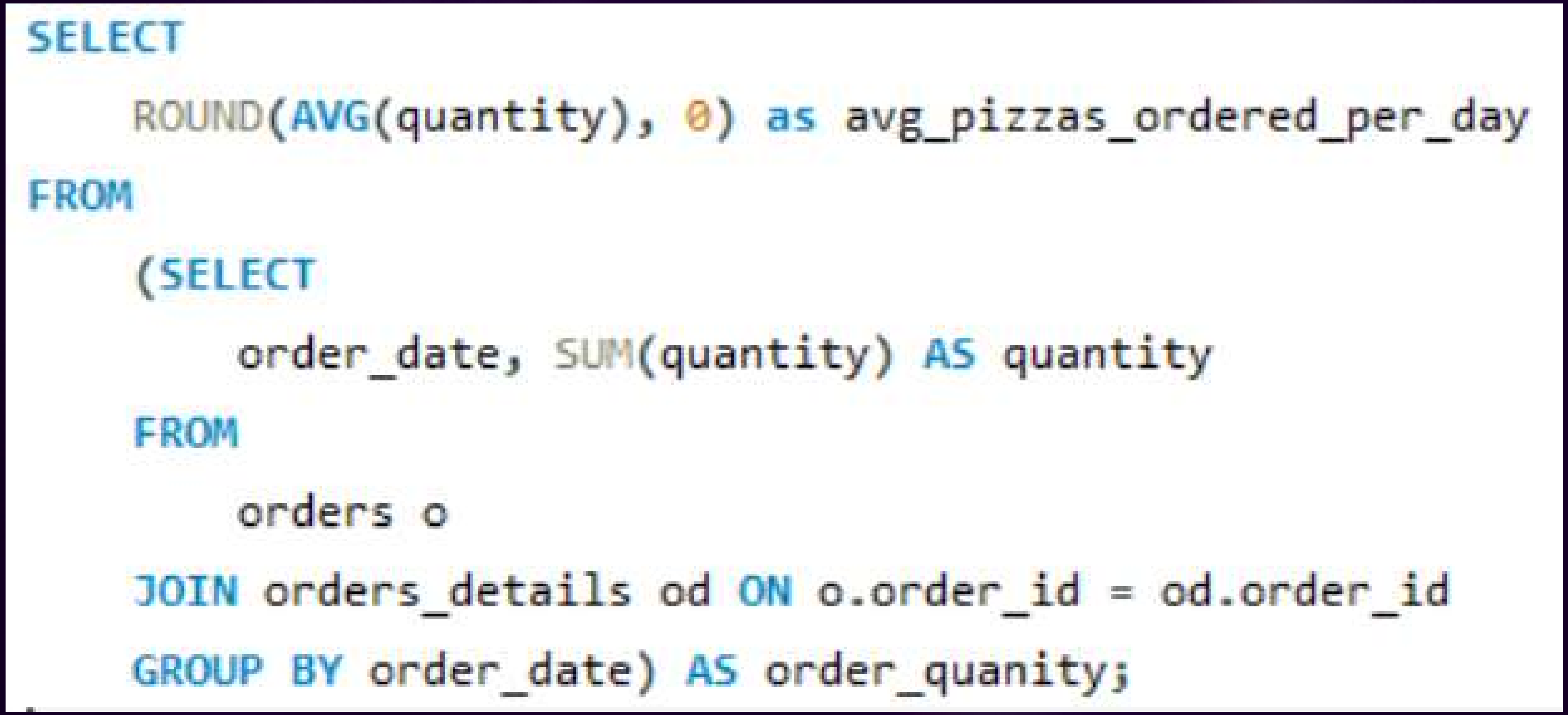
Output



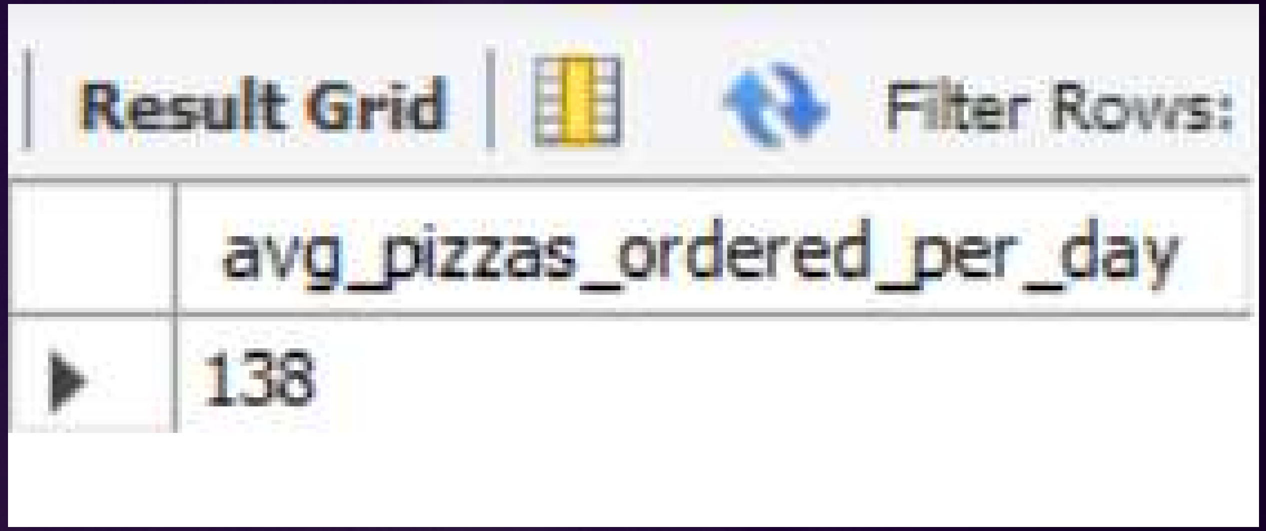
category	COUNT(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

Group the orders by date and calculate the average number of pizzas ordered per day.

Input



```
SELECT
    ROUND(AVG(quantity), 0) as avg_pizzas_ordered_per_day
FROM
    (SELECT
        order_date, SUM(quantity) AS quantity
    FROM
        orders o
    JOIN orders_details od ON o.order_id = od.order_id
    GROUP BY order_date) AS order_quanity;
```

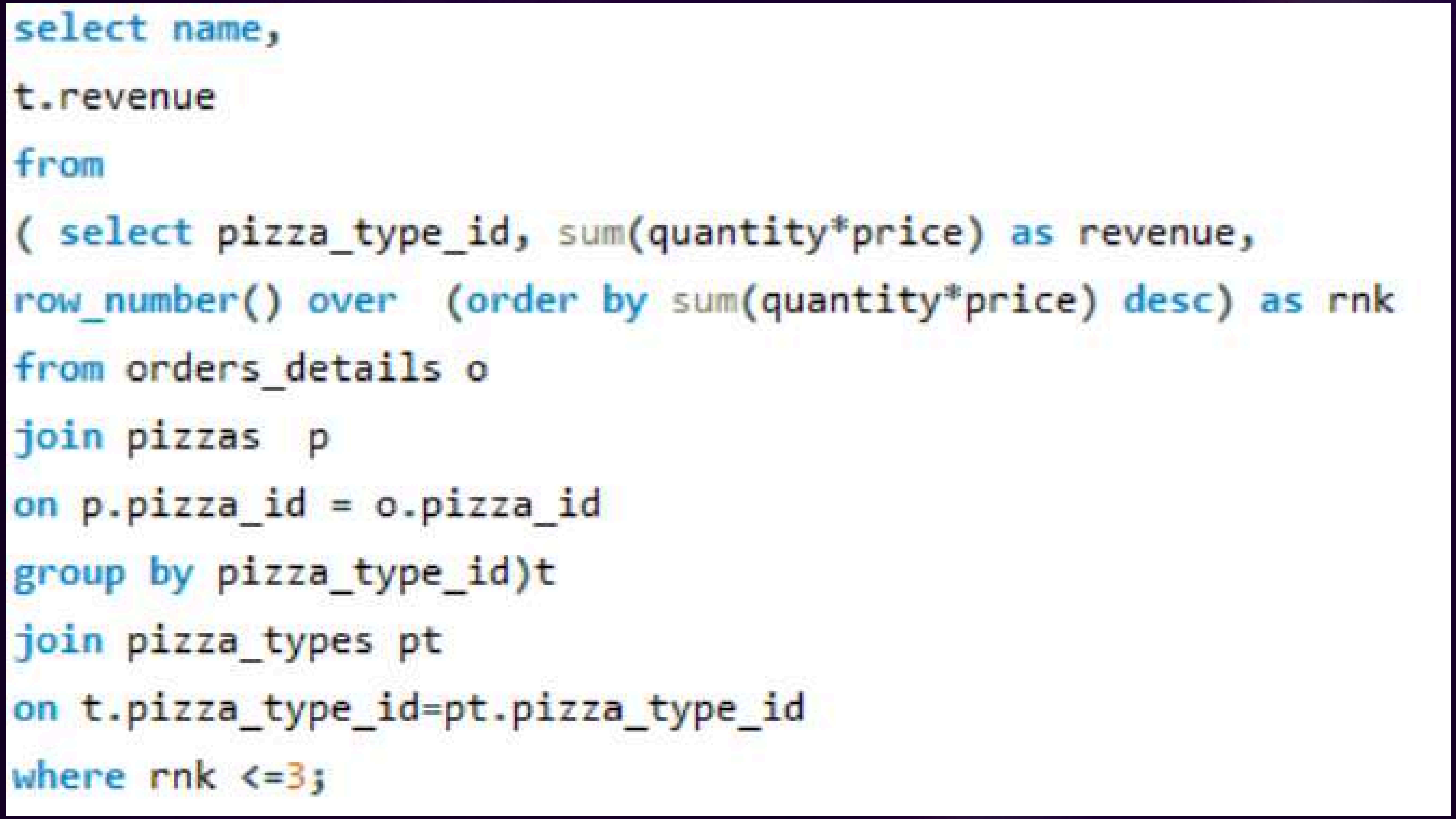


```
avg_pizzas_ordered_per_day
138
```

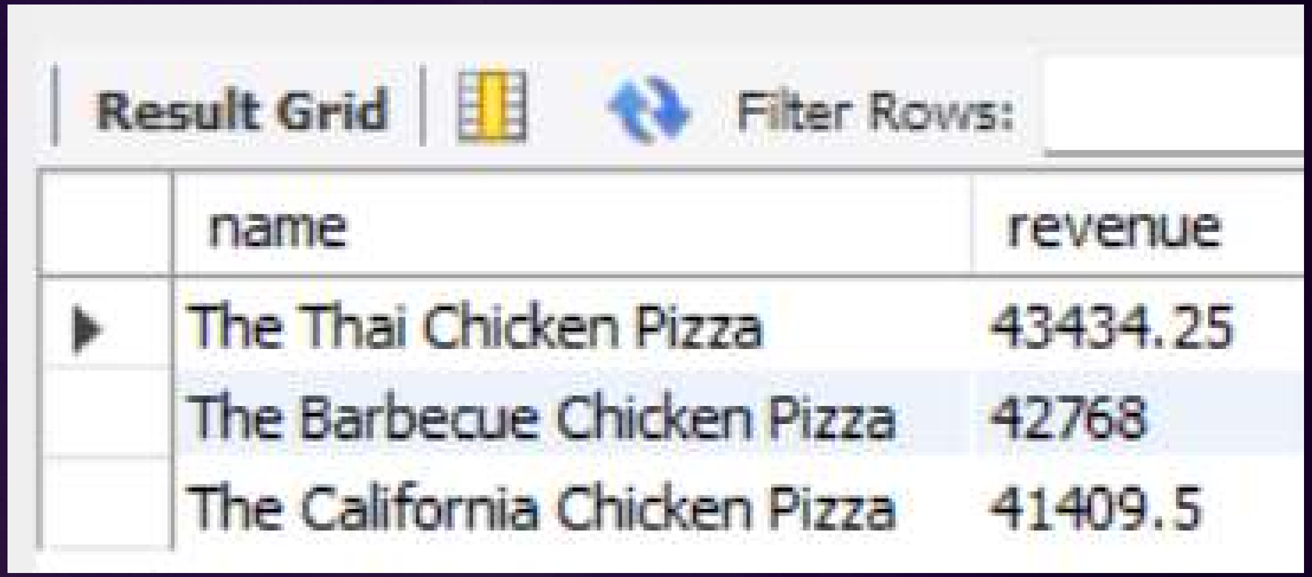
Output

Determine the top 3 most ordered pizza types based on revenue.

Input



```
select name,
t.revenue
from
( select pizza_type_id, sum(quantity*price) as revenue,
row_number() over  (order by sum(quantity*price) desc) as rnk
from orders_details o
join pizzas  p
on p.pizza_id = o.pizza_id
group by pizza_type_id)t
join pizza_types pt
on t.pizza_type_id=pt.pizza_type_id
where rnk <=3;
```

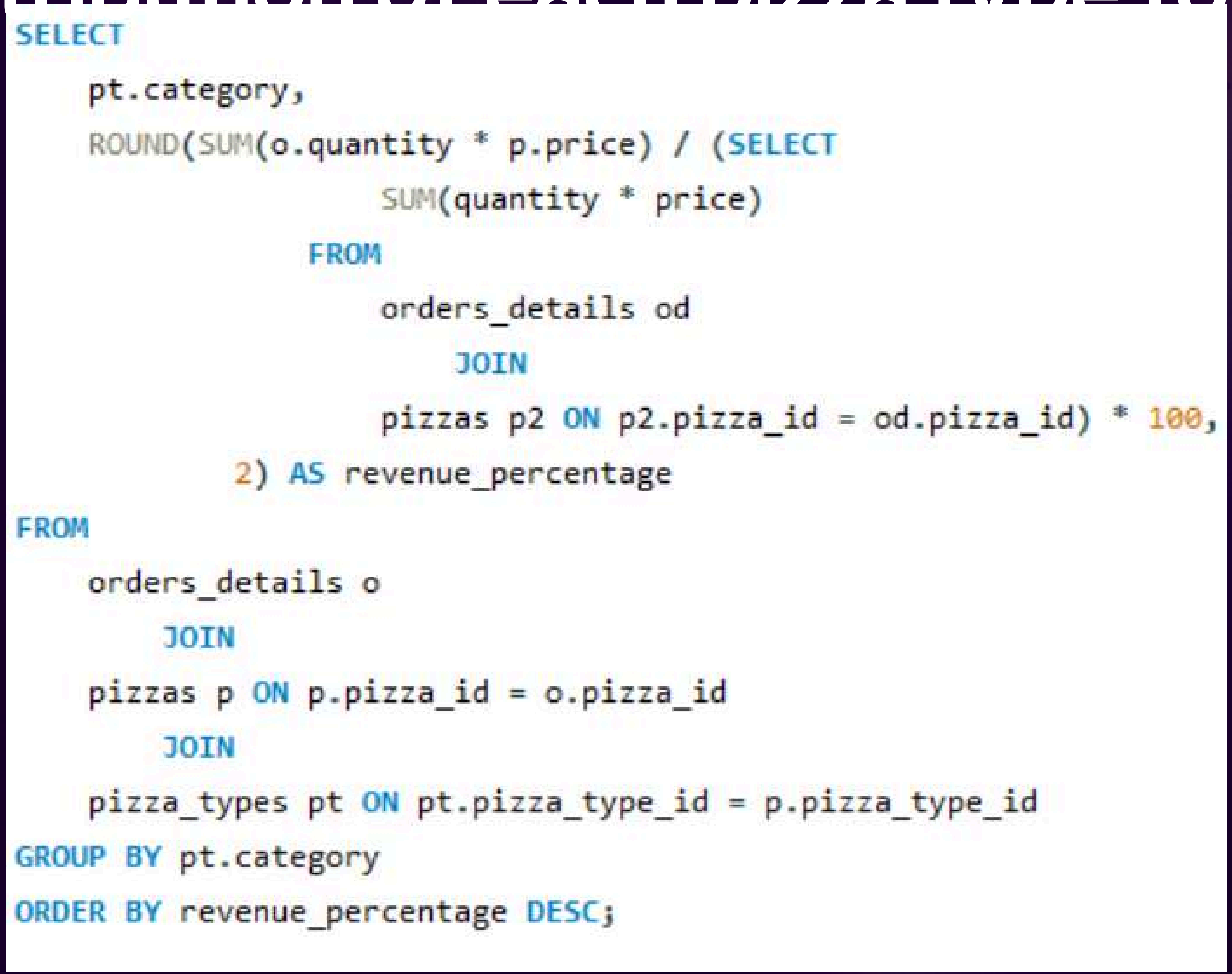


name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

Output

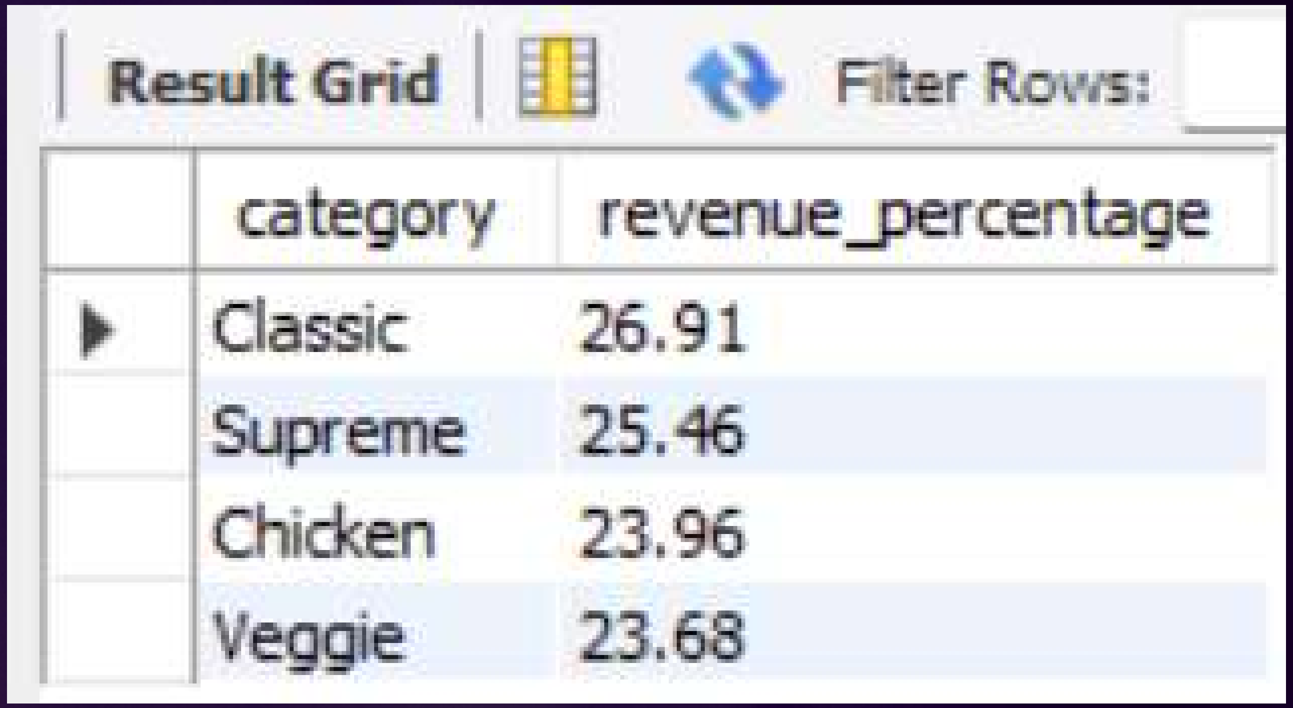
Calculate the percentage contribution of each pizza type to total revenue.

Input



```
SELECT
    pt.category,
    ROUND(SUM(o.quantity * p.price) / (SELECT
                    SUM(quantity * price)
                FROM
                    orders_details od
                        JOIN
                    pizzas p2 ON p2.pizza_id = od.pizza_id) * 100,
            2) AS revenue_percentage
FROM
    orders_details o
        JOIN
    pizzas p ON p.pizza_id = o.pizza_id
        JOIN
    pizza_types pt ON pt.pizza_type_id = p.pizza_type_id
GROUP BY pt.category
ORDER BY revenue_percentage DESC;
```

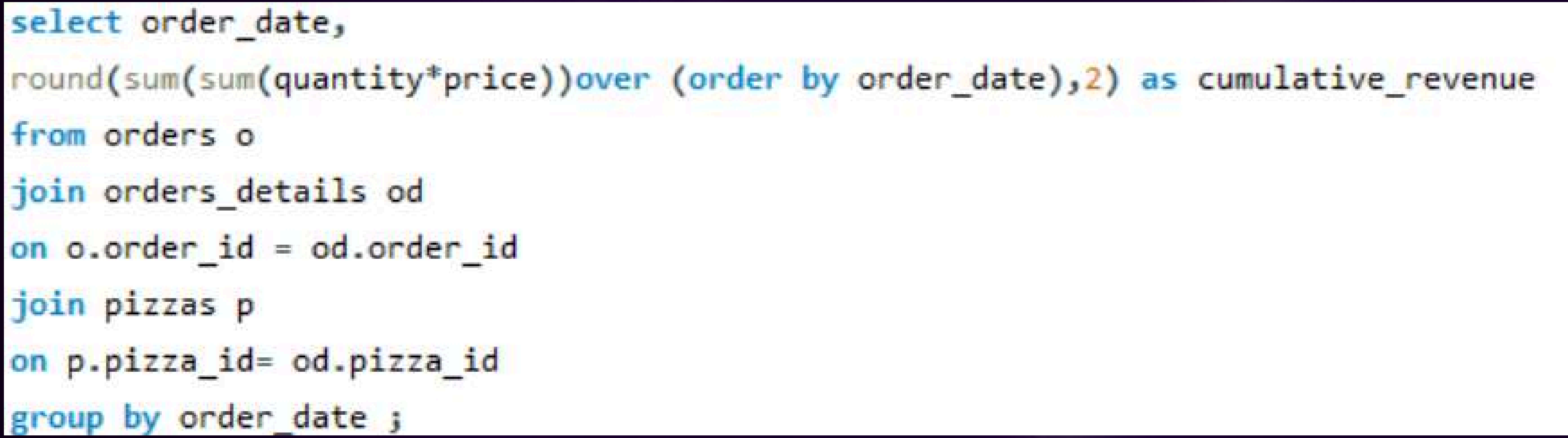
Output



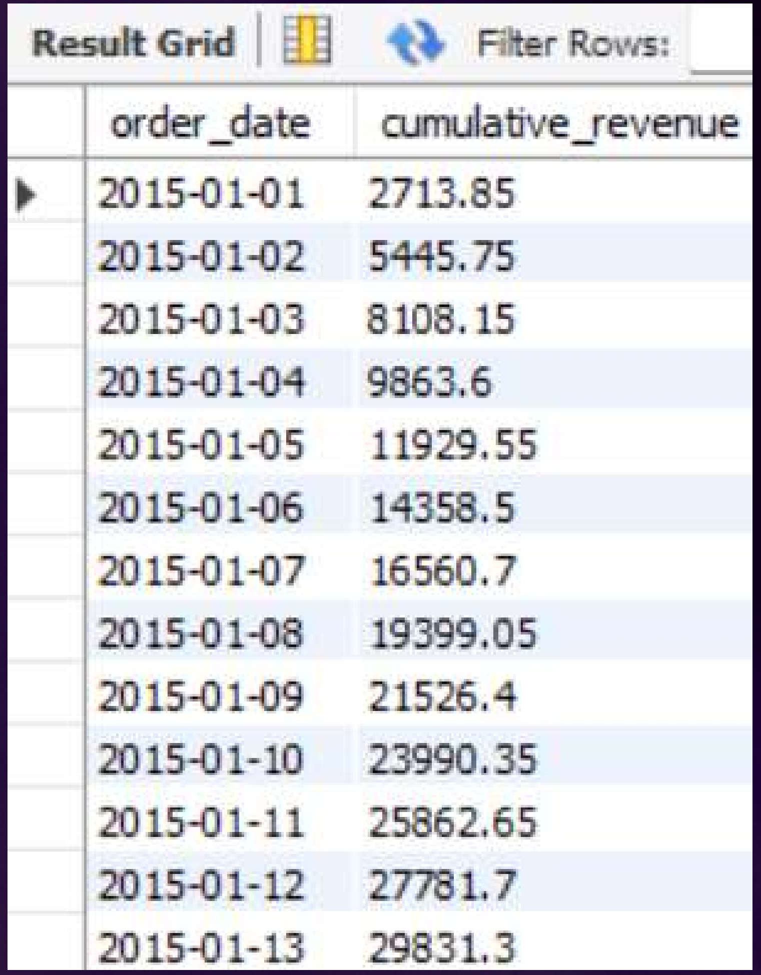
category	revenue_percentage
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

Analyze the cumulative revenue generated over time.

Input



```
select order_date,
round(sum(sum(quantity*price))over (order by order_date),2) as cumulative_revenue
from orders o
join orders_details od
on o.order_id = od.order_id
join pizzas p
on p.pizza_id= od.pizza_id
group by order_date ;
```

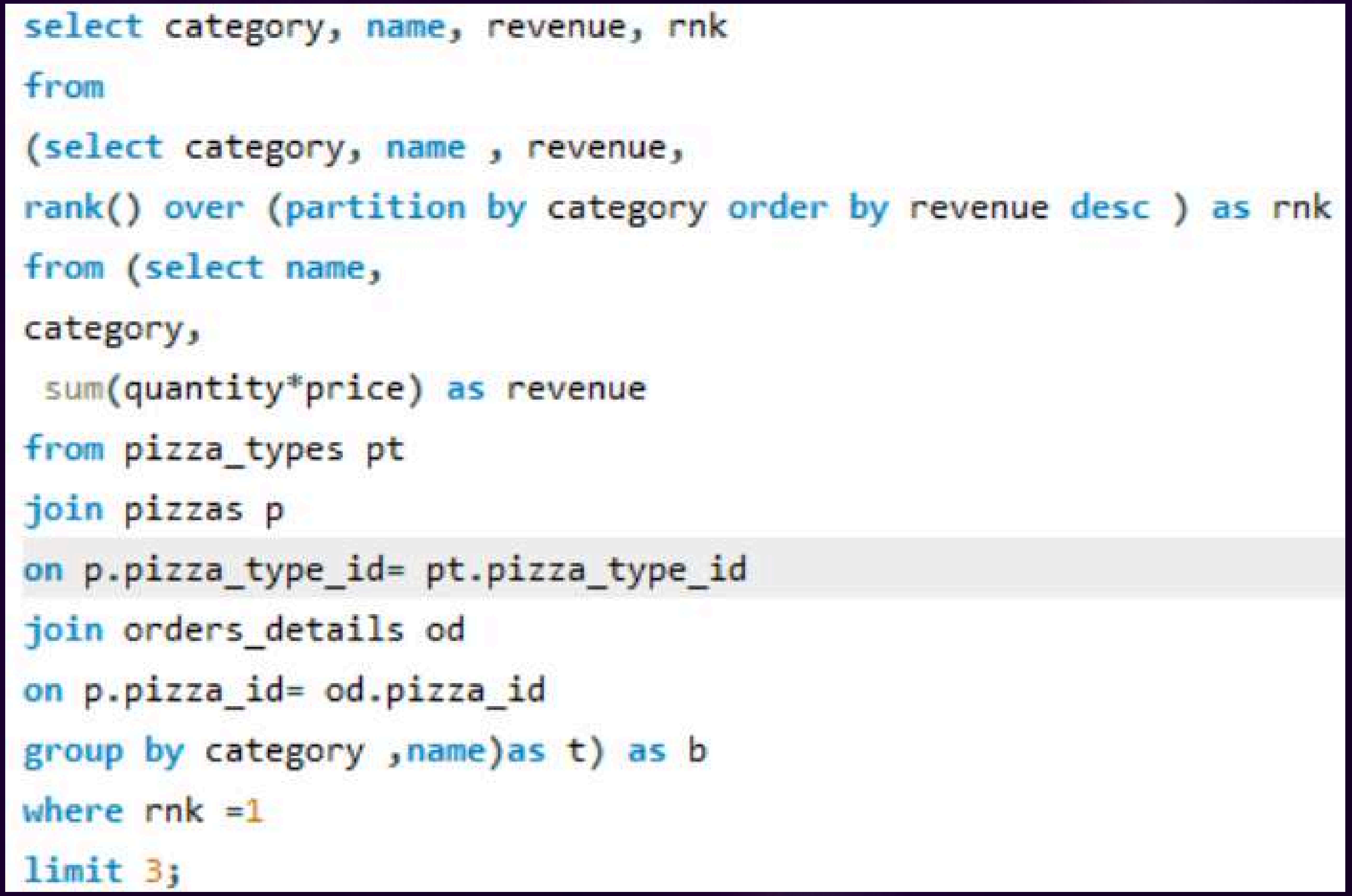


order_date	cumulative_revenue
2015-01-01	2713.85
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55
2015-01-06	14358.5
2015-01-07	16560.7
2015-01-08	19399.05
2015-01-09	21526.4
2015-01-10	23990.35
2015-01-11	25862.65
2015-01-12	27781.7
2015-01-13	29831.3

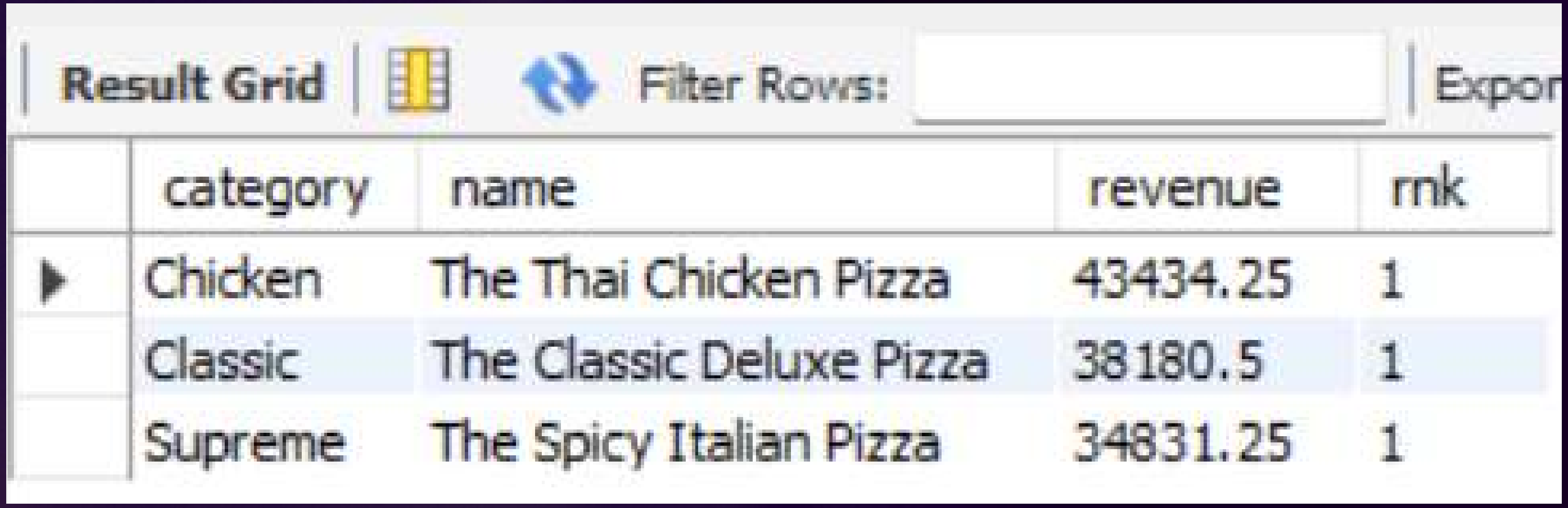
Output

Determine the top 3 most ordered pizza types based on revenue for each pizza category.

Input



```
select category, name, revenue, rnk
from
(select category, name , revenue,
rank() over (partition by category order by revenue desc ) as rnk
from (select name,
category,
 sum(quantity*price) as revenue
from pizza_types pt
join pizzas p
on p.pizza_type_id= pt.pizza_type_id
join orders_details od
on p.pizza_id= od.pizza_id
group by category ,name)as t) as b
where rnk =1
limit 3;
```



```
category  name                       revenue   rnk
Chicken   The Thai Chicken Pizza     43434.25  1
Classic   The Classic Deluxe Pizza   38180.5   1
Supreme   The Spicy Italian Pizza    34831.25  1
```

Output

Thank You for Watching! 🚀

This MySQL project has been an exciting ride!
💡 I got to play around with real data, write meaningful SQL queries, and see how even simple logic can bring powerful results.
The screenshots and outputs you saw — that's all part of the hands-on work I enjoyed doing.
Honestly, it wasn't just about writing queries — it was about understanding data better, thinking logically, and having fun along the way!
I'm grateful for what I've learned so far — and this is just the beginning.
Excited to explore more, improve more, and take my database skills to the next level!