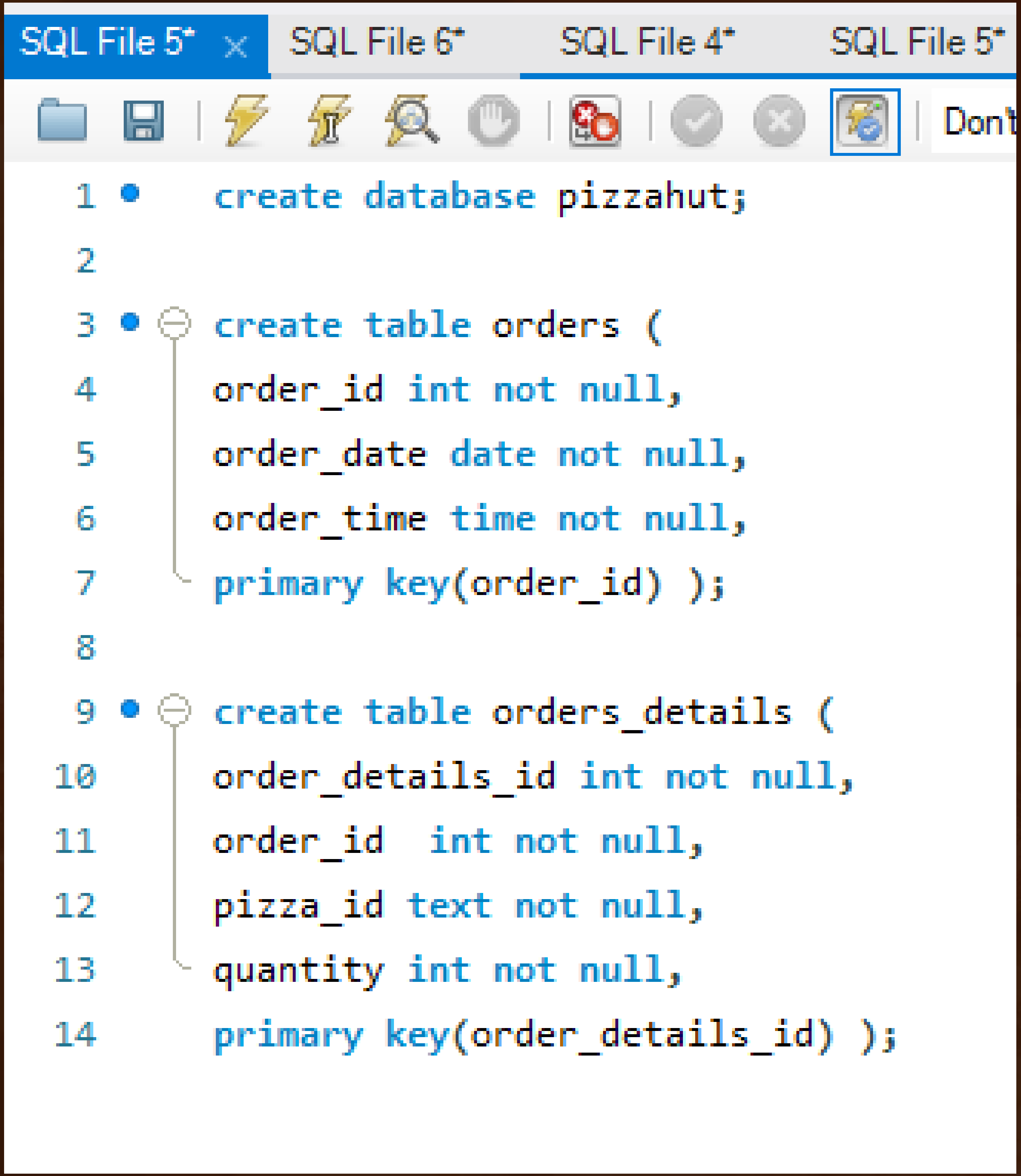
Pizza Sales analysis using SQL query in MySQL

DESCRIPTION:--

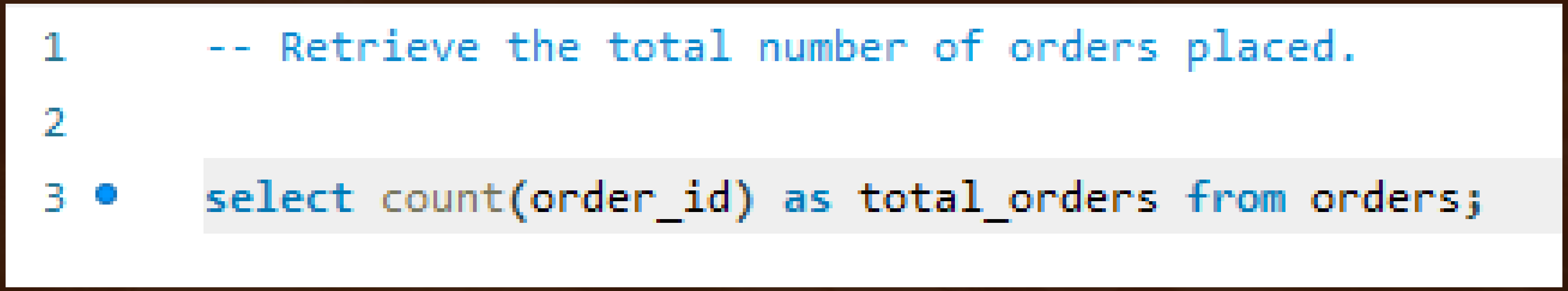
This project focuses on analyzing pizza sales data using SQL queries within the MySQL database. I utilized SQL to:
→ Identify total revenue generated from pizza and quantity sold, identify top selling pizzas, their size and category using various functions and group by.
→Obtain distribution of orders placed by hour of the day, average number of pizza orders per day using table joins and sub-queries.
→ Identify top selling pizza based on revenue for each pizza category and percentage contribution on revenue by each category of pizza using some calculations and functions.
→ Also, calculate the cummulative revenue generated over time using the window functions.

PROBLEM STATEMENT:--

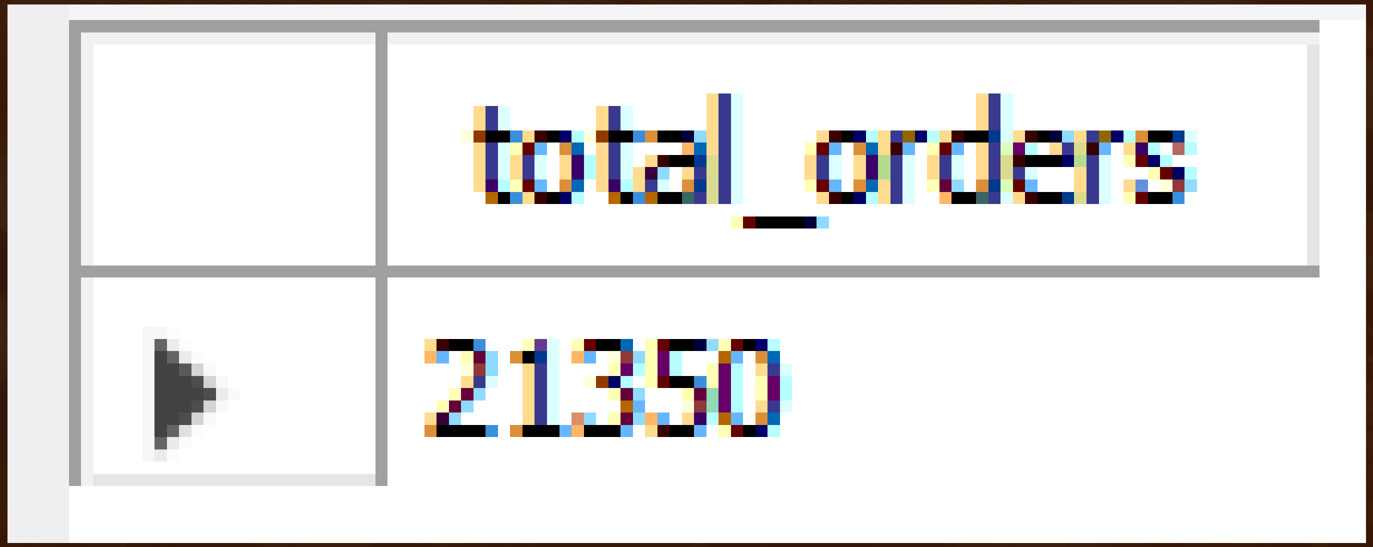
1. Retrieve the total number of orders placed.
2. Calculate the total revenue generated from pizza sales.
3. Identify the highest-priced pizza.
4. Identify the most common pizza size ordered.
5. List the top 5 most ordered pizza types along with their quantities.
6. Join the necessary tables to find the total quantity of each pizza category ordered .
7. Determine the distribution of orders by hour of the day.
8. Join relevant tables to find the category-wise distribution of pizzas.
9. Group the orders by date and calculate the average number of pizzas ordered per day.
10. Determine the top 3 most ordered pizza types based on revenue.
11. Calculate the percentage contribution of each pizza category to total revenue.
12. Analyze the cumulative revenue generated over time.
13. Determine the top 3 most ordered pizza types based on revenue for each pizza category.



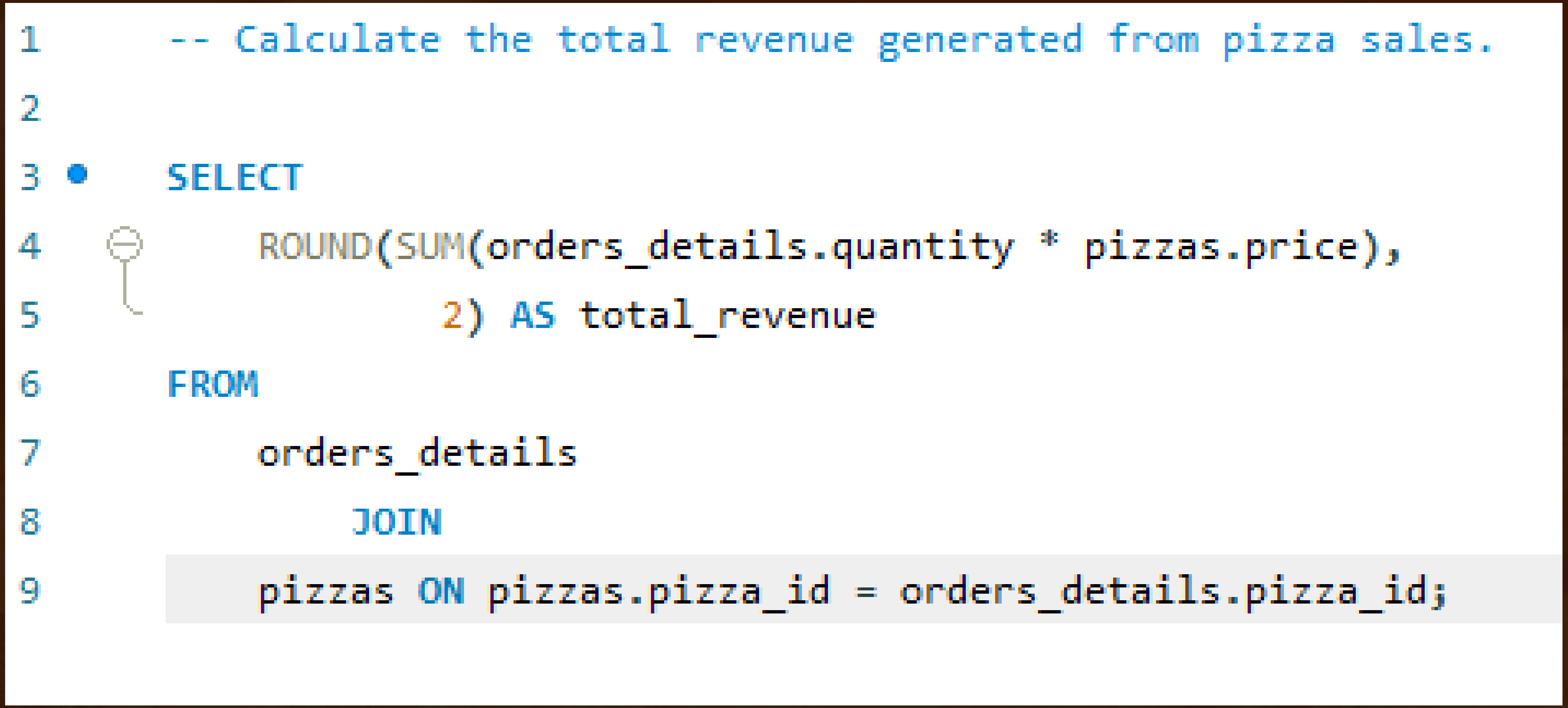
```sql
create database pizzahut;

create table orders (
order_id int not null,
order_date date not null,
order_time time not null,
primary key(order_id) );

create table orders_details (
order_details_id int not null,
order_id  int not null,
pizza_id text not null,
quantity int not null,
primary key(order_details_id) );
```



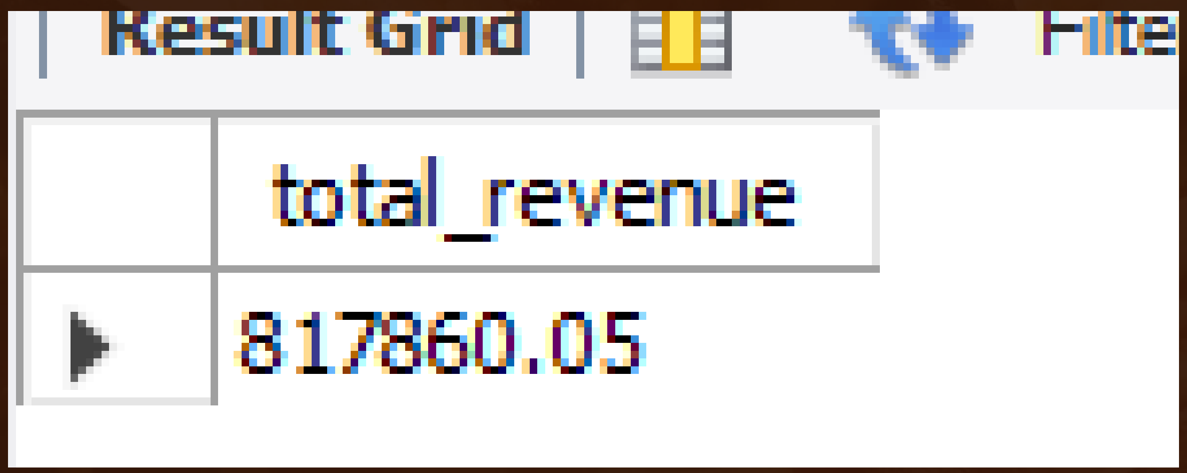
```sql
-- Retrieve the total number of orders placed.

select count(order_id) as total_orders from orders;
```



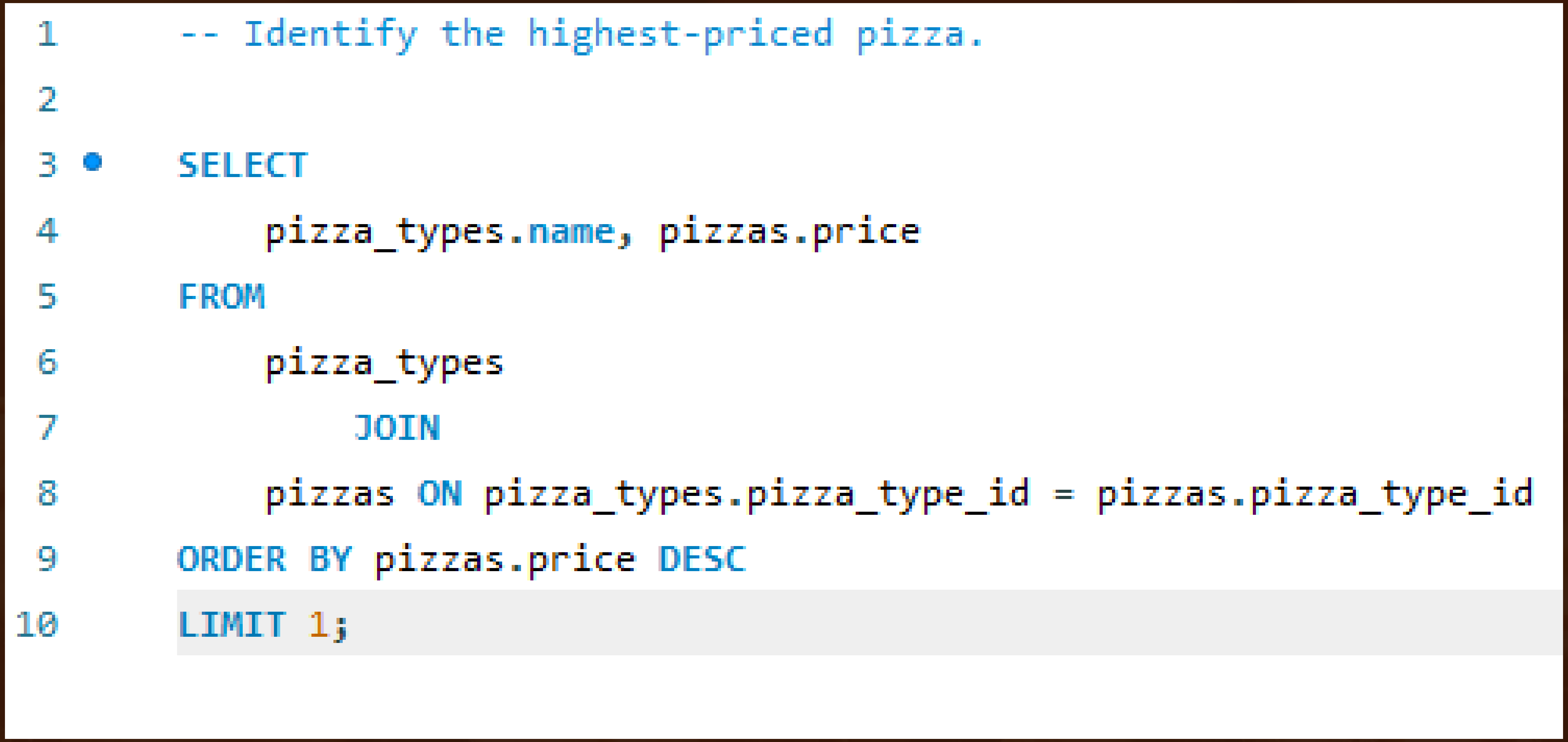
	total_orders
▶	21350



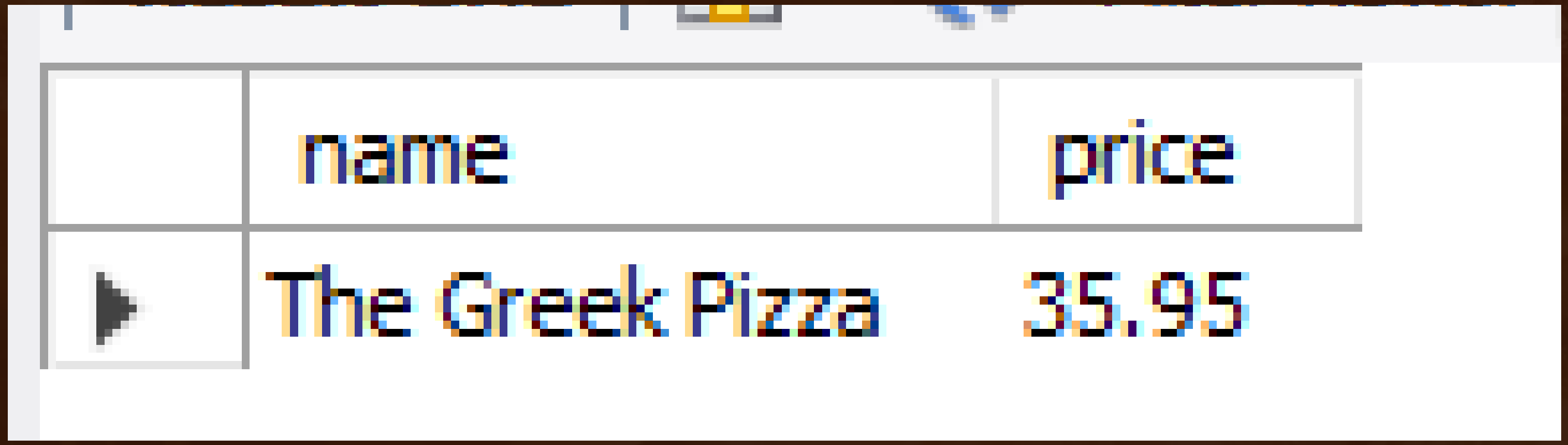
```sql
-- Calculate the total revenue generated from pizza sales.

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



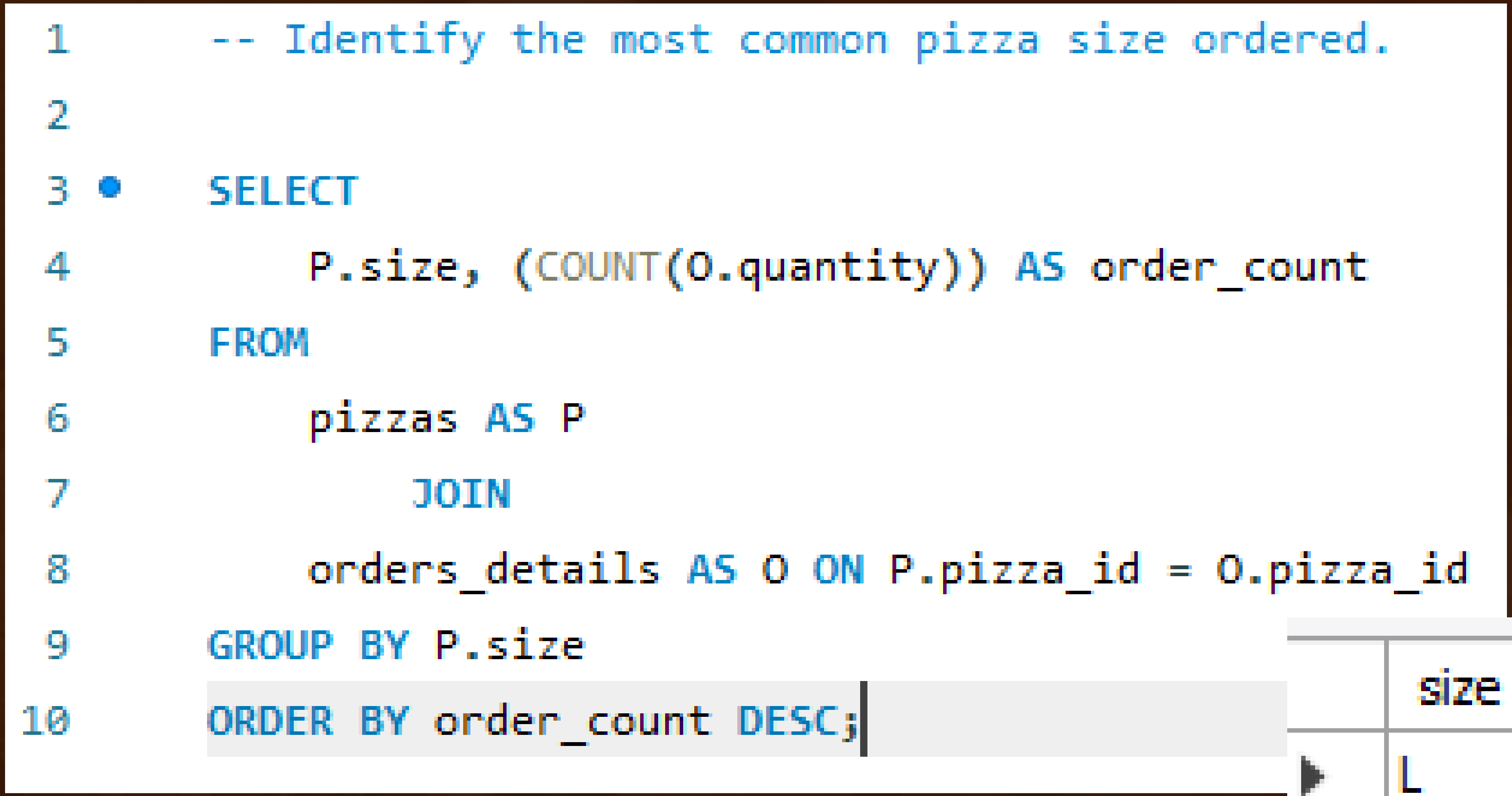
```
total_revenue
817860.05
```



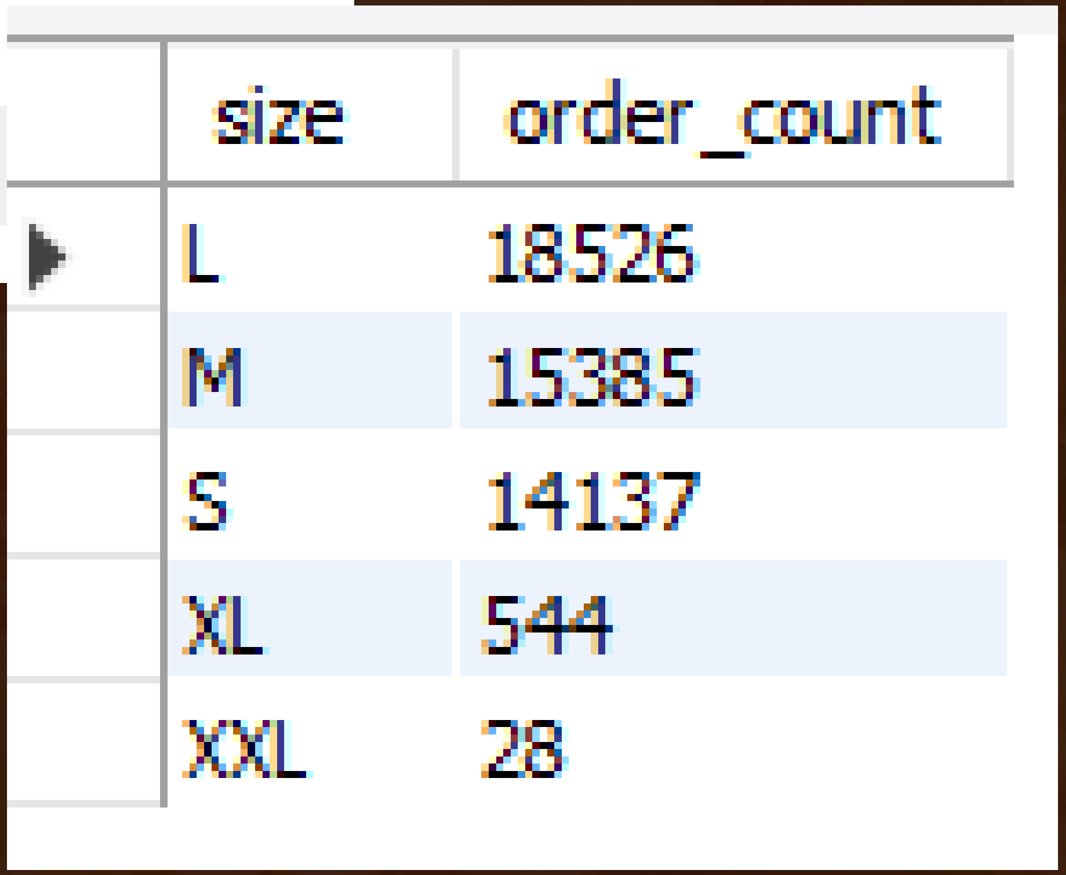
```sql
-- Identify the highest-priced pizza.

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



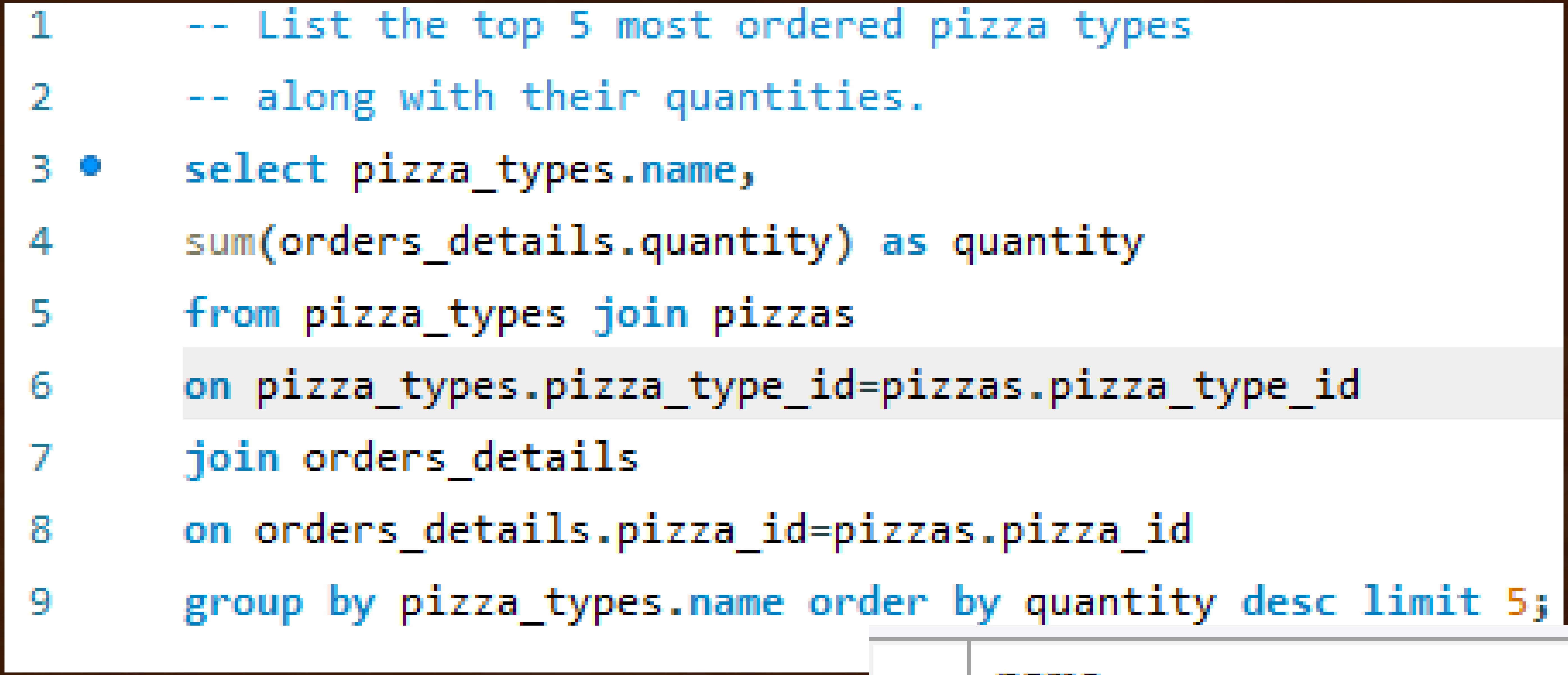
```
name              price
The Greek Pizza   35.95
```



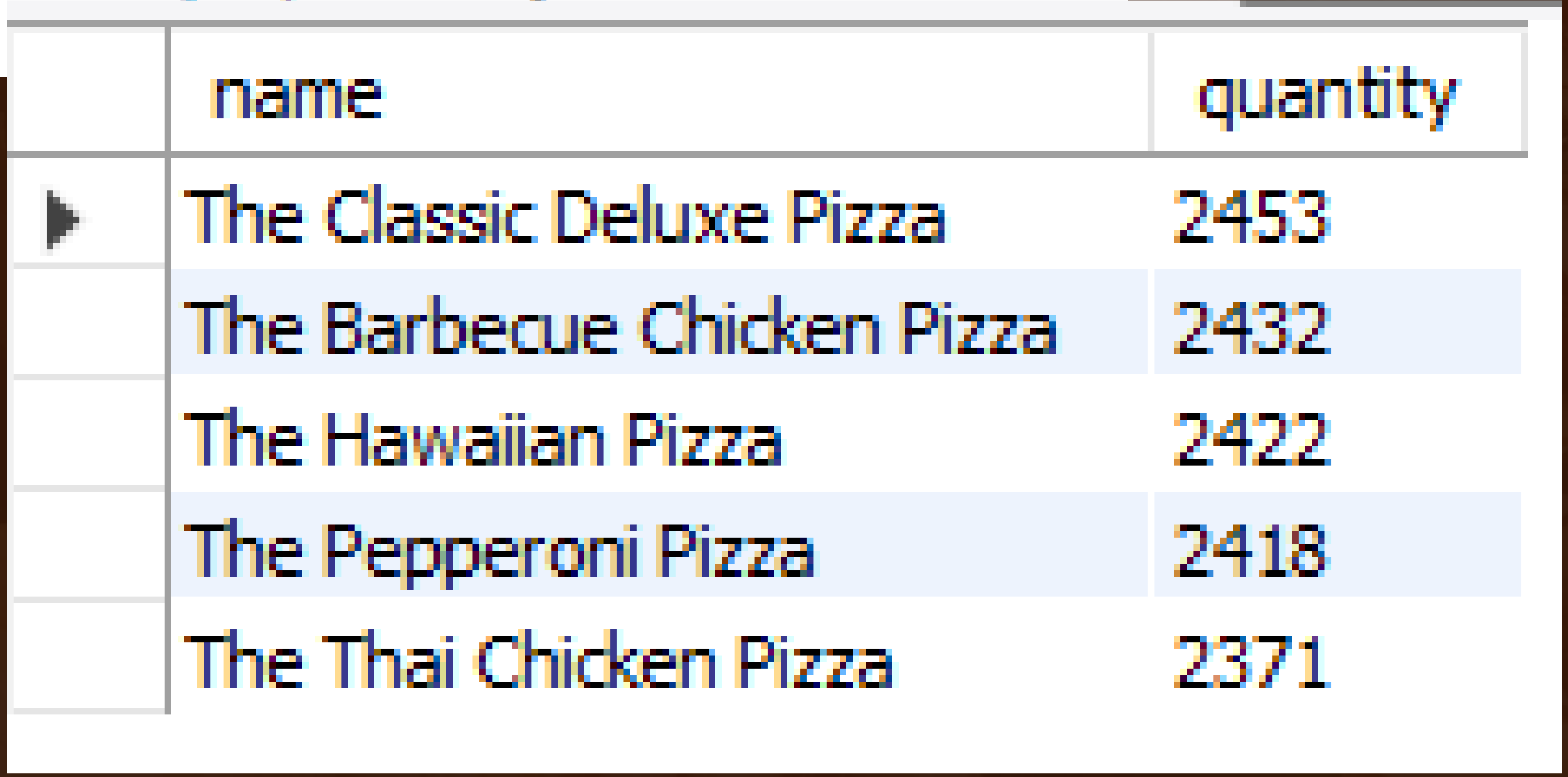
```sql
-- Identify the most common pizza size ordered.

SELECT
    P.size, (COUNT(O.quantity)) AS order_count
FROM
    pizzas AS P
        JOIN
    orders_details AS O ON P.pizza_id = O.pizza_id
GROUP BY P.size
ORDER BY order_count DESC;
```



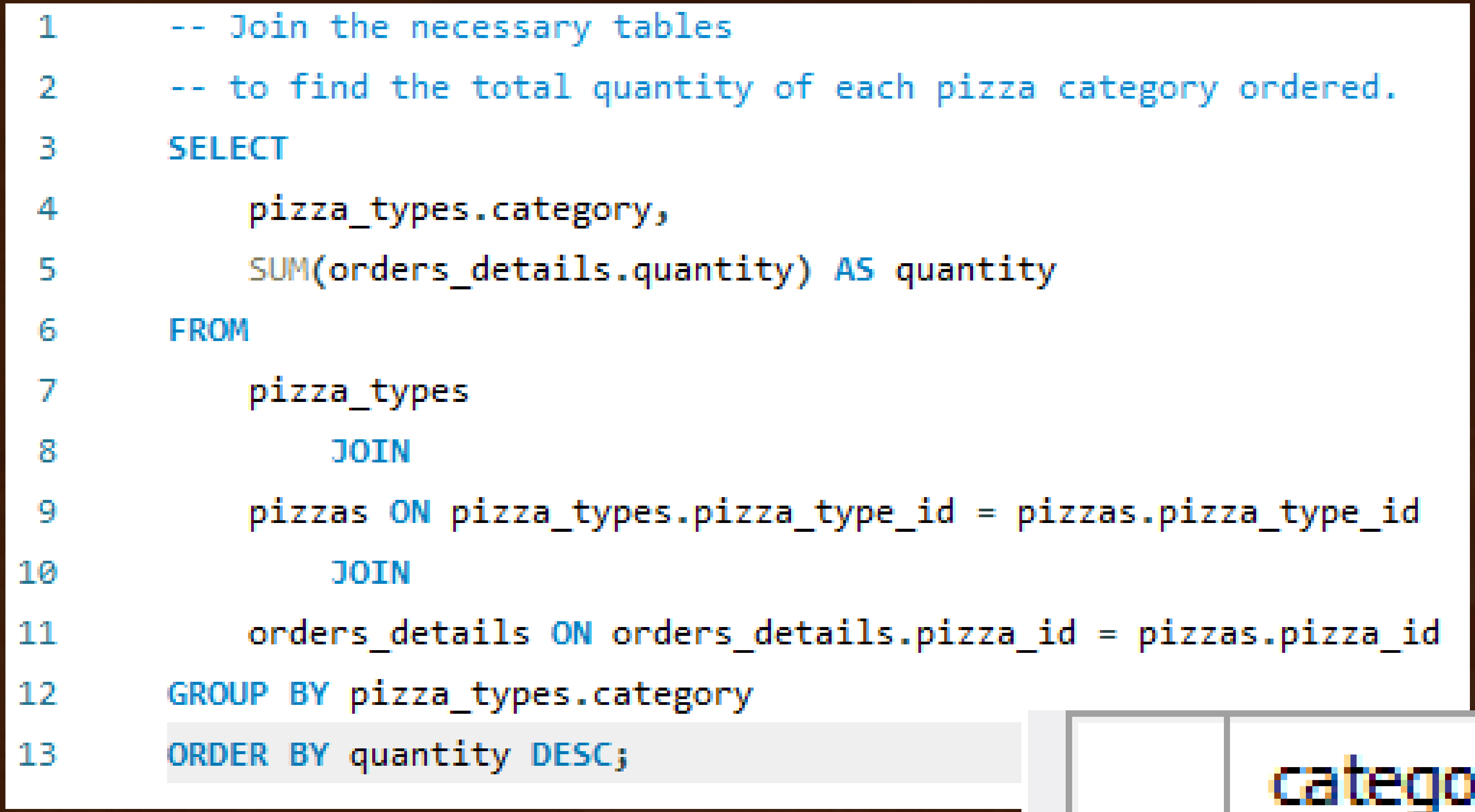
size	order_count
L	18526
M	15385
S	14137
XL	544
XXL	28



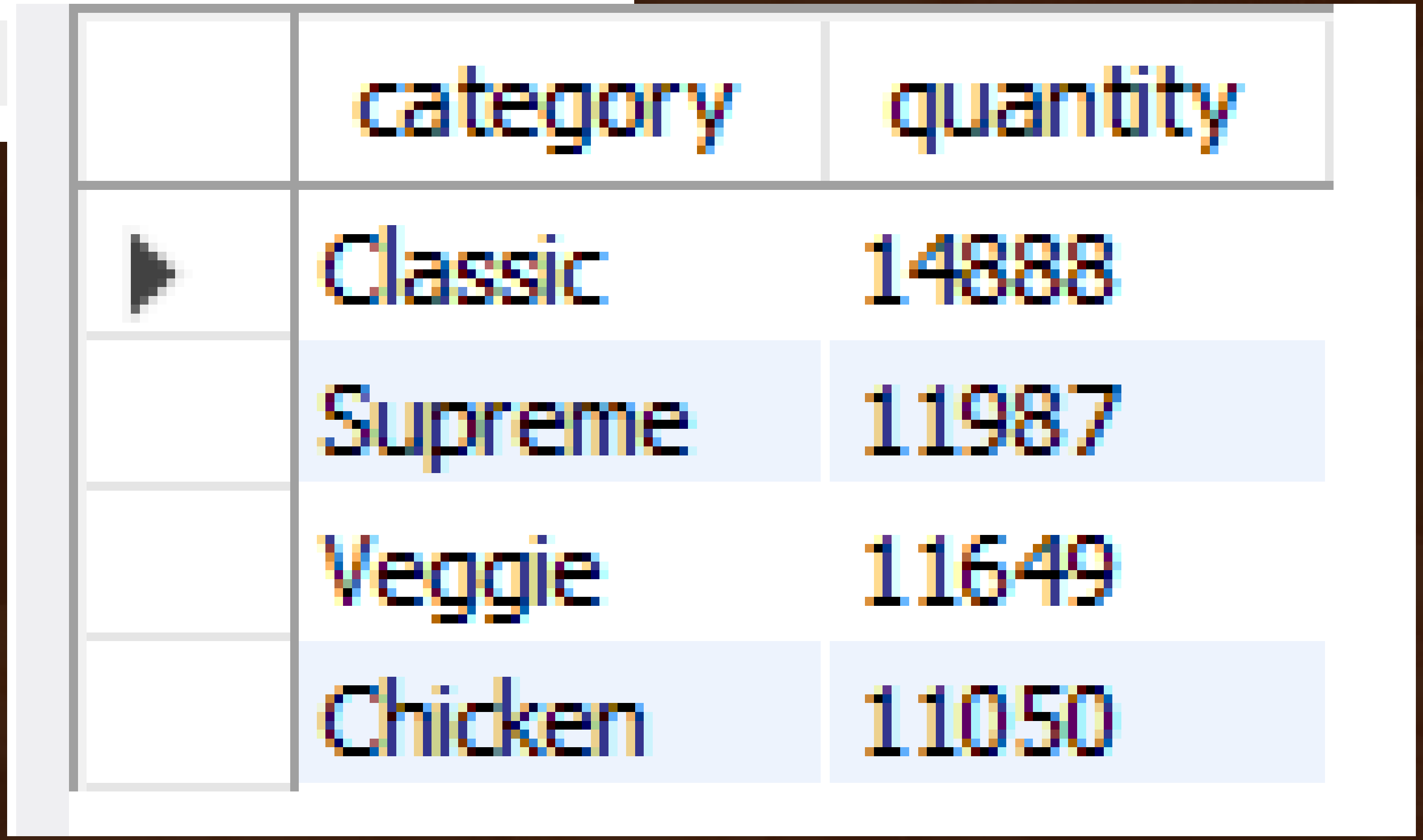
```sql
-- List the top 5 most ordered pizza types
-- along with their quantities.
select pizza_types.name,
sum(orders_details.quantity) as quantity
from pizza_types join pizzas
on pizza_types.pizza_type_id=pizzas.pizza_type_id
join orders_details
on orders_details.pizza_id=pizzas.pizza_id
group by pizza_types.name order by quantity desc limit 5;
```



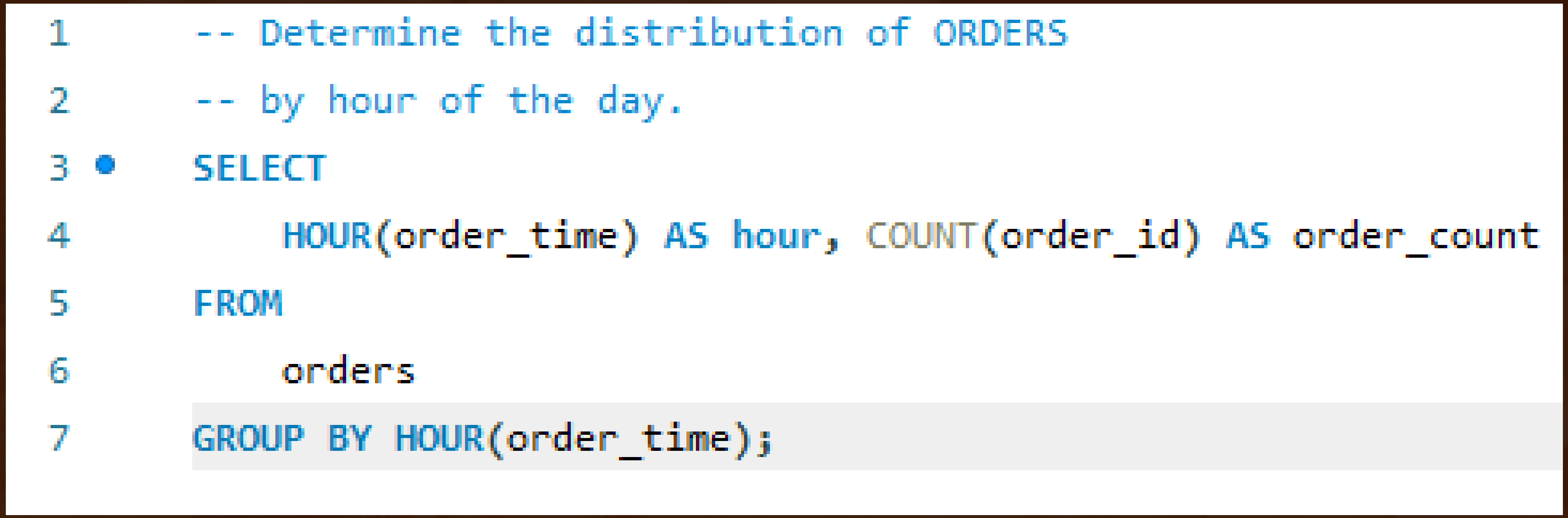
name	quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371



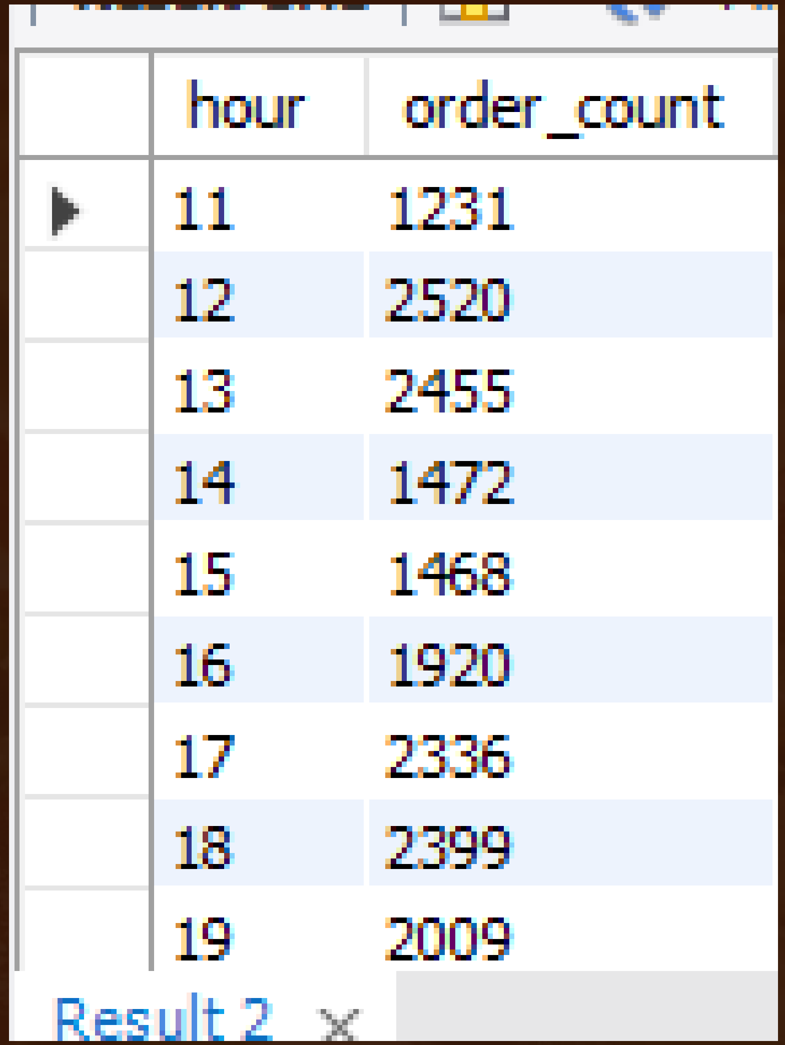
```sql
-- Join the necessary tables
-- to find the total quantity of each pizza category ordered.
SELECT
    pizza_types.category,
    SUM(orders_details.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity DESC;
```



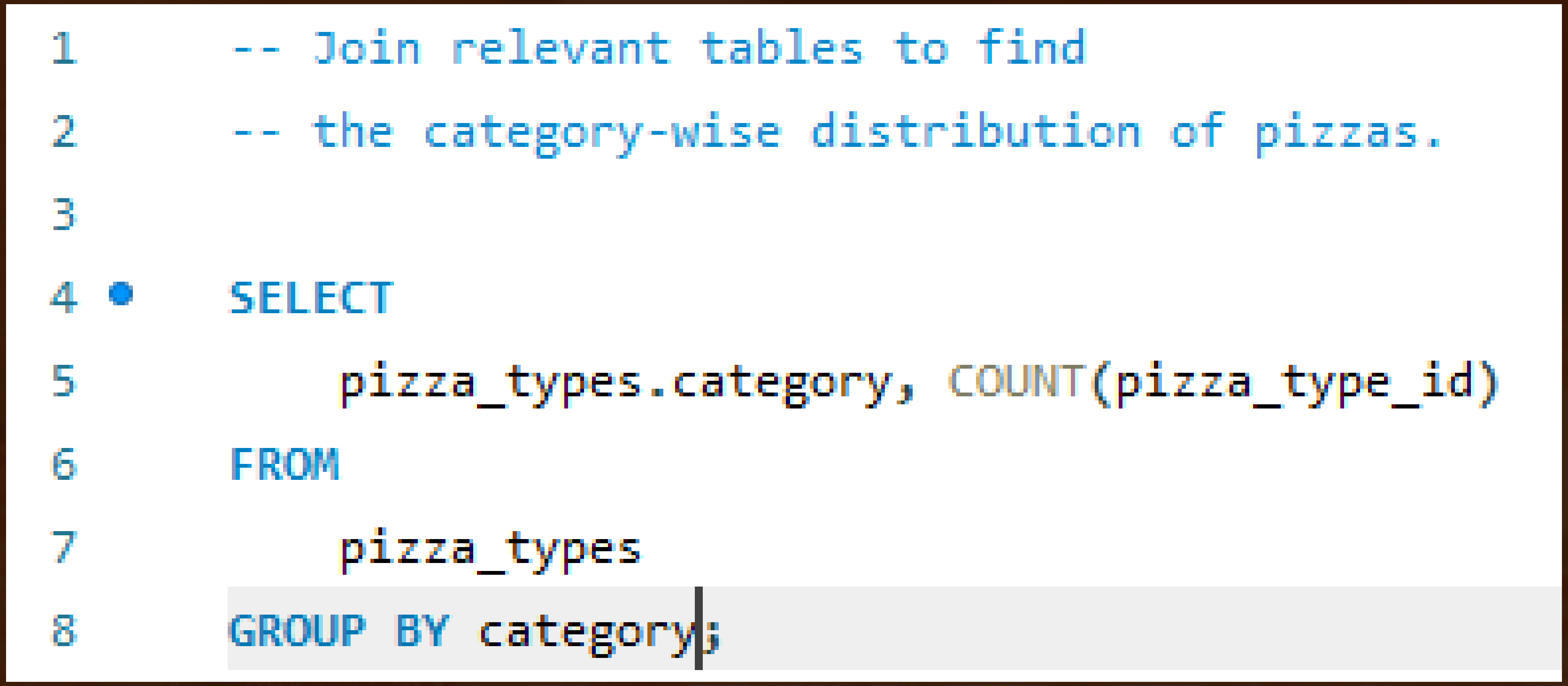
category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050



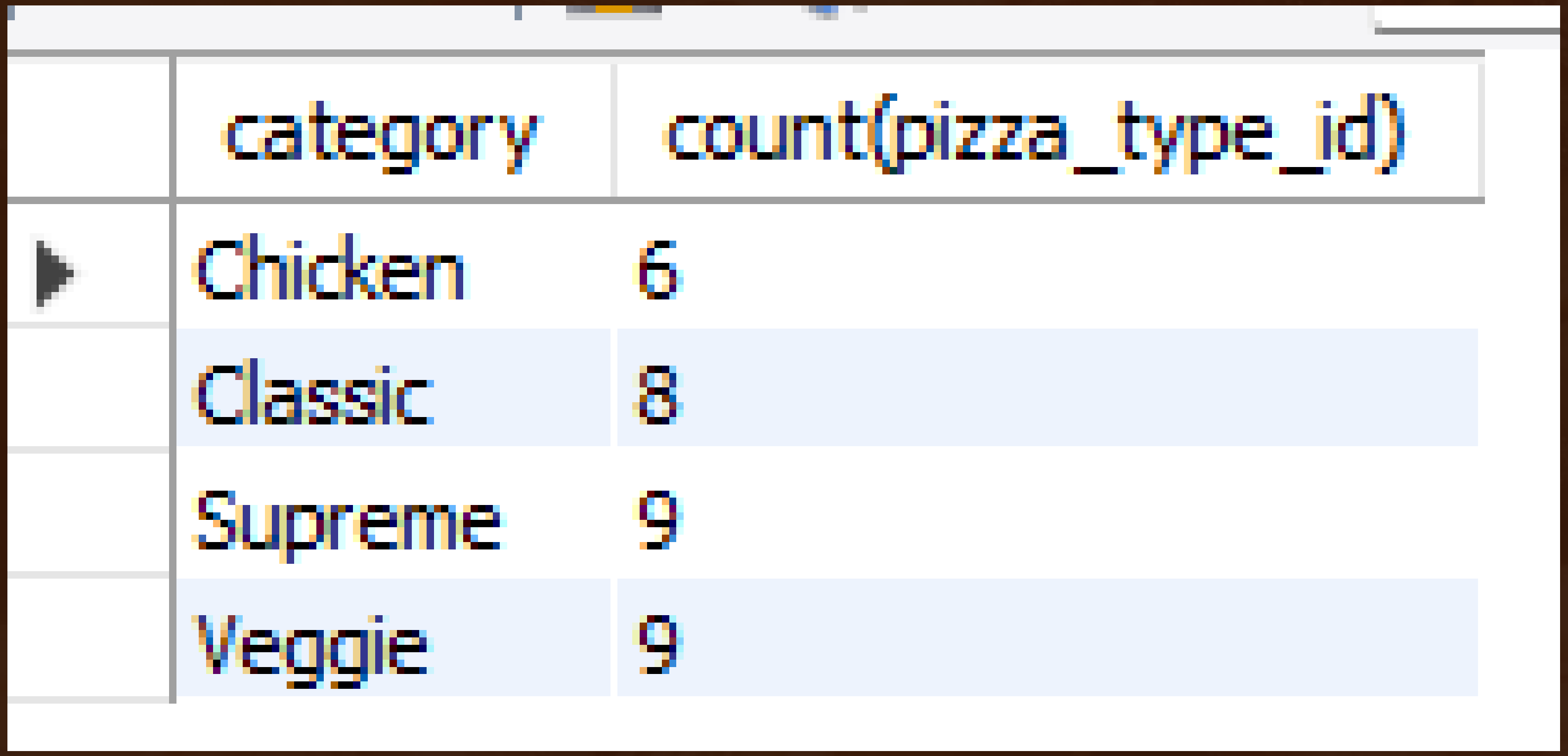
```sql
-- Determine the distribution of ORDERS
-- by hour of the day.
SELECT
    HOUR(order_time) AS hour, COUNT(order_id) AS order_count
FROM
    orders
GROUP BY HOUR(order_time);
```



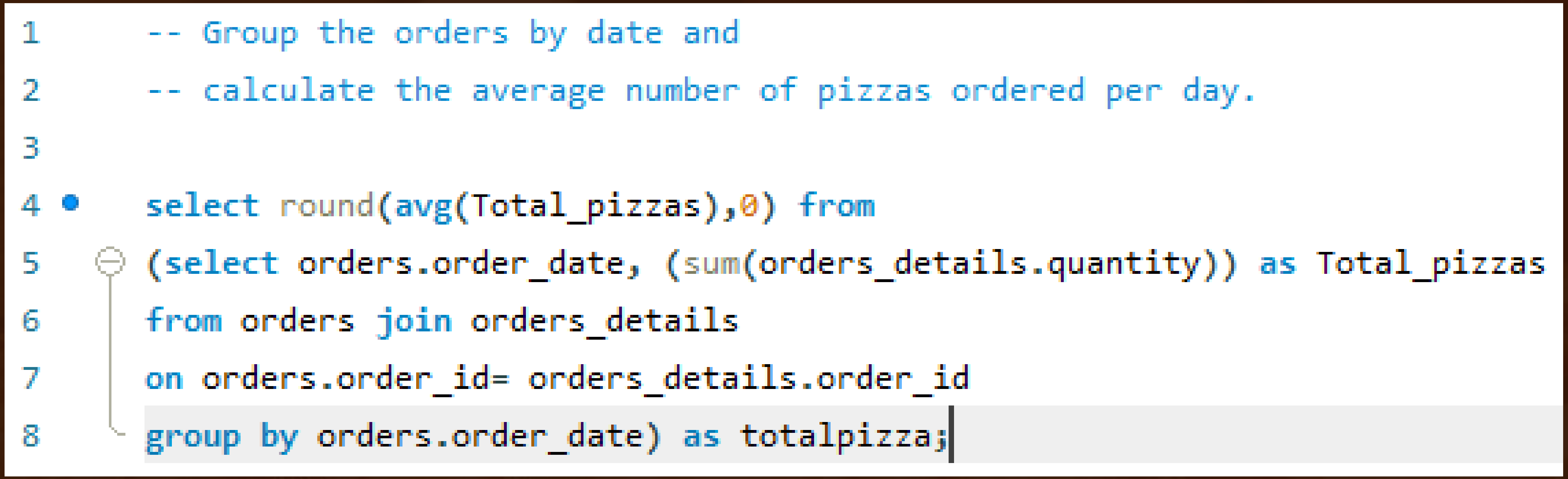
hour	order_count
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009



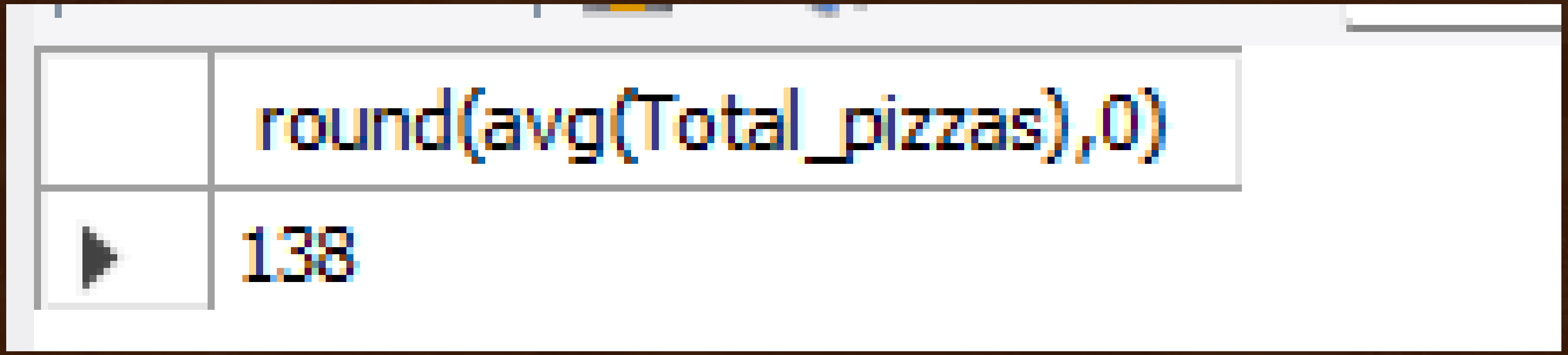
```sql
-- Join relevant tables to find
-- the category-wise distribution of pizzas.

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



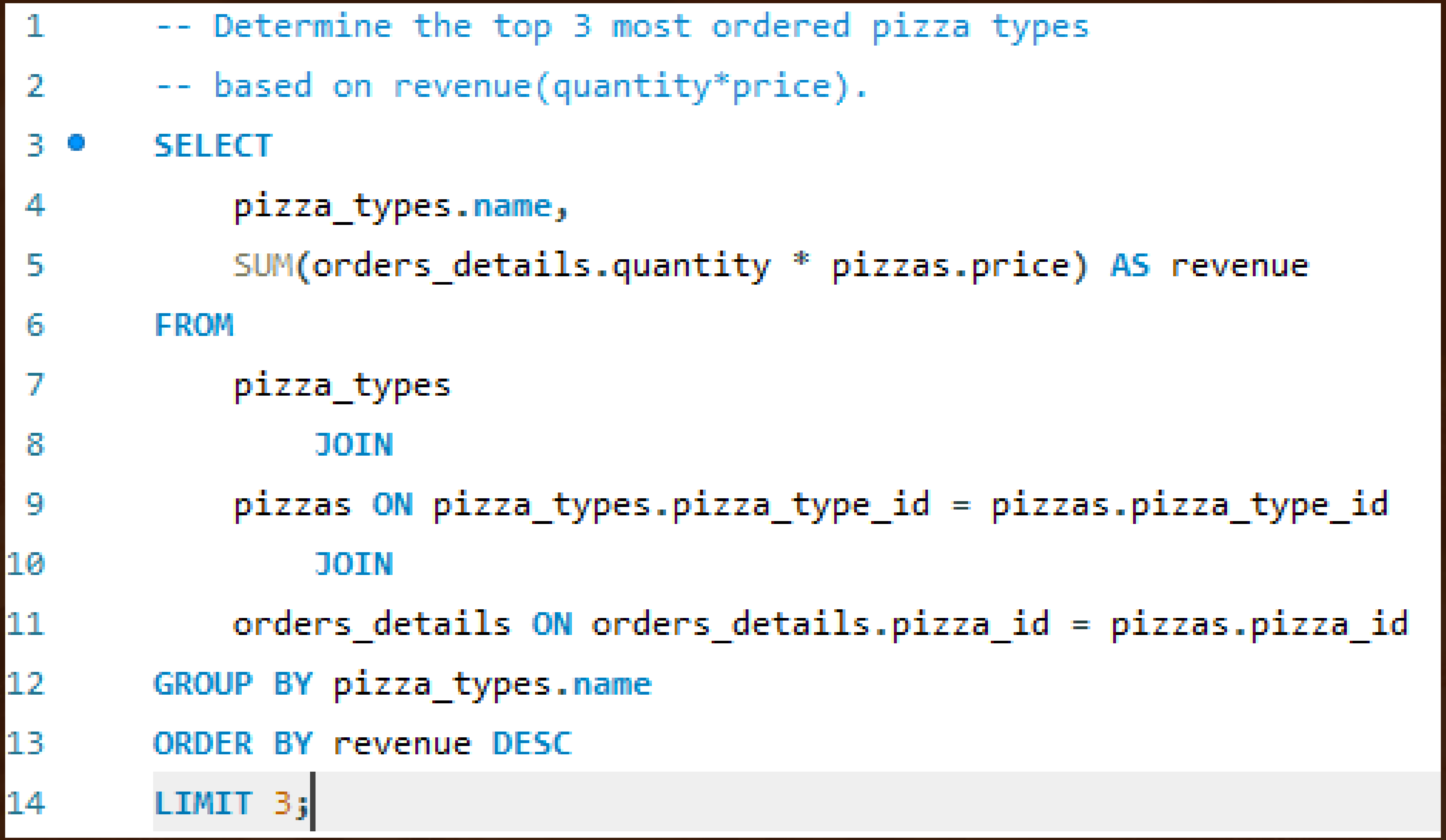
category	count(pizza_type_id)
Chicken	6
Classic	8
Supreme	9
Veggie	9



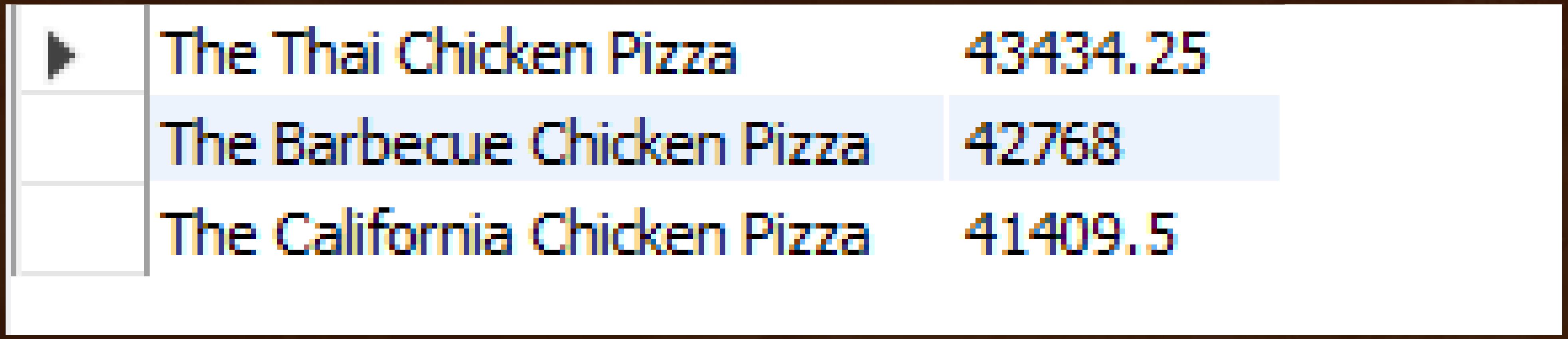
```sql
-- Group the orders by date and
-- calculate the average number of pizzas ordered per day.

select round(avg(Total_pizzas),0) from
(select orders.order_date, (sum(orders_details.quantity)) as Total_pizzas
from orders join orders_details
on orders.order_id= orders_details.order_id
group by orders.order_date) as totalpizza;
```



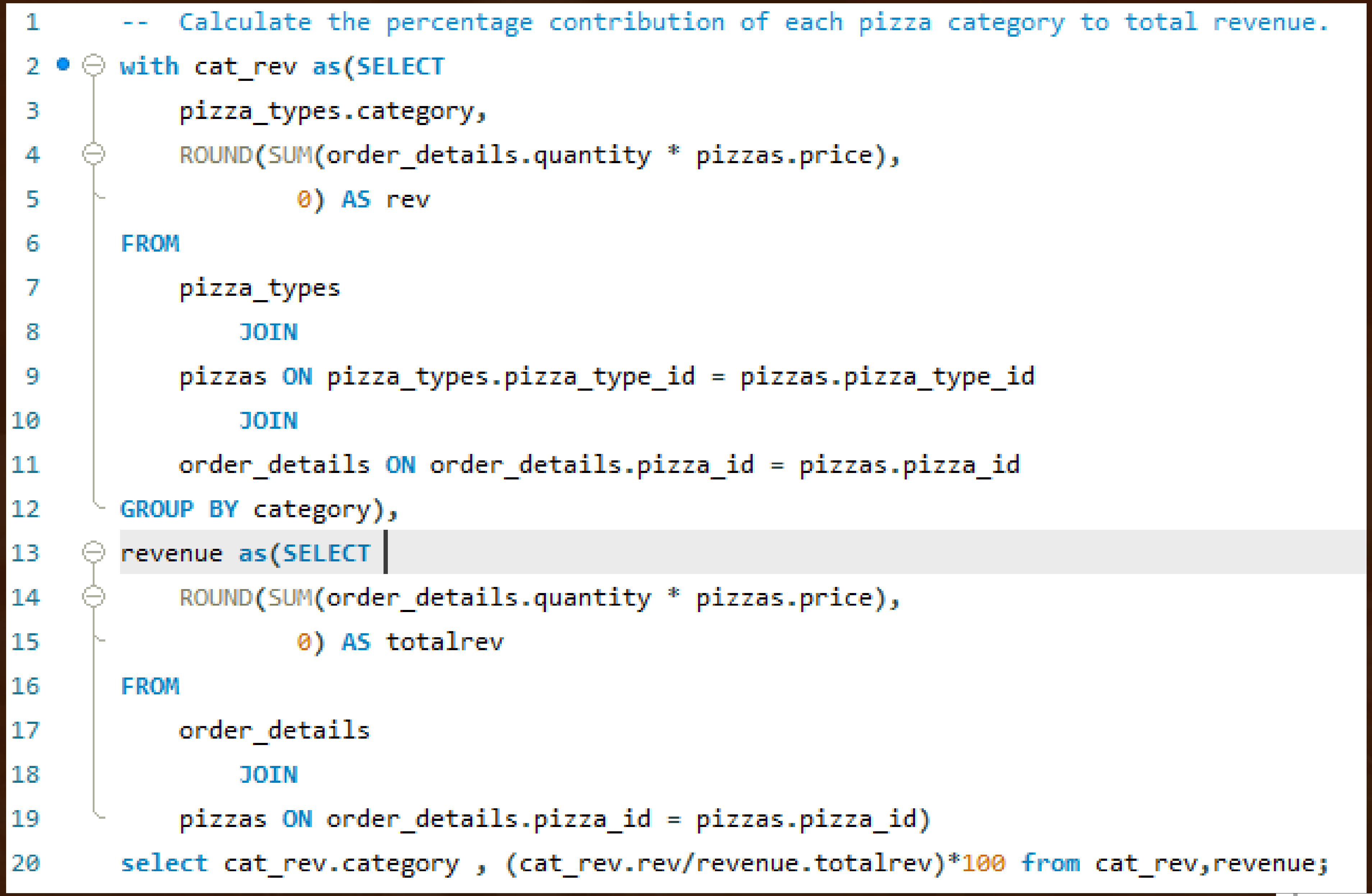
```
round(avg(Total_pizzas),0)
138
```



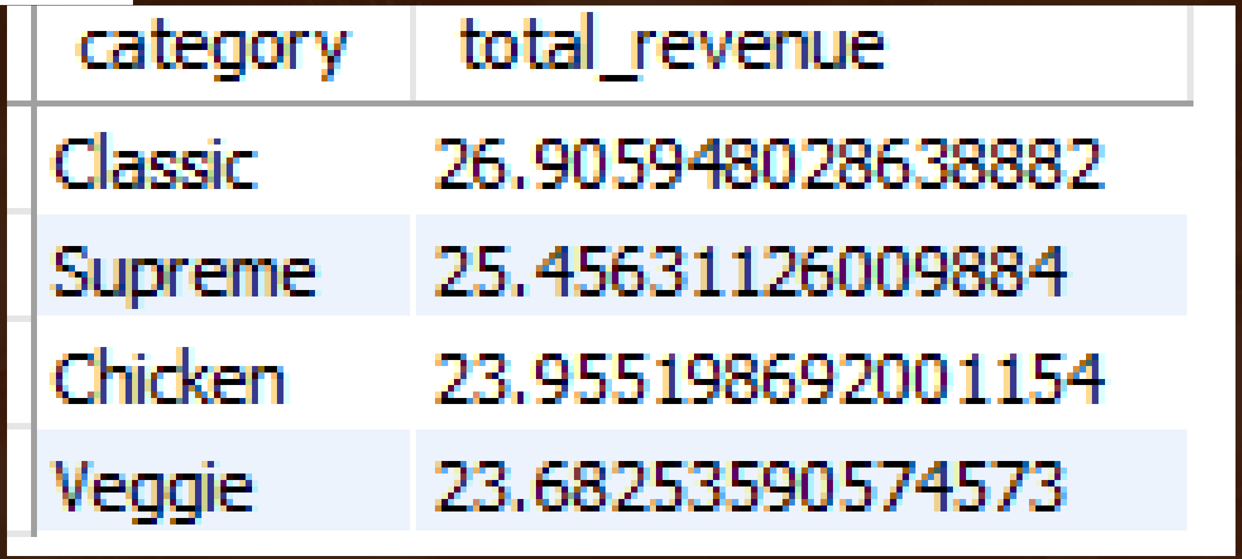
```sql
-- Determine the top 3 most ordered pizza types
-- based on revenue(quantity*price).
SELECT
    pizza_types.name,
    SUM(orders_details.quantity * pizzas.price) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```



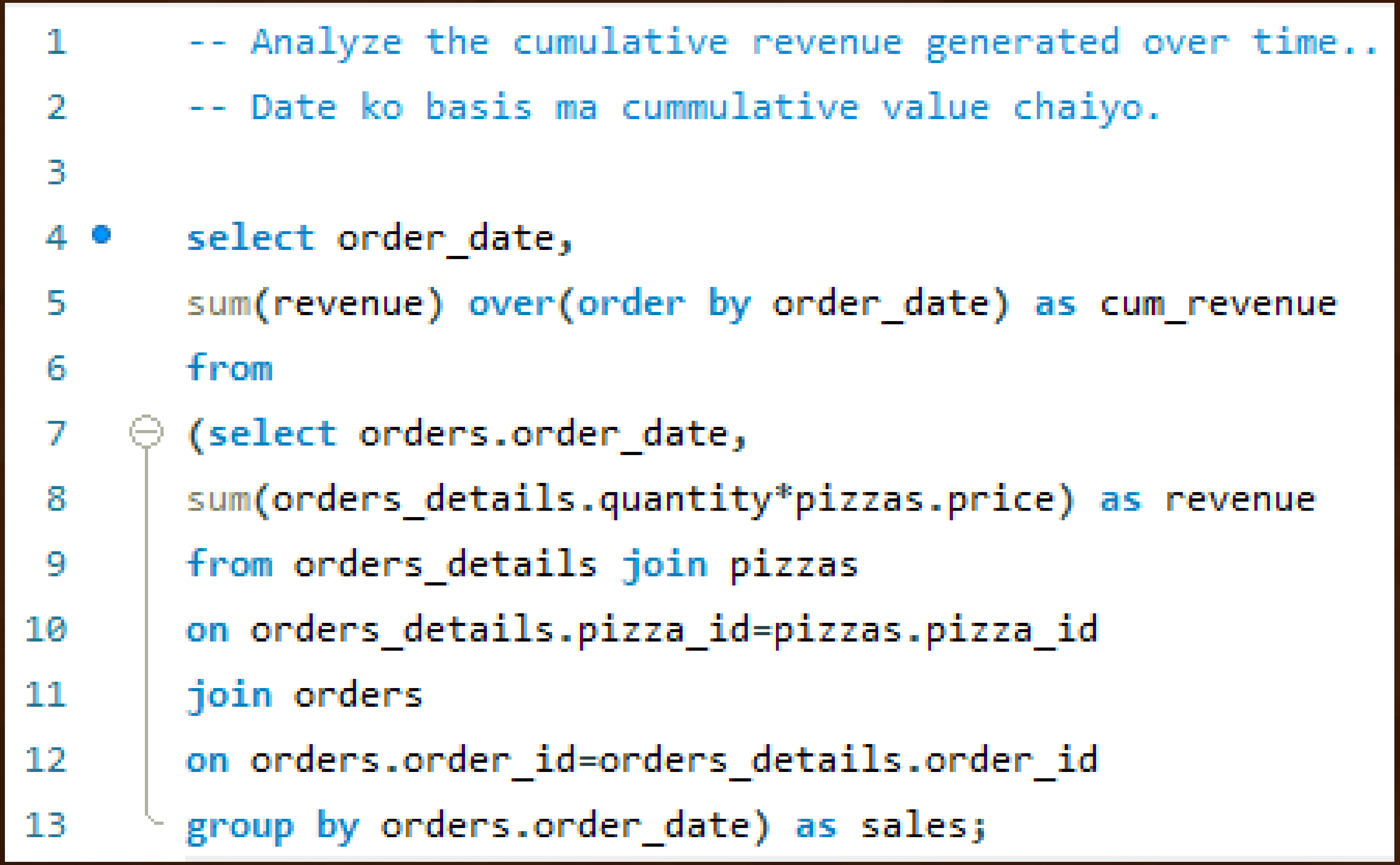
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5



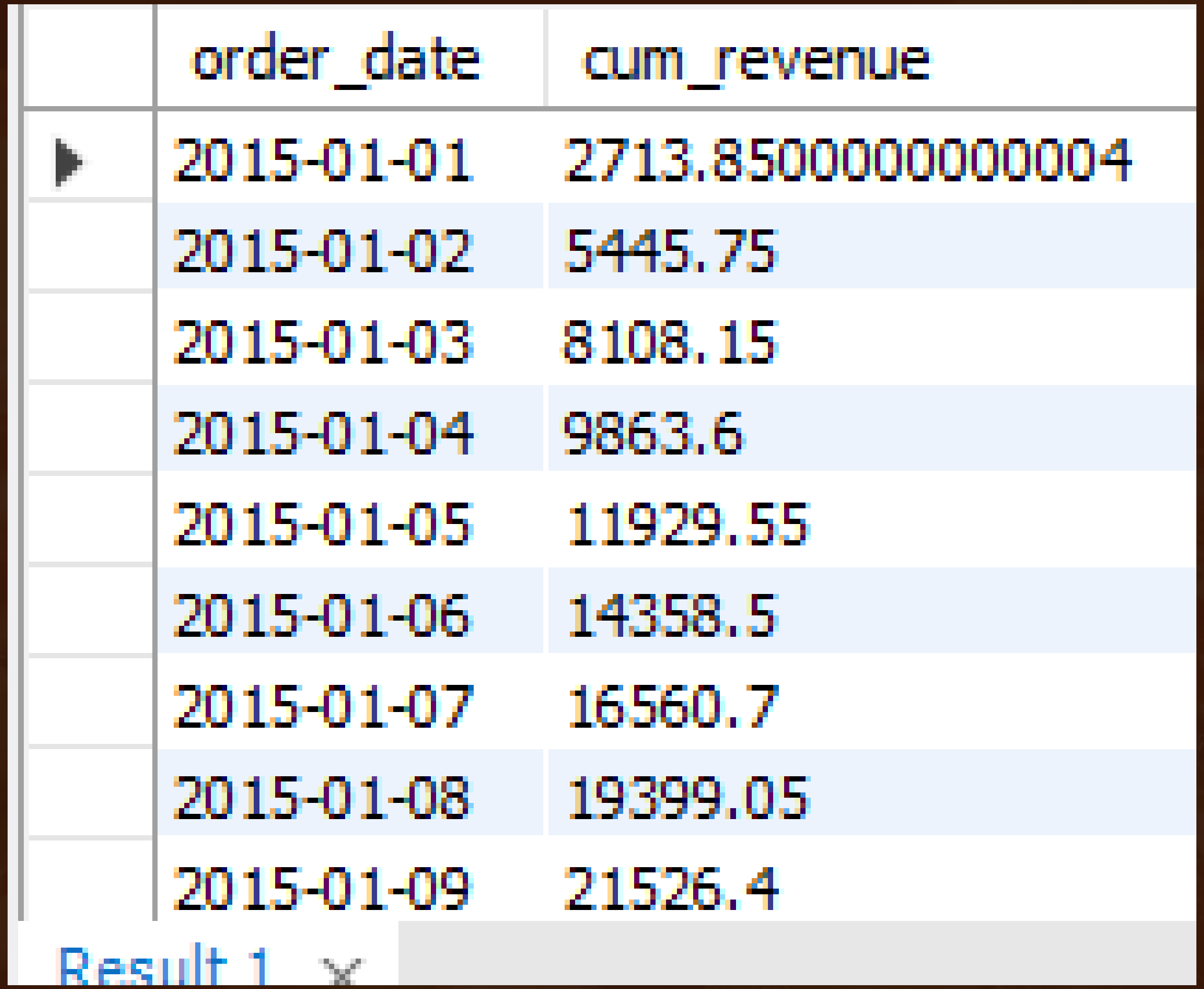
```
-- Calculate the percentage contribution of each pizza category to total revenue.
with cat_rev as(SELECT
    pizza_types.category,
    ROUND(SUM(order_details.quantity * pizzas.price),
            0) AS rev
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY category),
revenue as(SELECT
    ROUND(SUM(order_details.quantity * pizzas.price),
            0) AS totalrev
FROM
    order_details
        JOIN
    pizzas ON order_details.pizza_id = pizzas.pizza_id)
select cat_rev.category , (cat_rev.rev/revenue.totalrev)*100 from cat_rev,revenue;
```



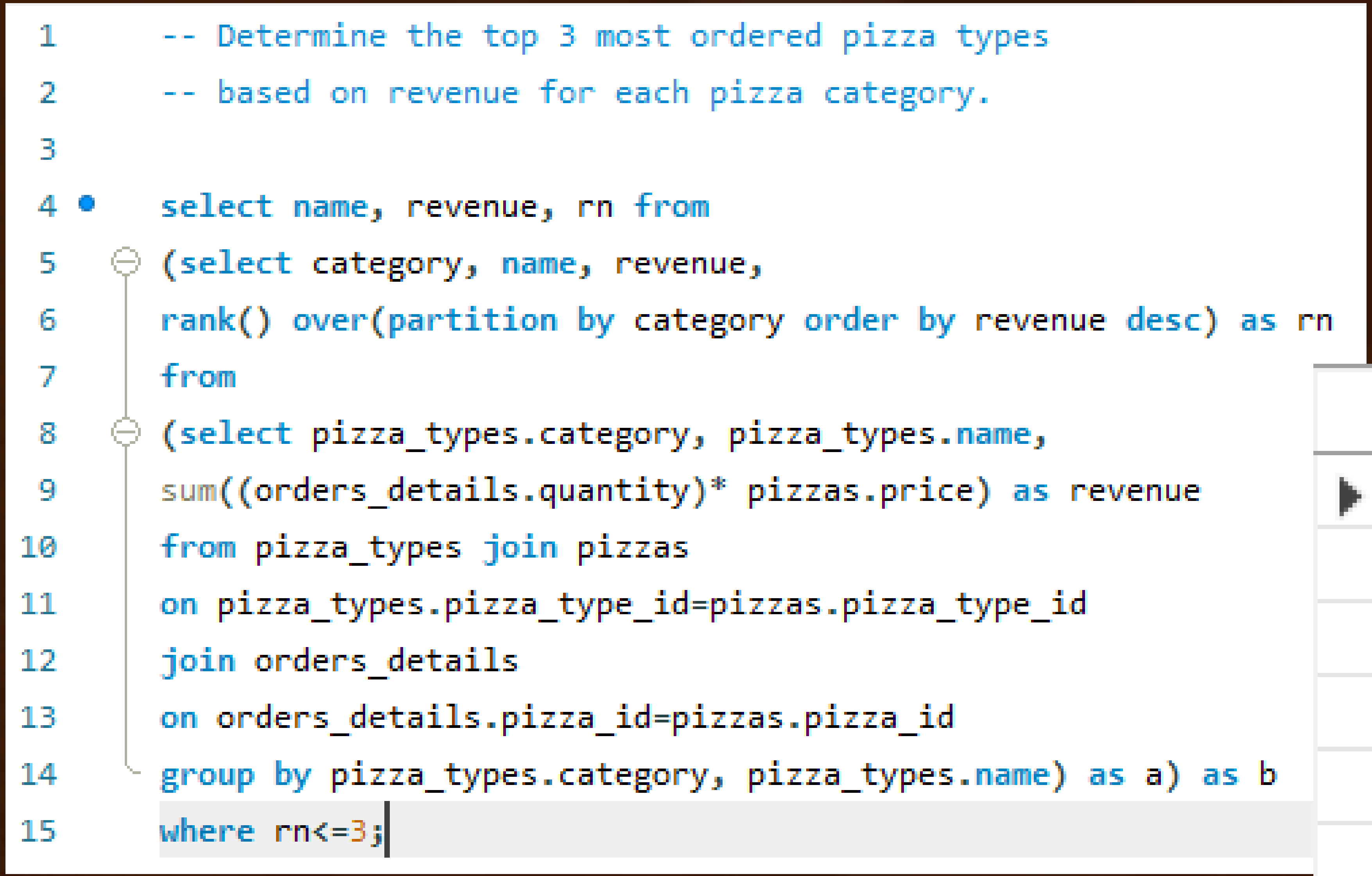
category	total_revenue
Classic	26.905948028638882
Supreme	25.456311260098840
Chicken	23.955198692001154
Veggie	23.682535905745730



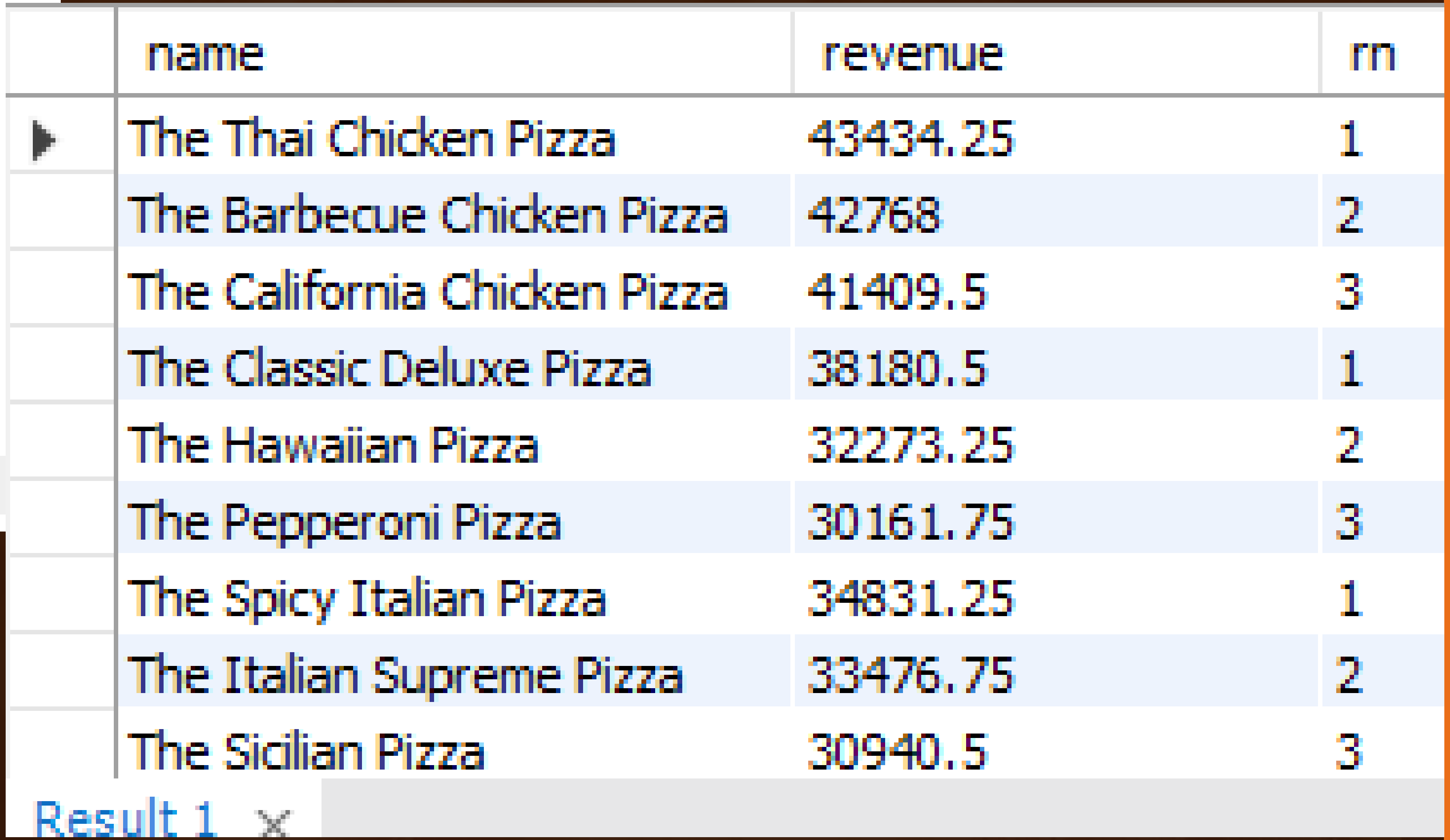
```
-- Analyze the cumulative revenue generated over time..
-- Date ko basis ma cummulative value chaiyo.

select order_date,
sum(revenue) over(order by order_date) as cum_revenue
from
(select orders.order_date,
sum(orders_details.quantity*pizzas.price) as revenue
from orders_details join pizzas
on orders_details.pizza_id=pizzas.pizza_id
join orders
on orders.order_id=orders_details.order_id
group by orders.order_date) as sales;
```



order_date	cum_revenue
2015-01-01	2713.8500000000004
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



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

select name, revenue, rn from
(select category, name, revenue,
rank() over(partition by category order by revenue desc) as rn
from
(select pizza_types.category, pizza_types.name,
sum((orders_details.quantity)* pizzas.price) as revenue
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, pizza_types.name) as a) as b
where rn<=3;
```



name	revenue	rn
The Thai Chicken Pizza	43434.25	1
The Barbecue Chicken Pizza	42768	2
The California Chicken Pizza	41409.5	3
The Classic Deluxe Pizza	38180.5	1
The Hawaiian Pizza	32273.25	2
The Pepperoni Pizza	30161.75	3
The Spicy Italian Pizza	34831.25	1
The Italian Supreme Pizza	33476.75	2
The Sicilian Pizza	30940.5	3

Result 1 ×

THANK YOU FOR ATTENTION

www.linkedin.com/in/ajay-narayan-poddar

AJAY N. PODDAR