Home About Contact

SQL EDA PROJECT – PIZZA SALES ANALYSIS

Home About Contact

ABOUT THIS PROJECT

This project demonstrates how SQL can be used for Exploratory Data Analysis (EDA) on a real-world pizza sales dataset. It involves designing a relational database, importing raw CSV data, writing analytical queries, and deriving insights on sales performance, product demand, and customer ordering trends.

SCHEMA

1. orders

- order_id (Primary Key): A unique ID for each order.
- date: The date the order was placed.
- time: The time the order was placed.

2. pizza_types

- pizza_type_id (Primary Key): A unique ID for each pizza type (e.g., bbq_ckn).
- name: The display name of the pizza.
- category: The category of the pizza (e.g., Chicken, Classic, Veggie).
- ingredients: The list of ingredients for that pizza type.

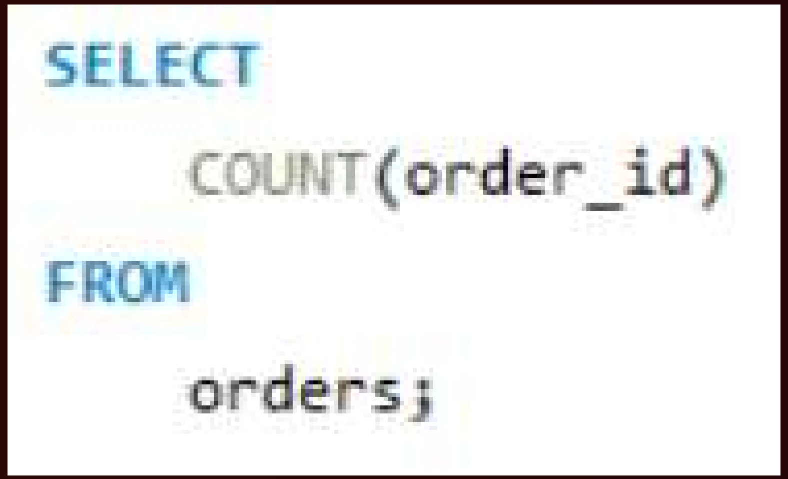
3. pizzas

- pizza_id (Primary Key): A unique ID for each specific pizza product (e.g., bbq_ckn_s).
- pizza_type_id (Foreign Key): This links to the pizza_types table to identify what kind of pizza it is.
- size: The size of the pizza (e.g., S, M, L).
- price: The price for this specific pizza size.

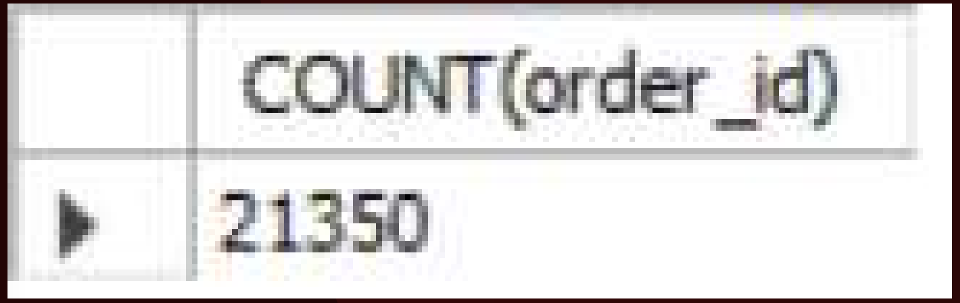
4. order_details

- order_details_id (Primary Key): A unique ID for each line item in an order.
- order_id (Foreign Key): This links to the orders table to show which order this item belongs to.
- pizza_id (Foreign Key): This links to the pizzas table to show which specific pizza (and size) was ordered.
- quantity: How many of this specific pizza were ordered.

1. RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

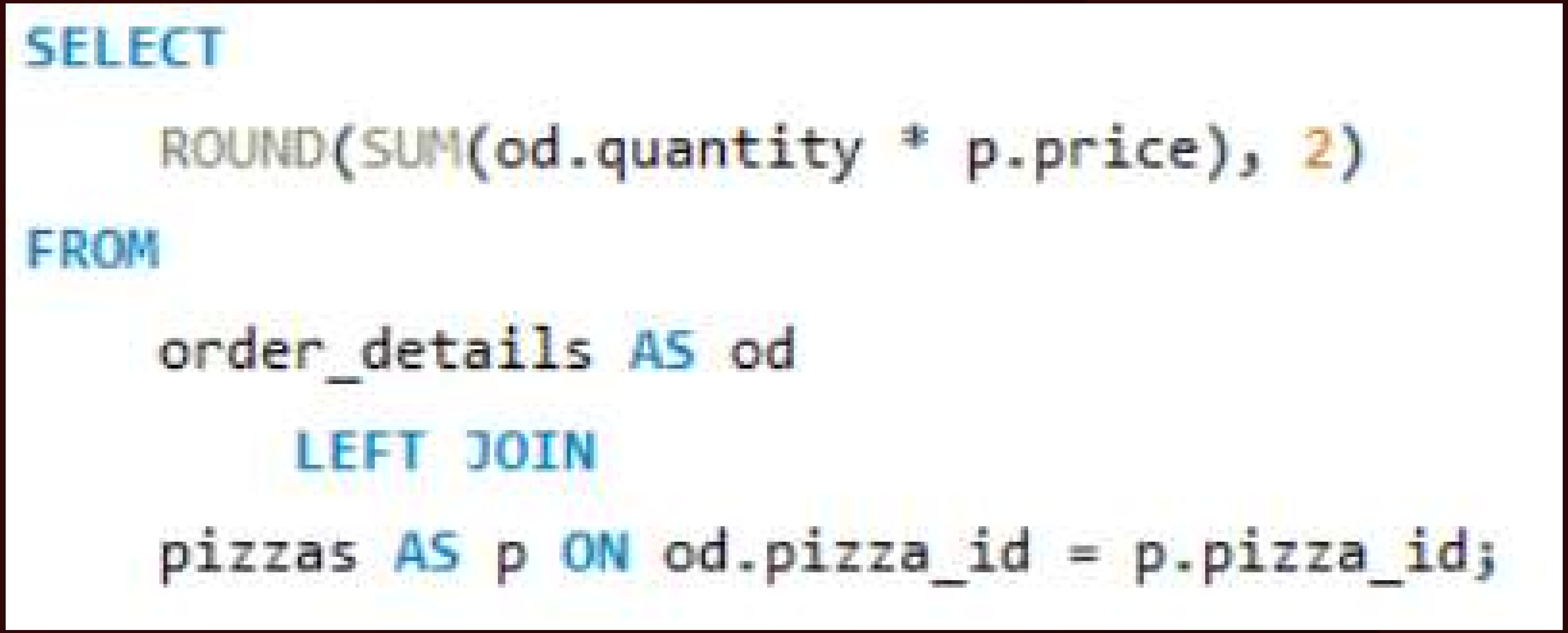


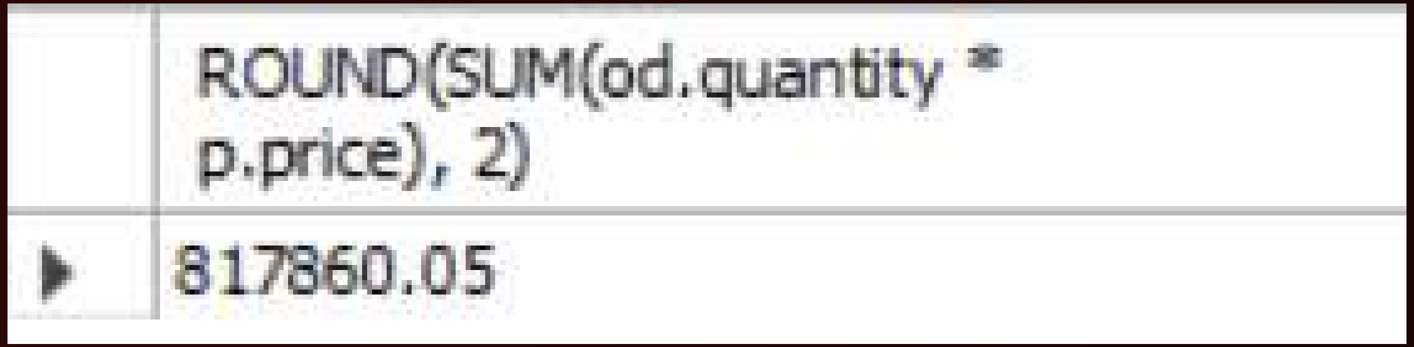
```
SELECT
    COUNT(order_id)
FROM
    orders;
```



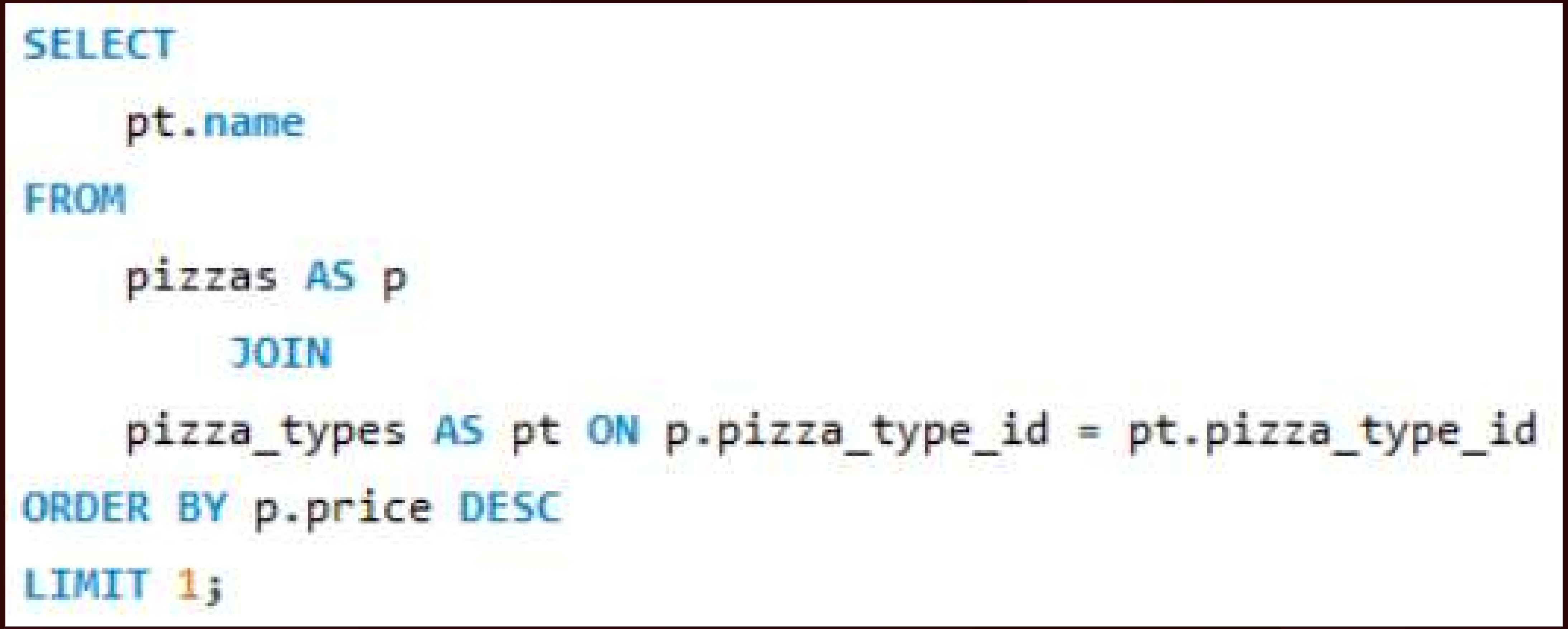
```
  | COUNT(order_id)
▶ | 21350
```

2. CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

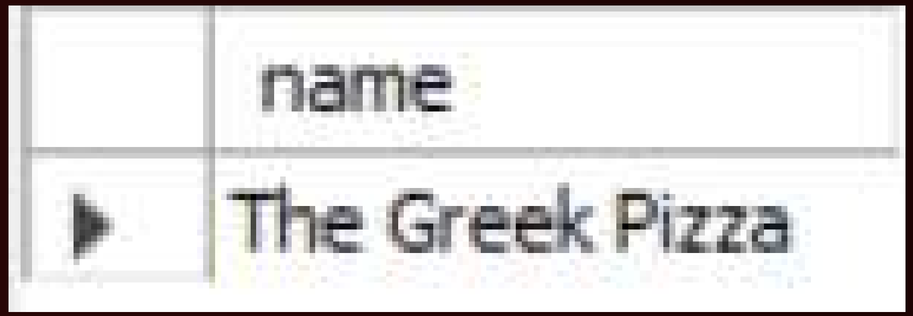

```
SELECT
    ROUND(SUM(od.quantity * p.price), 2)
FROM
    order_details AS od
        LEFT JOIN
    pizzas AS p ON od.pizza_id = p.pizza_id;
```


```
  ROUND(SUM(od.quantity *
  p.price), 2)
▶ 817860.05
```

3. IDENTIFY THE HIGHEST-PRICED PIZZA.

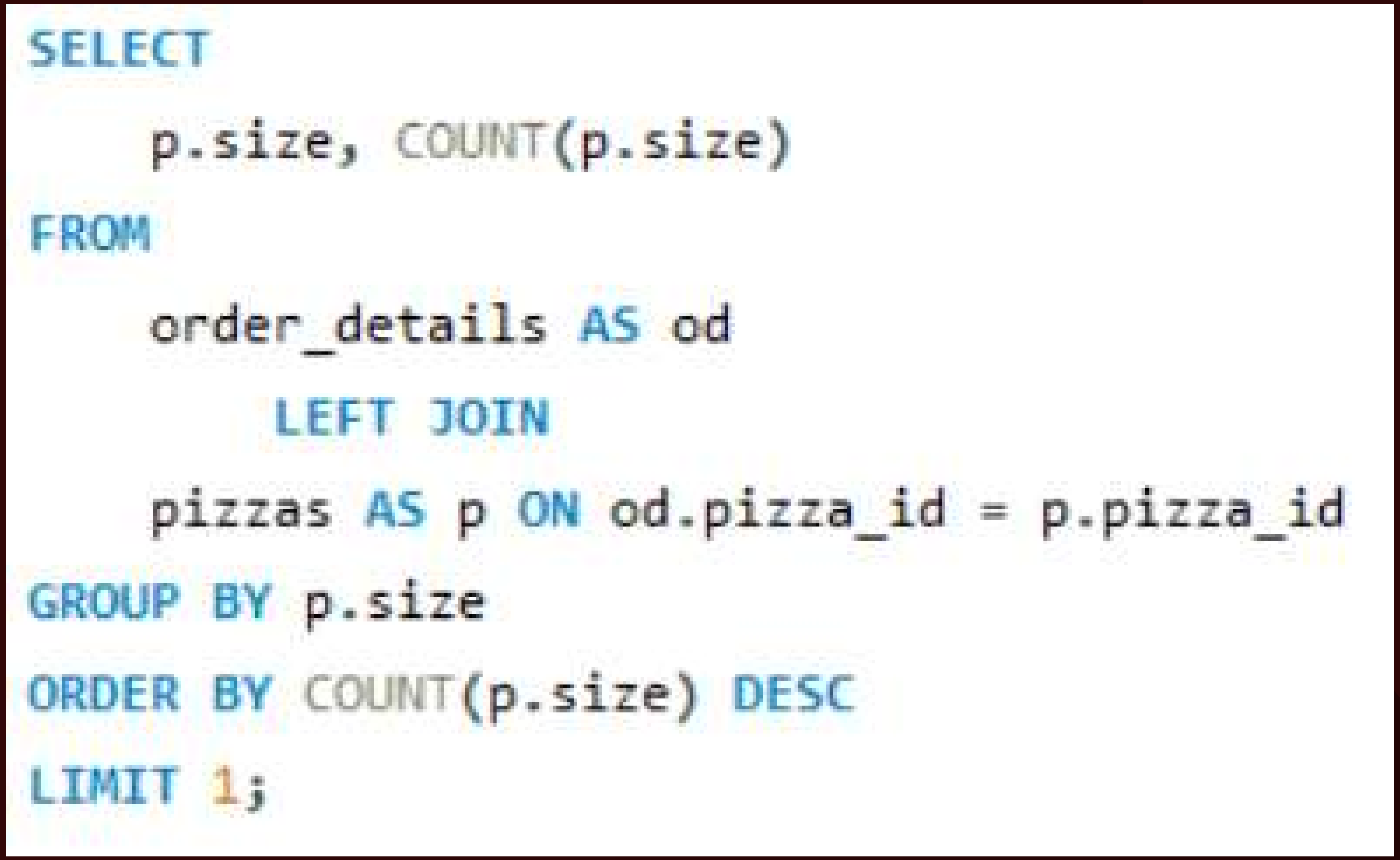


```
SELECT
    pt.name
FROM
    pizzas AS p
        JOIN
    pizza_types AS pt ON p.pizza_type_id = pt.pizza_type_id
ORDER BY p.price DESC
LIMIT 1;
```

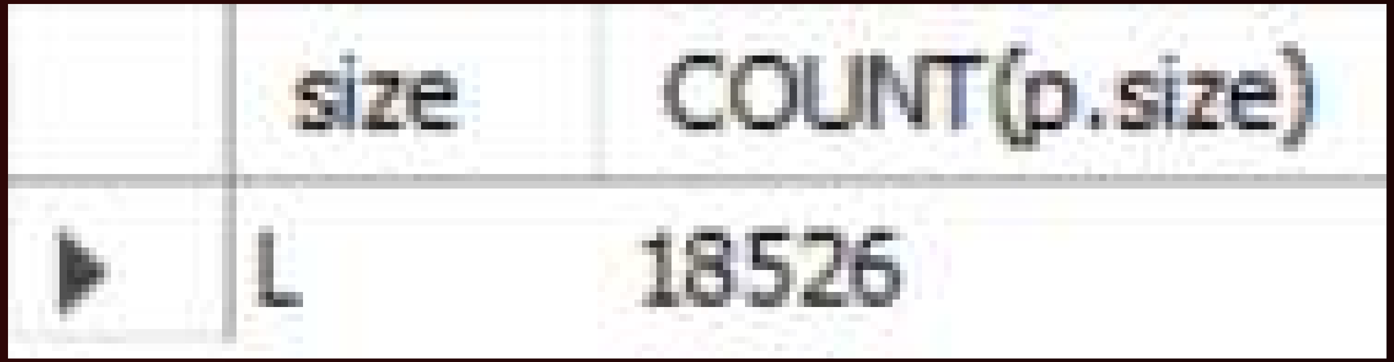


	name
▶	The Greek Pizza

4. IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

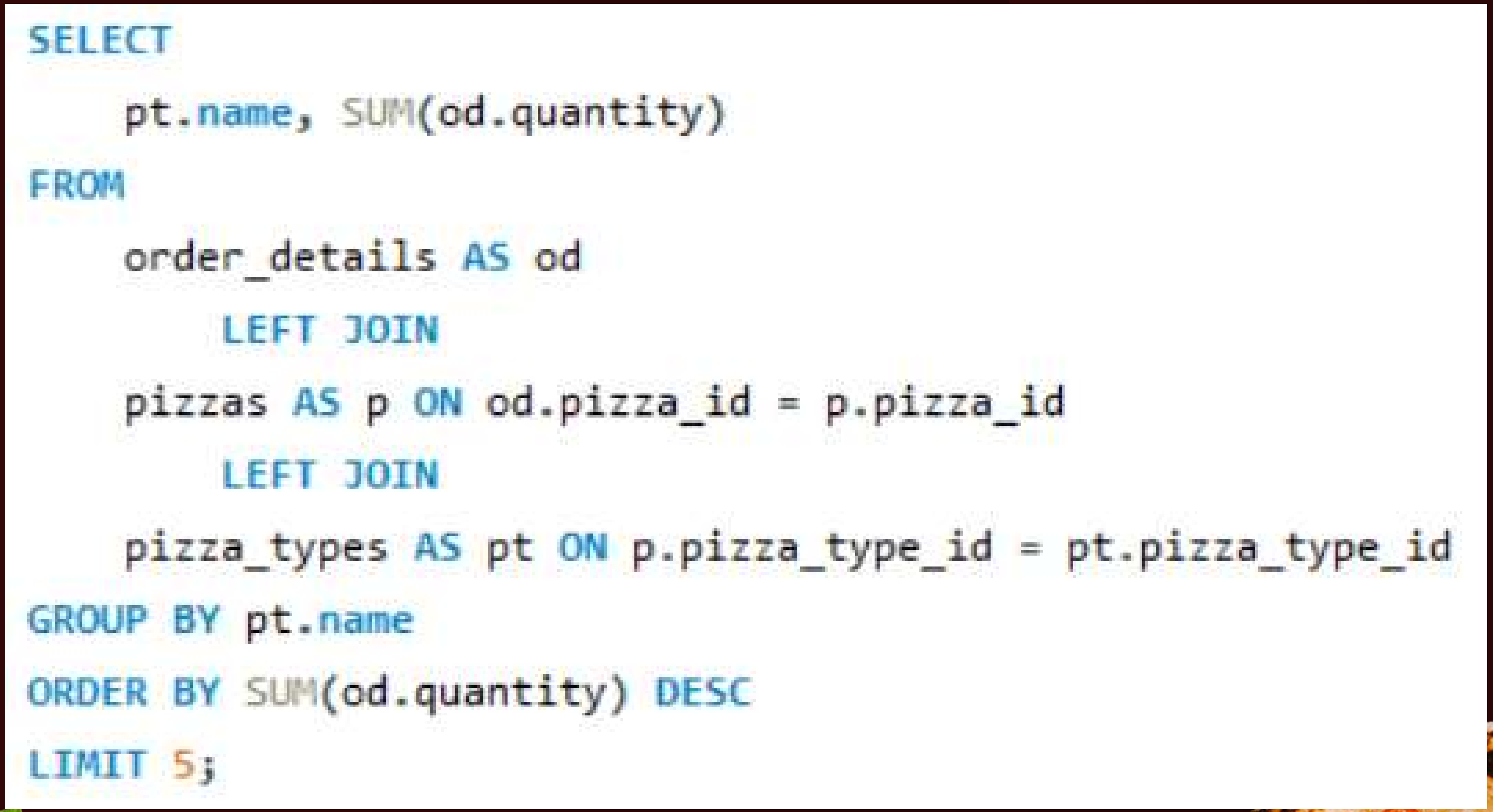


```
SELECT
    p.size, COUNT(p.size)
FROM
    order_details AS od
        LEFT JOIN
    pizzas AS p ON od.pizza_id = p.pizza_id
GROUP BY p.size
ORDER BY COUNT(p.size) DESC
LIMIT 1;
```

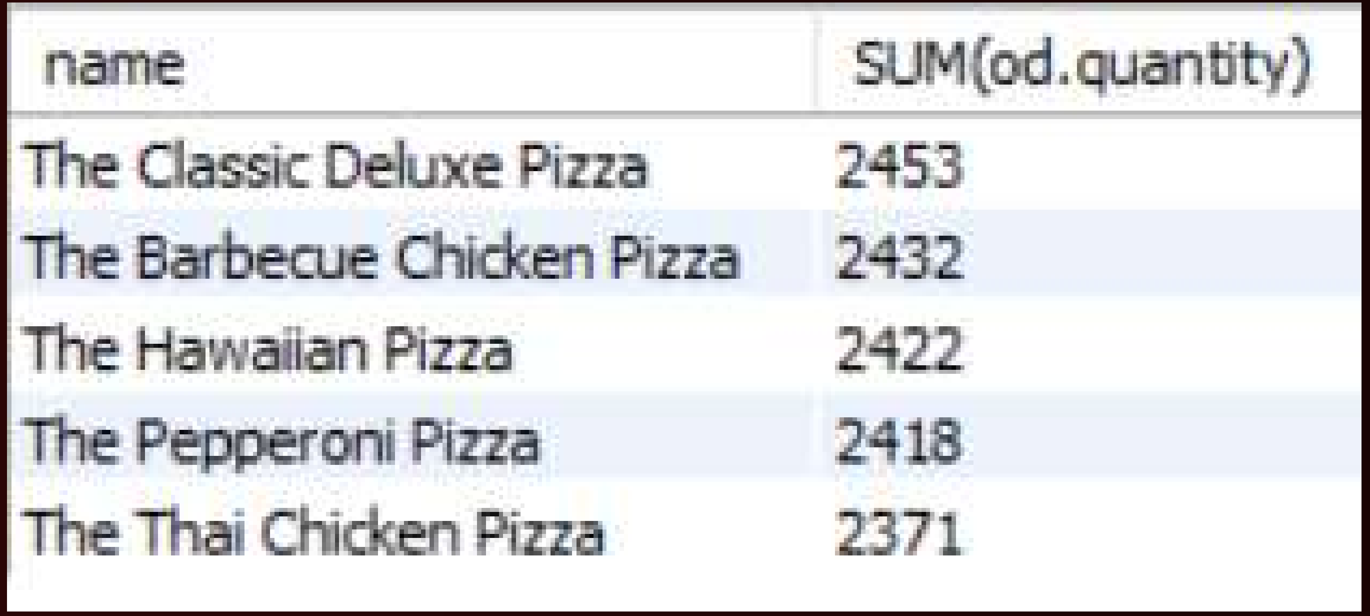


size	COUNT(p.size)
L	18526

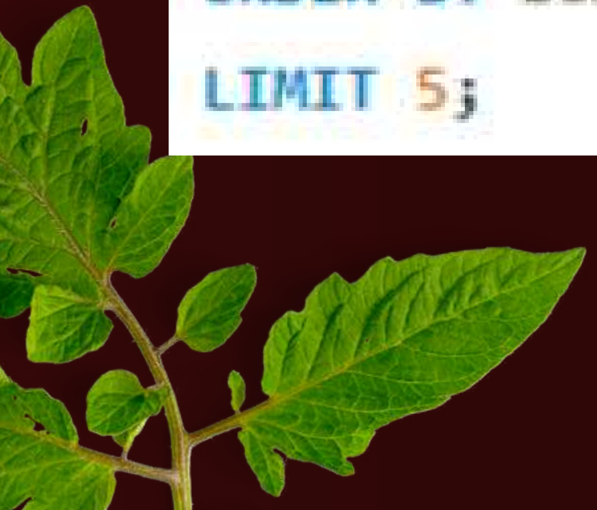
5. LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.



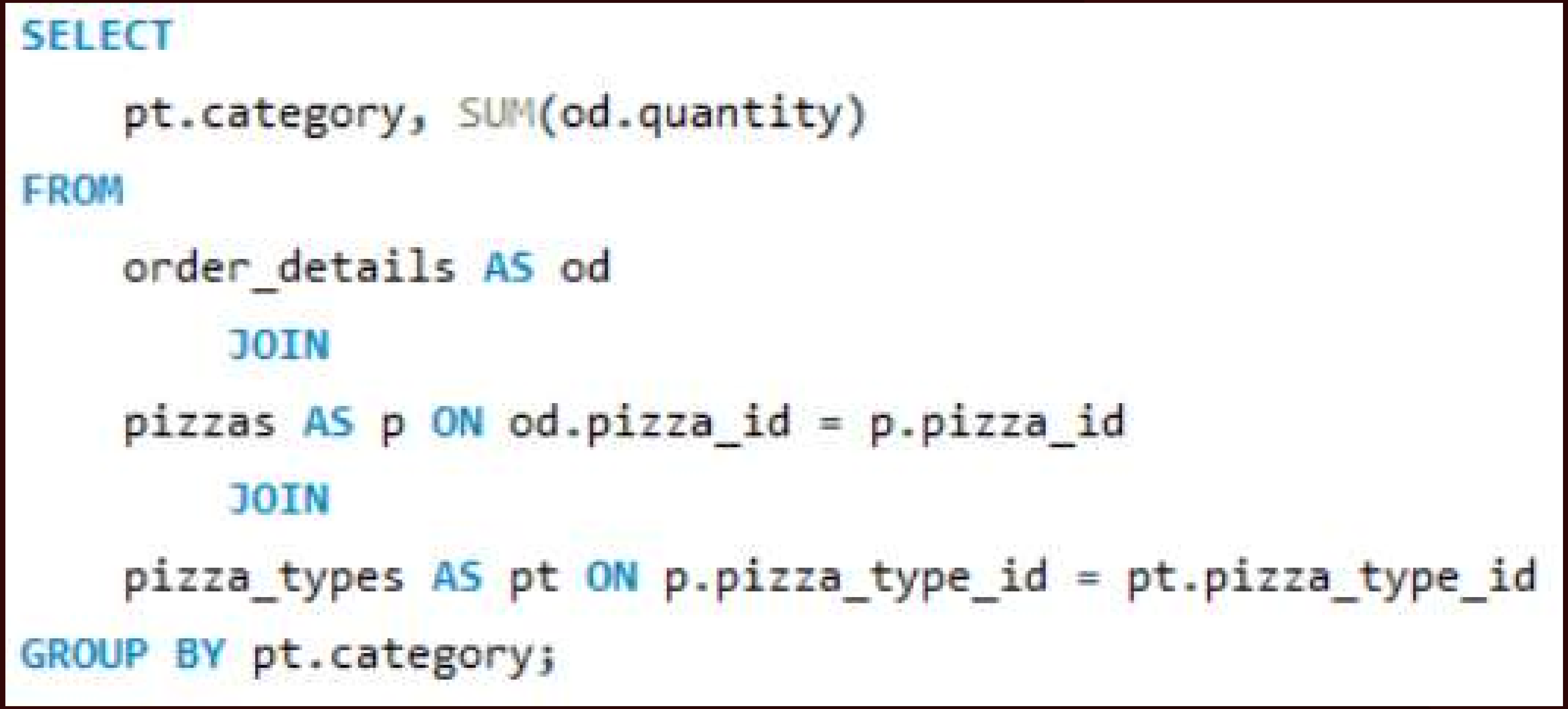
```
SELECT
    pt.name, SUM(od.quantity)
FROM
    order_details AS od
        LEFT JOIN
    pizzas AS p ON od.pizza_id = p.pizza_id
        LEFT JOIN
    pizza_types AS pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.name
ORDER BY SUM(od.quantity) DESC
LIMIT 5;
```



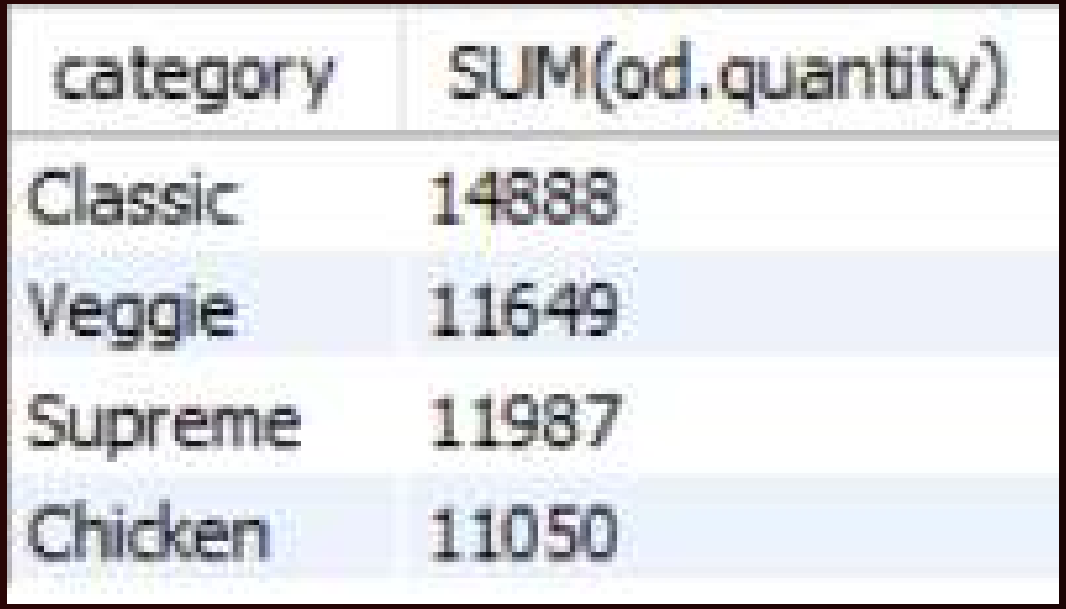
name	SUM(od.quantity)
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371



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

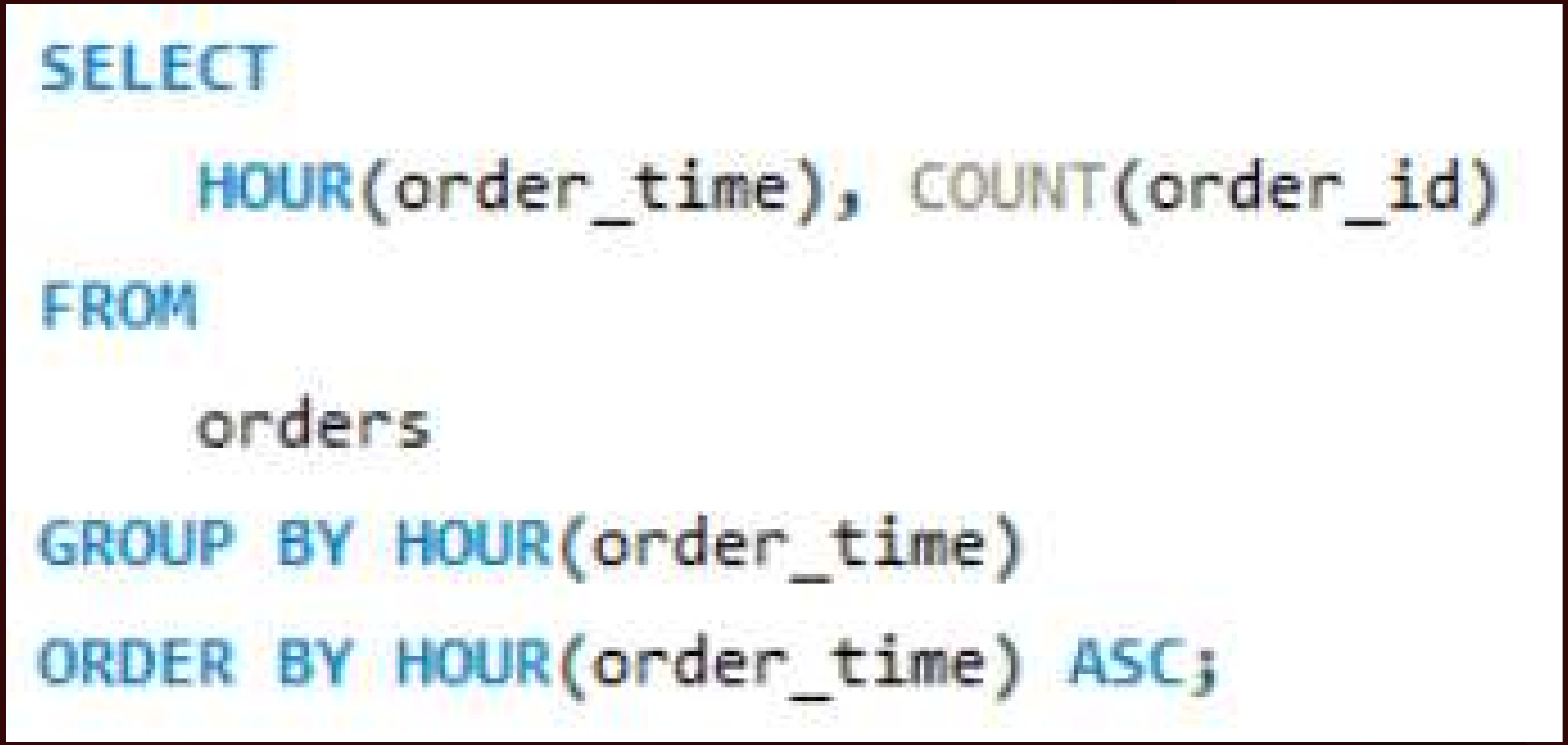


```
SELECT
    pt.category, SUM(od.quantity)
FROM
    order_details AS od
        JOIN
    pizzas AS p ON od.pizza_id = p.pizza_id
        JOIN
    pizza_types AS pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.category;
```

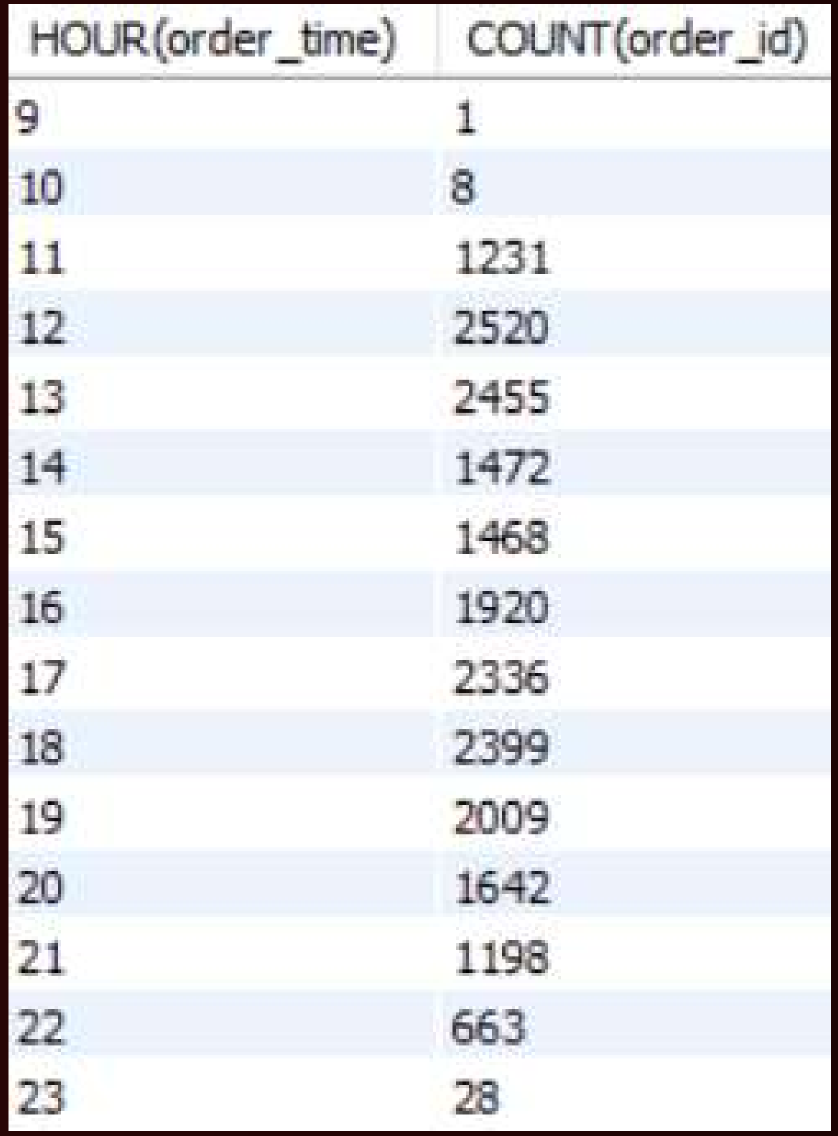


category	SUM(od.quantity)
Classic	14888
Veggie	11649
Supreme	11987
Chicken	11050

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



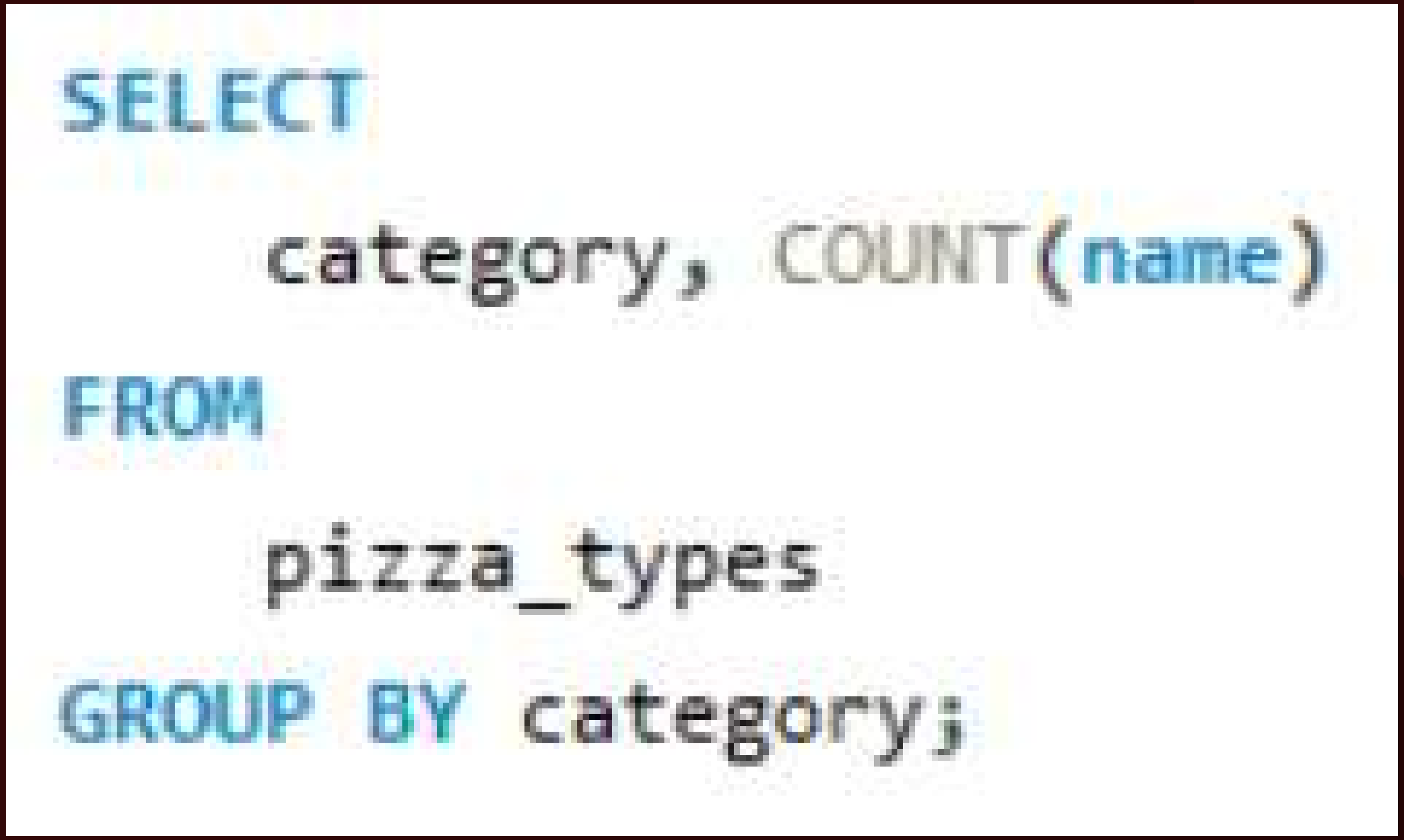
```
SELECT
    HOUR(order_time), COUNT(order_id)
FROM
    orders
GROUP BY HOUR(order_time)
ORDER BY HOUR(order_time) ASC;
```



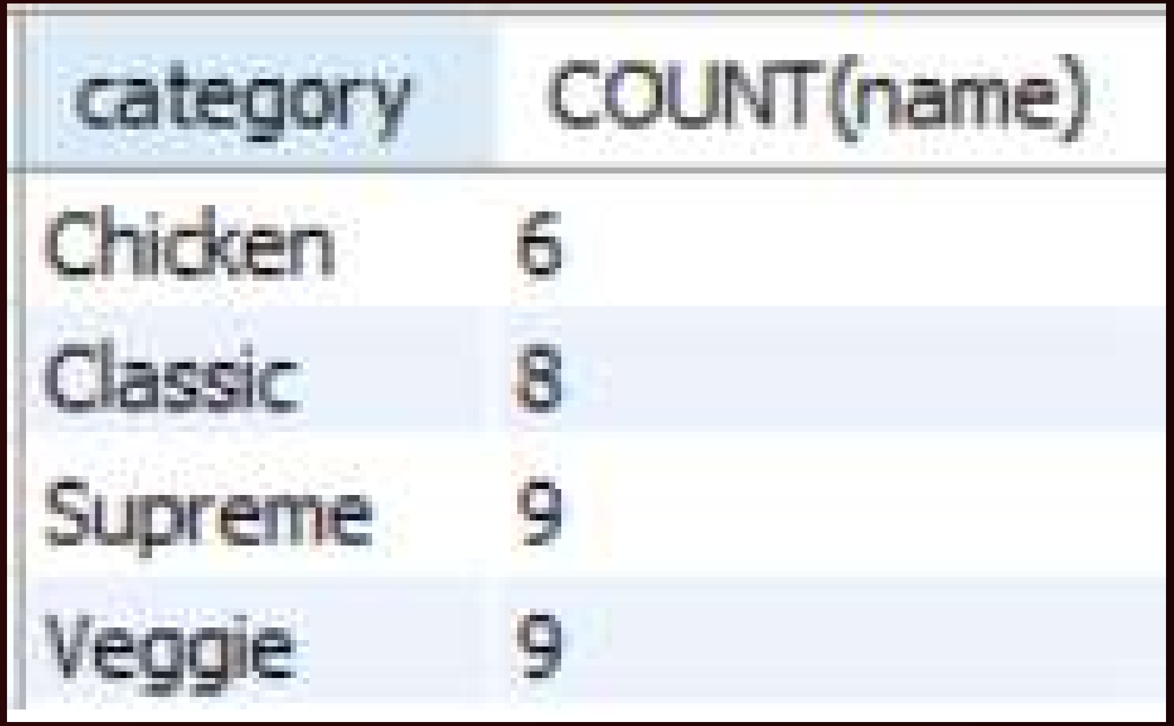
HOUR(order_time)	COUNT(order_id)
9	1
10	8
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009
20	1642
21	1198
22	663
23	28

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

About Contact



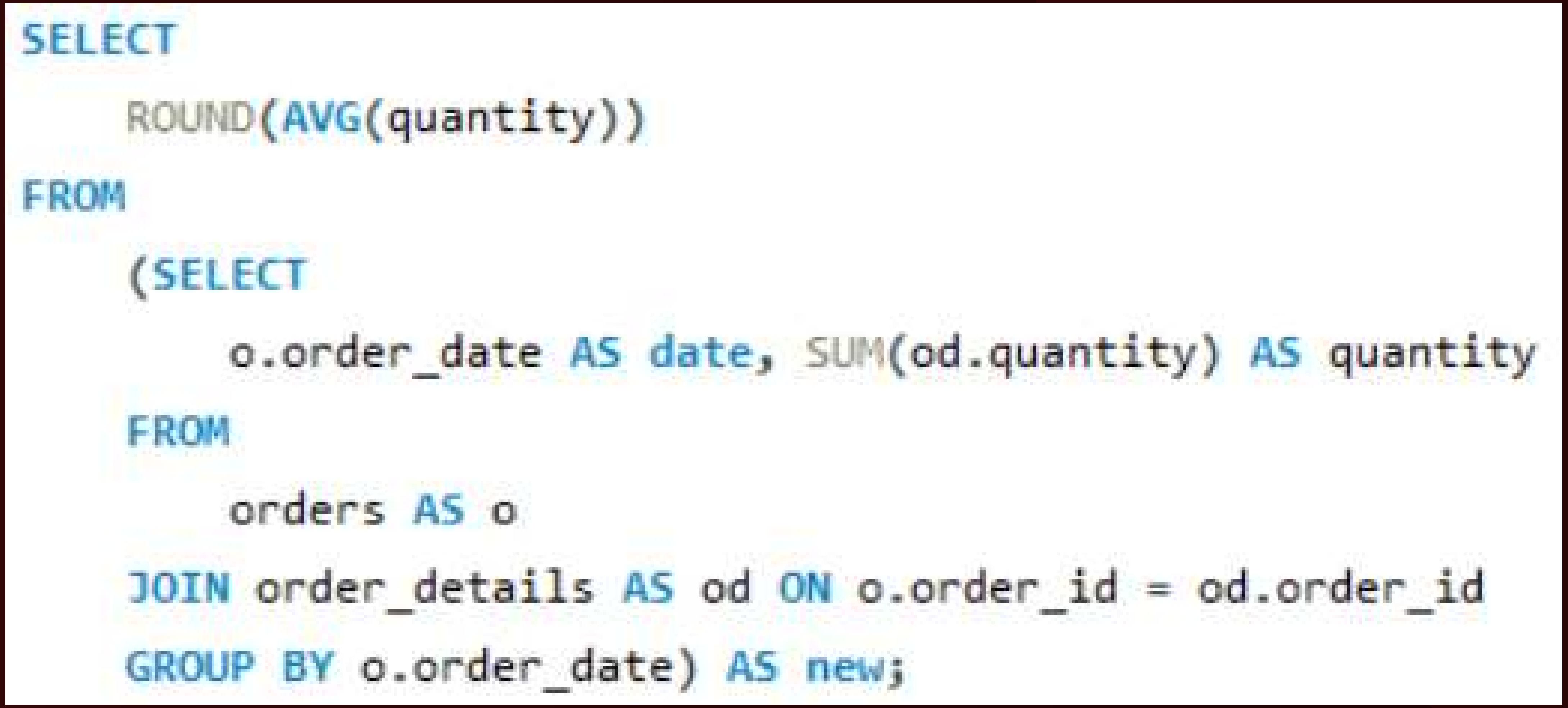
```
SELECT
    category, COUNT(name)
FROM
    pizza_types
GROUP BY category;
```



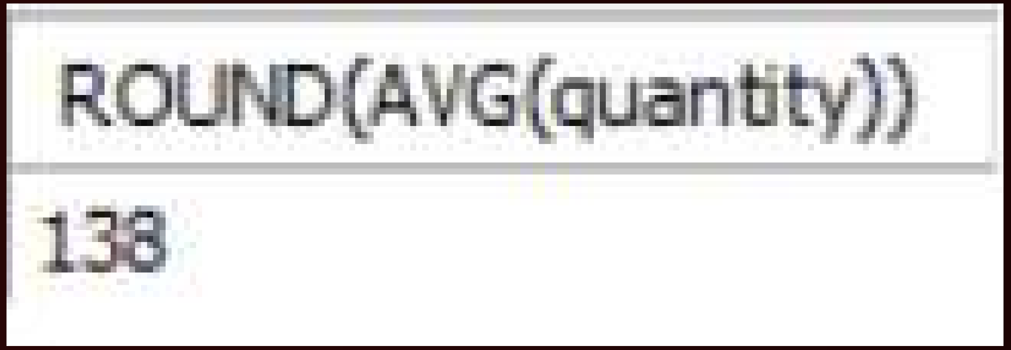
category	COUNT(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

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

Contact



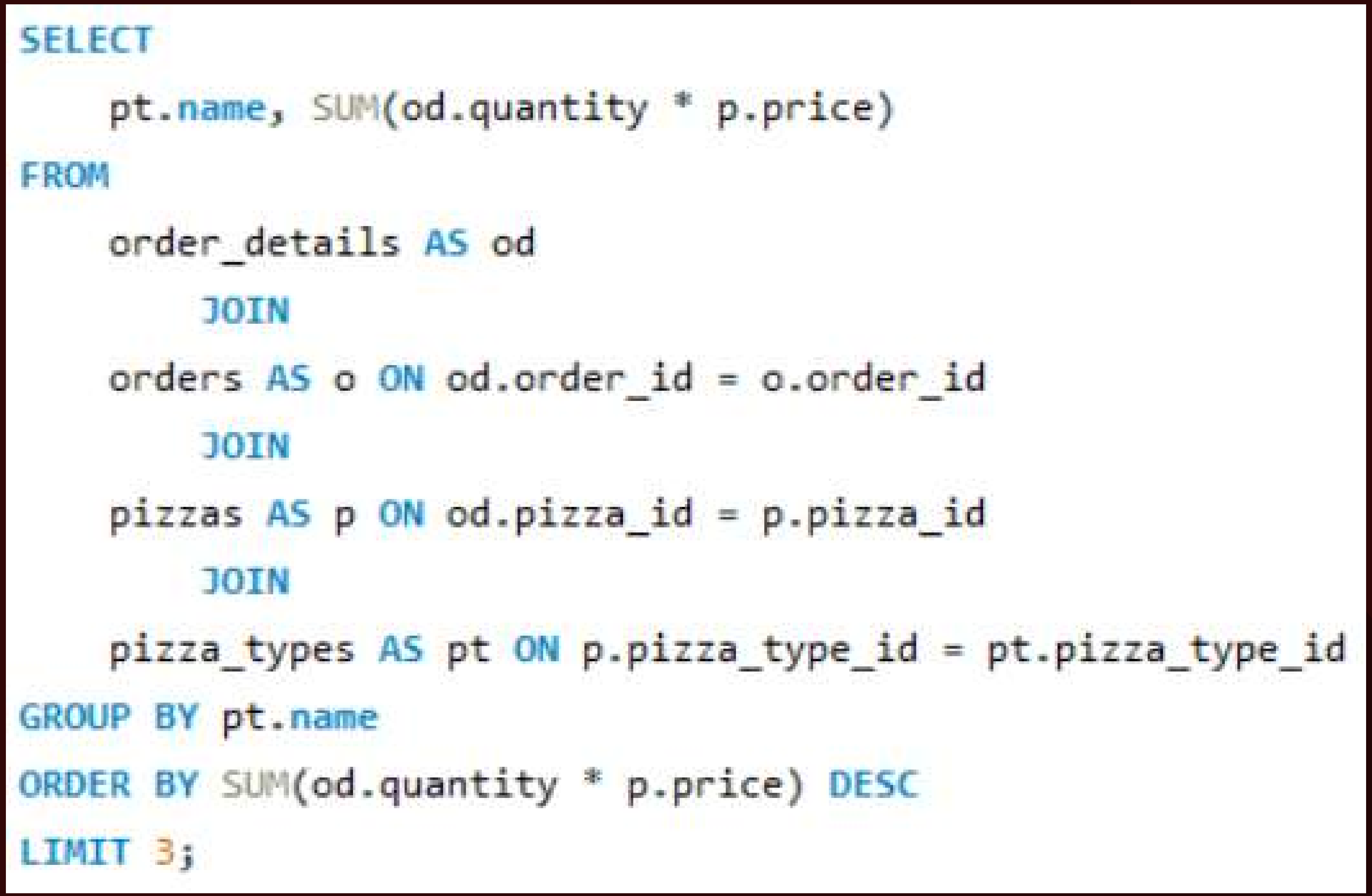
```
SELECT
    ROUND(AVG(quantity))
FROM
    (SELECT
        o.order_date AS date, SUM(od.quantity) AS quantity
    FROM
        orders AS o
    JOIN order_details AS od ON o.order_id = od.order_id
    GROUP BY o.order_date) AS new;
```



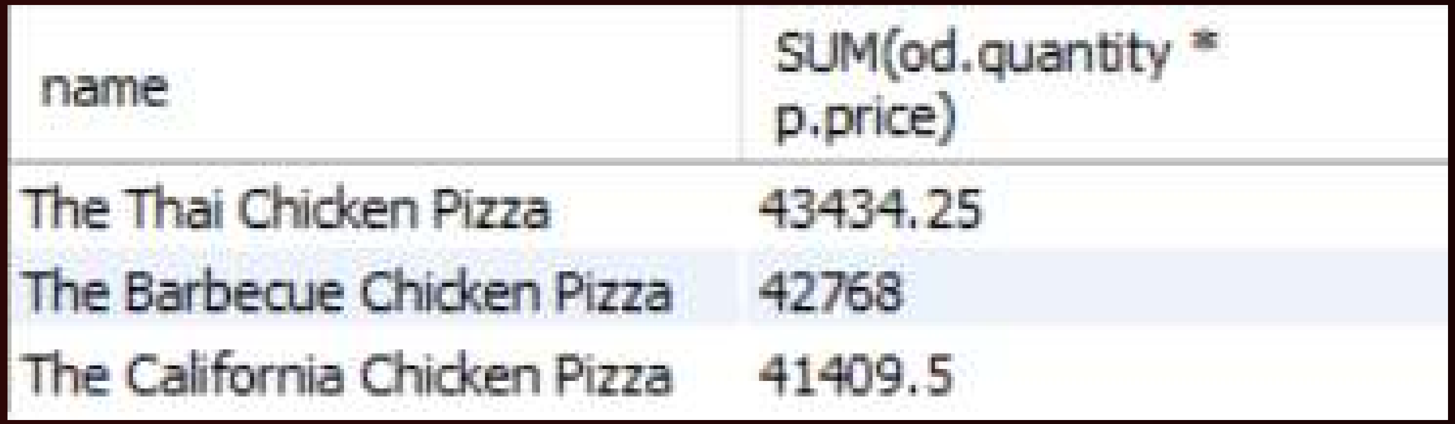
```
ROUND(AVG(quantity))
138
```

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

Contact

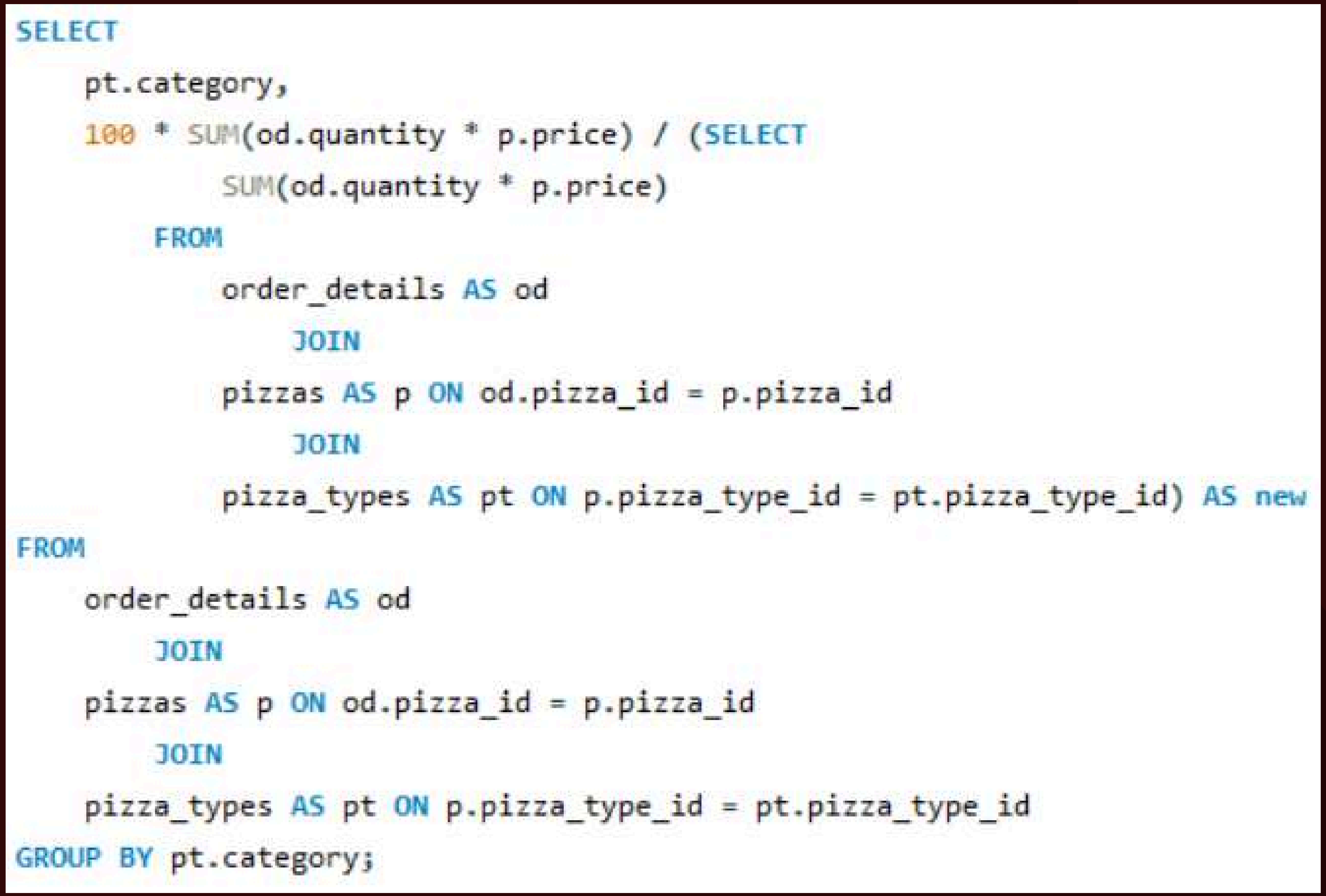


```
SELECT
    pt.name, SUM(od.quantity * p.price)
FROM
    order_details AS od
        JOIN
    orders AS o ON od.order_id = o.order_id
        JOIN
    pizzas AS p ON od.pizza_id = p.pizza_id
        JOIN
    pizza_types AS pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.name
ORDER BY SUM(od.quantity * p.price) DESC
LIMIT 3;
```

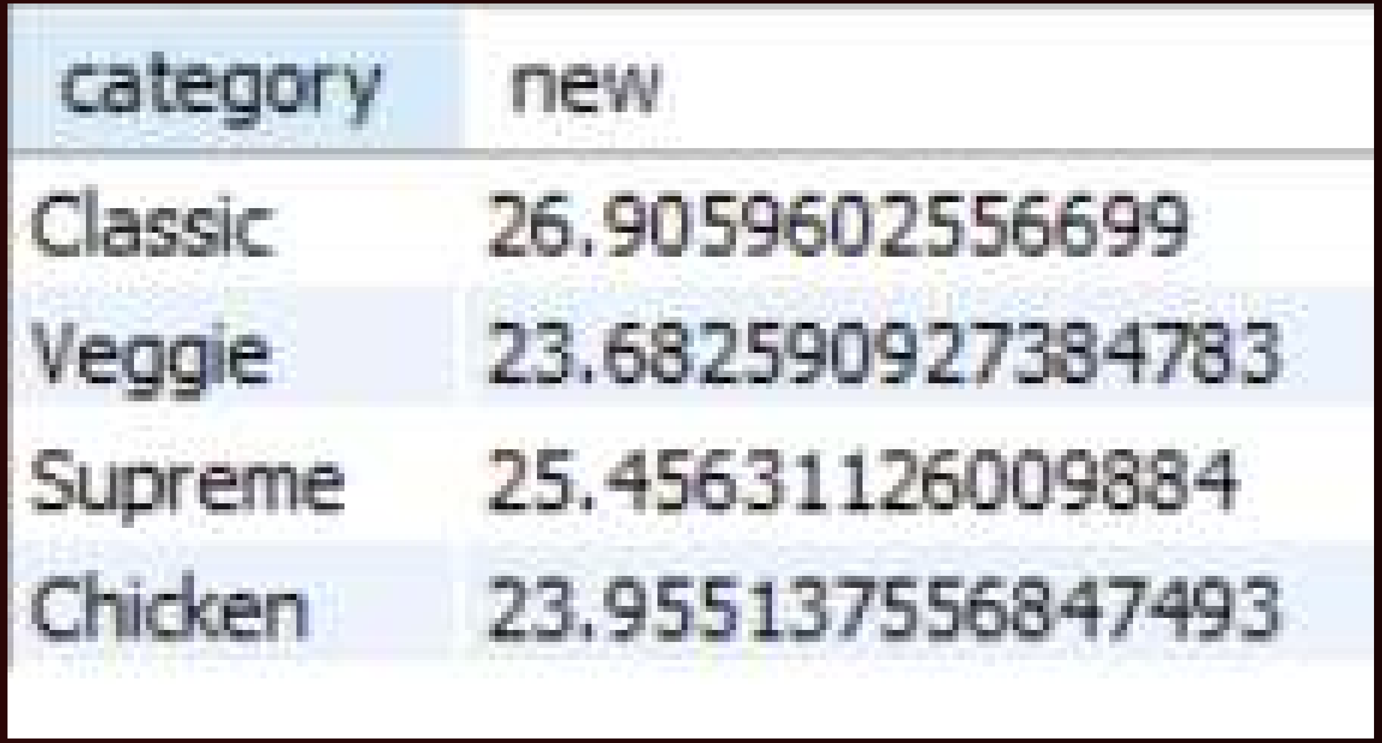


name	SUM(od.quantity * p.price)
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

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



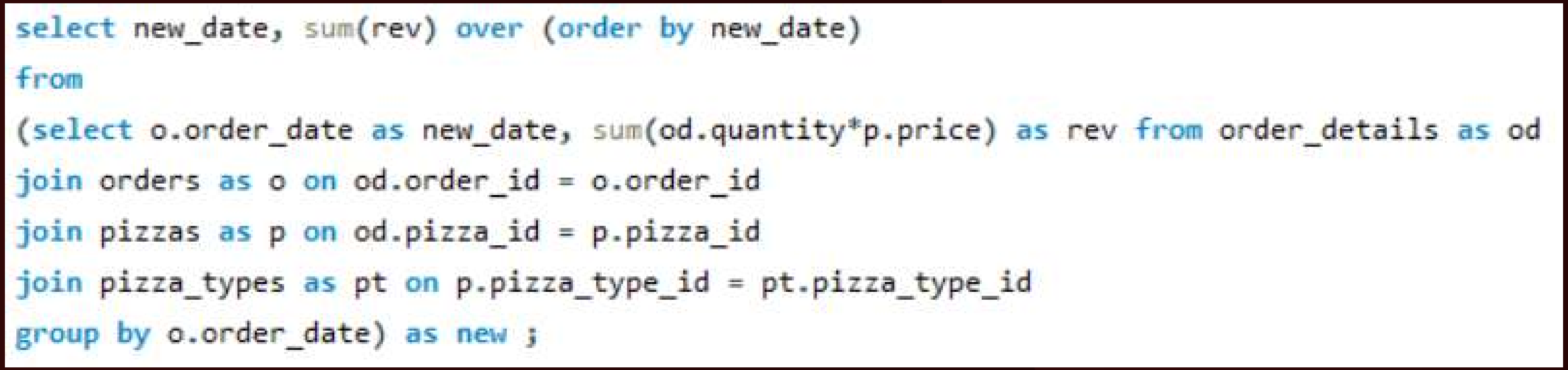
```
SELECT
    pt.category,
    100 * SUM(od.quantity * p.price) / (SELECT
            SUM(od.quantity * p.price)
        FROM
            order_details AS od
                JOIN
            pizzas AS p ON od.pizza_id = p.pizza_id
                JOIN
            pizza_types AS pt ON p.pizza_type_id = pt.pizza_type_id) AS new
FROM
    order_details AS od
        JOIN
    pizzas AS p ON od.pizza_id = p.pizza_id
        JOIN
    pizza_types AS pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.category;
```



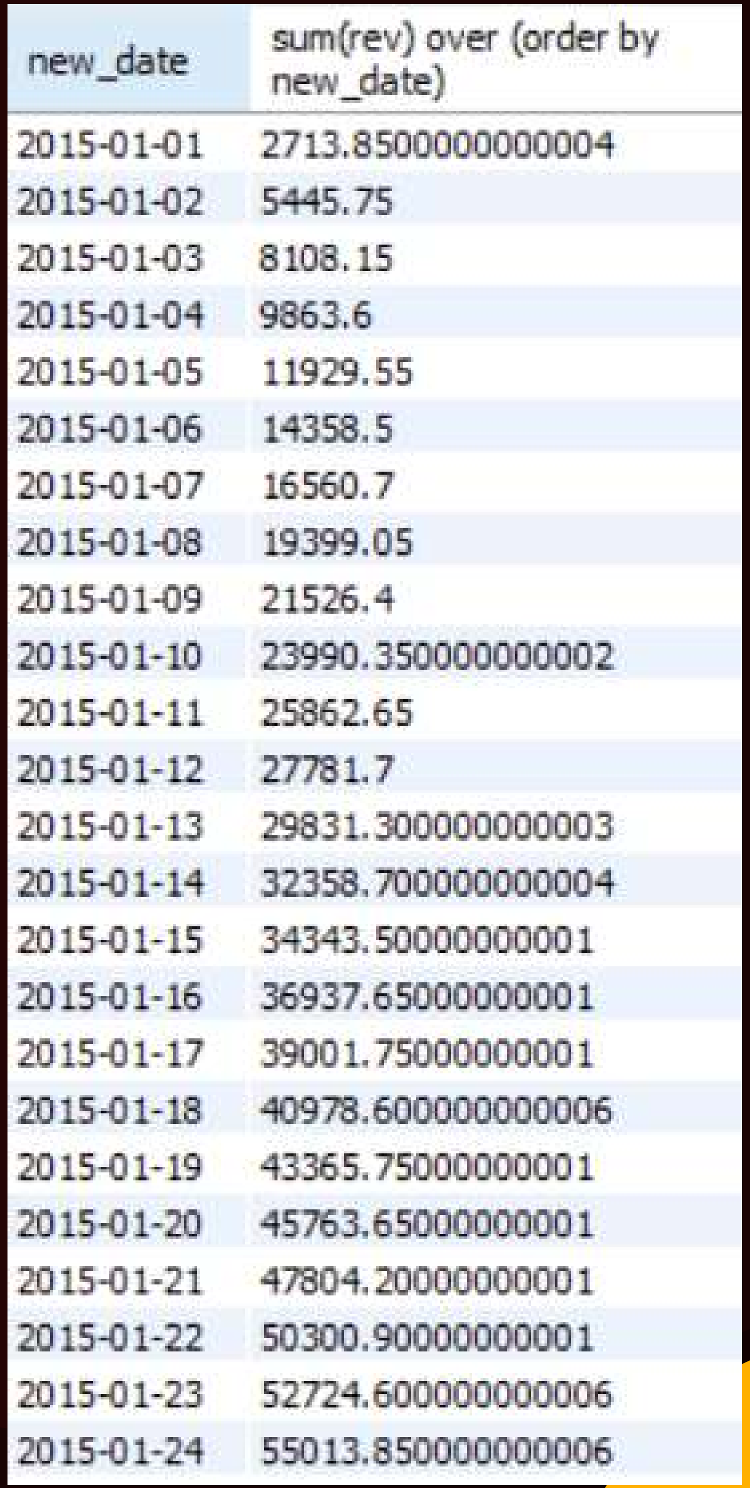
category	new
Classic	26.905960255669
Veggie	23.682590927384783
Supreme	25.4563112600984
Chicken	23.955137556847493

12. ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

Contact

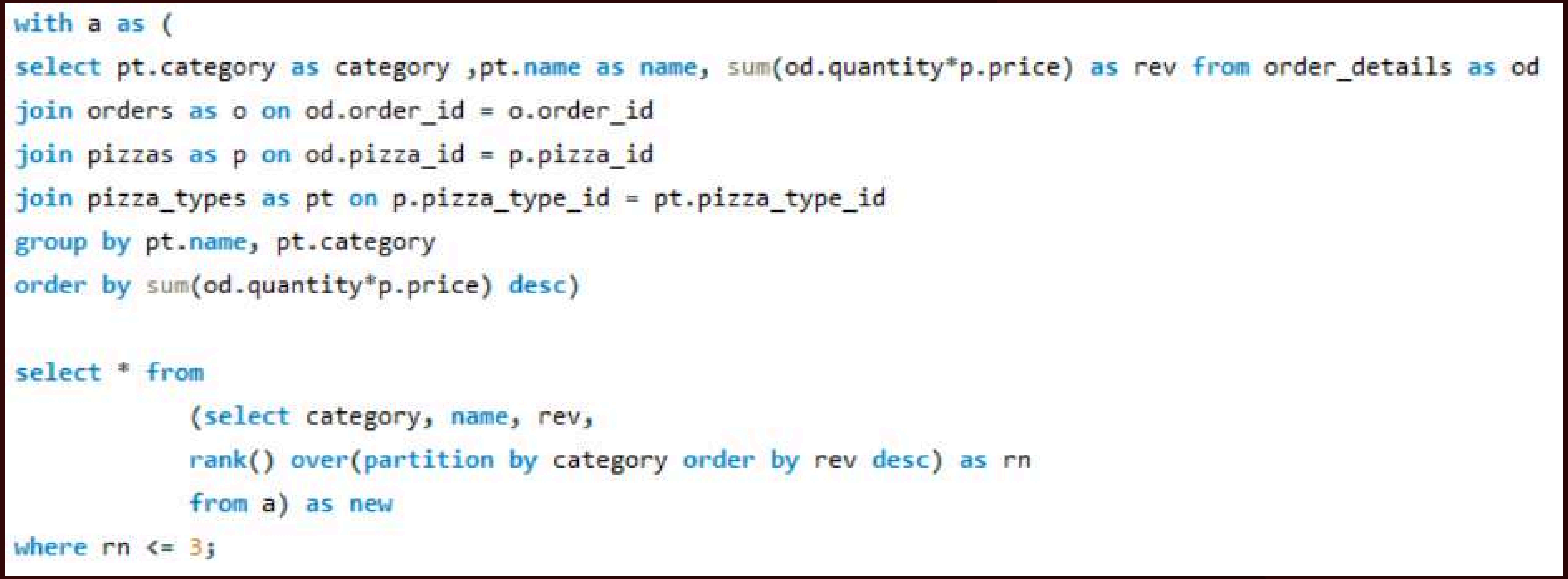


```
select new_date, sum(rev) over (order by new_date)
from
(select o.order_date as new_date, sum(od.quantity*p.price) as rev from order_details as od
join orders as o on od.order_id = o.order_id
join pizzas as p on od.pizza_id = p.pizza_id
join pizza_types as pt on p.pizza_type_id = pt.pizza_type_id
group by o.order_date) as new ;
```



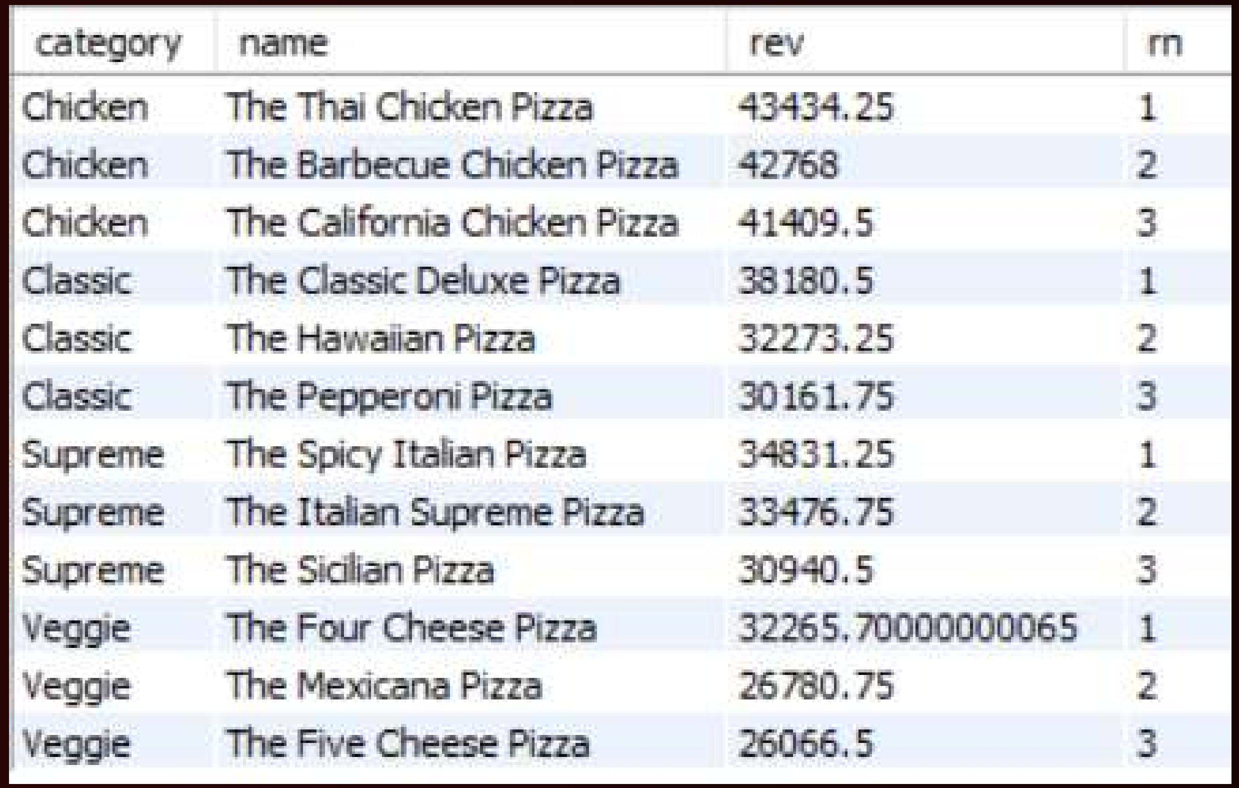
new_date	sum(rev) over (order by new_date)
2015-01-01	2713.8500000000004
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55
2015-01-06	14358.5
2015-01-07	16560.7
2015-01-08	19399.05
2015-01-09	21526.4
2015-01-10	23990.350000000002
2015-01-11	25862.65
2015-01-12	27781.7
2015-01-13	29831.300000000003
2015-01-14	32358.700000000004
2015-01-15	34343.50000000001
2015-01-16	36937.65000000001
2015-01-17	39001.75000000001
2015-01-18	40978.600000000006
2015-01-19	43365.75000000001
2015-01-20	45763.65000000001
2015-01-21	47804.20000000001
2015-01-22	50300.90000000001
2015-01-23	52724.600000000006
2015-01-24	55013.850000000006

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



```
with a as (
select pt.category as category ,pt.name as name, sum(od.quantity*p.price) as rev from order_details as od
join orders as o on od.order_id = o.order_id
join pizzas as p on od.pizza_id = p.pizza_id
join pizza_types as pt on p.pizza_type_id = pt.pizza_type_id
group by pt.name, pt.category
order by sum(od.quantity*p.price) desc)

select * from
        (select category, name, rev,
        rank() over(partition by category order by rev desc) as rn
        from a) as new
where rn <= 3;
```



category	name	rev	rn
Chicken	The Thai Chicken Pizza	43434.25	1
Chicken	The Barbecue Chicken Pizza	42768	2
Chicken	The California Chicken Pizza	41409.5	3
Classic	The Classic Deluxe Pizza	38180.5	1
Classic	The Hawaiian Pizza	32273.25	2
Classic	The Pepperoni Pizza	30161.75	3
Supreme	The Spicy Italian Pizza	34831.25	1
Supreme	The Italian Supreme Pizza	33476.75	2
Supreme	The Sicilian Pizza	30940.5	3
Veggie	The Four Cheese Pizza	32265.70000000065	1
Veggie	The Mexicana Pizza	26780.75	2
Veggie	The Five Cheese Pizza	26066.5	3

Home About Contact

THANK YOU