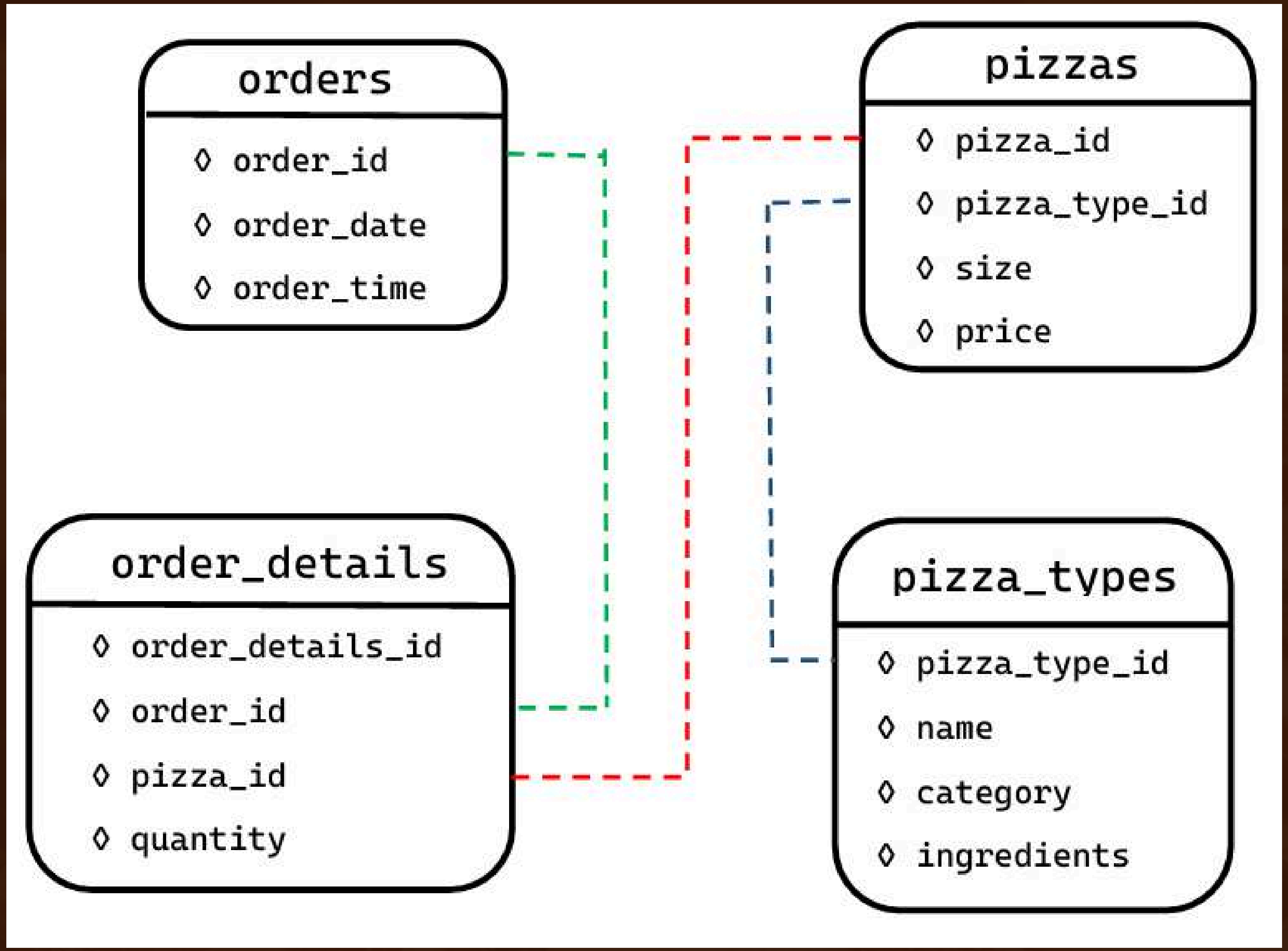
SQL- BASED ANALYSIS OF PIZZA SALES FOR LA PIZZERIA

HELLO!

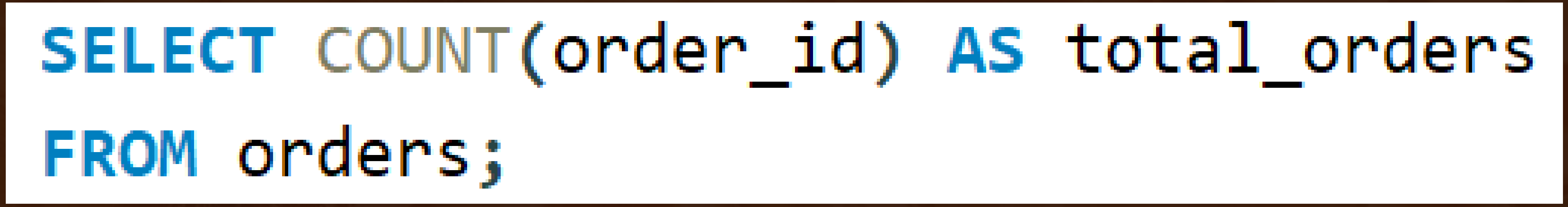
This is a project created utilizing SQL queries to address and solve various questions related to pizza sales. The project involves analyzing and extracting meaningful insights from data to help understand sales trends and customer preferences within the pizza business. By leveraging SQL, I was able to manipulate and query large datasets efficiently, enabling data-driven decision-making.

INSIGHTS OF OUR SCHEMA

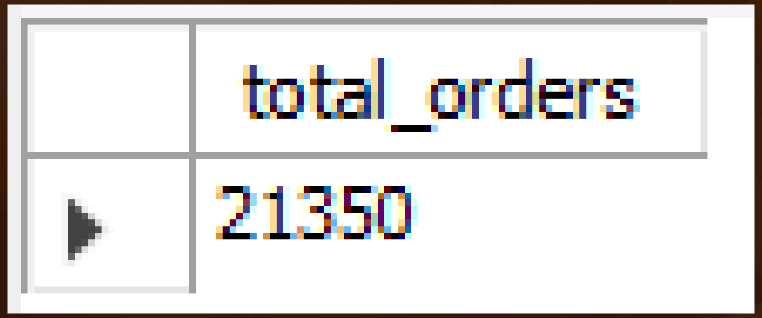
Now, let me introduce you to the data schema that forms the foundation of this project. The schema is designed to organize and structure the data related to pizza sales, including key entities like pizza, pizza types, orders and order details. It defines how the data is stored, how different tables relate to one another, and ensures that we can efficiently query the database to derive insights. With this clear schema, we can better analyze sales trends and make data-driven decisions.



RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED

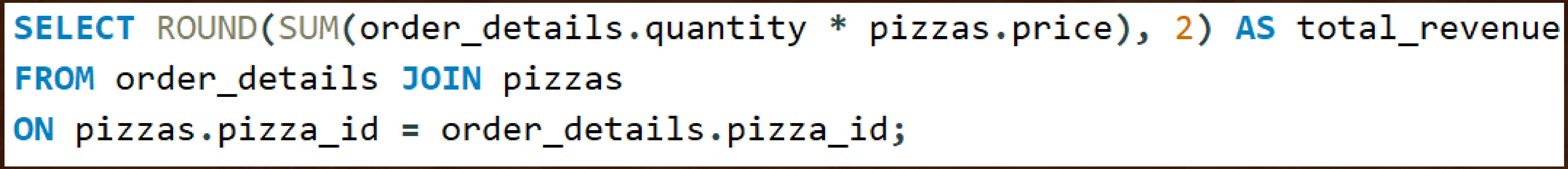


```
SELECT COUNT(order_id) AS total_orders
FROM orders;
```

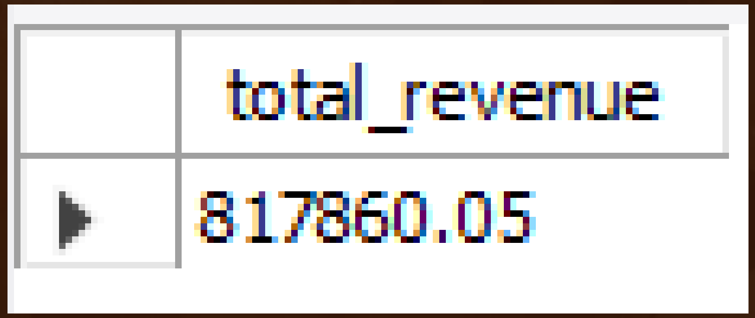


```
total_orders
21350
```

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES

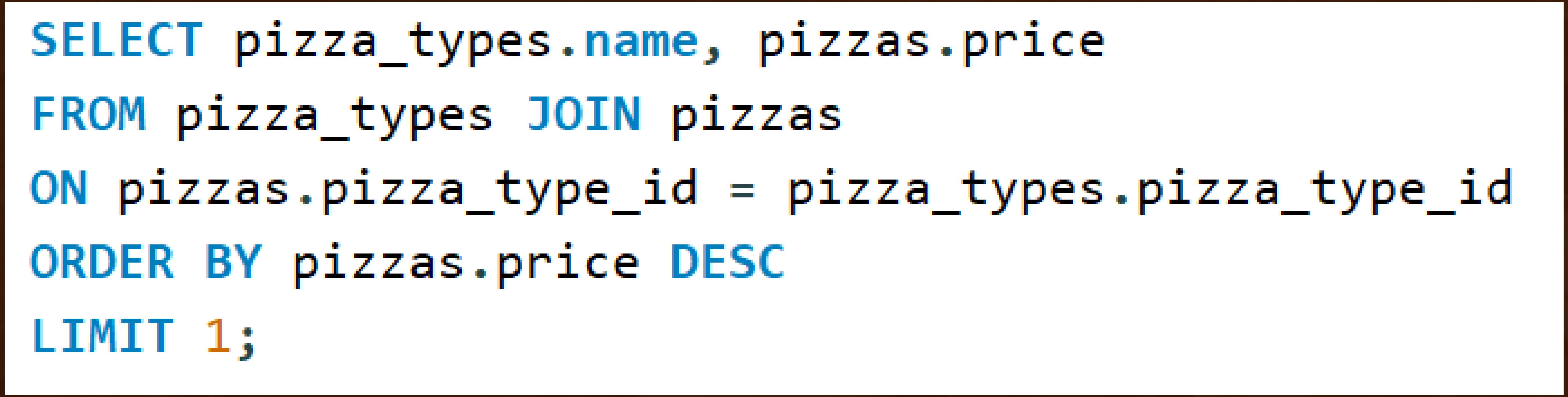


```
SELECT ROUND(SUM(order_details.quantity * pizzas.price), 2) AS total_revenue
FROM order_details JOIN pizzas
ON pizzas.pizza_id = order_details.pizza_id;
```

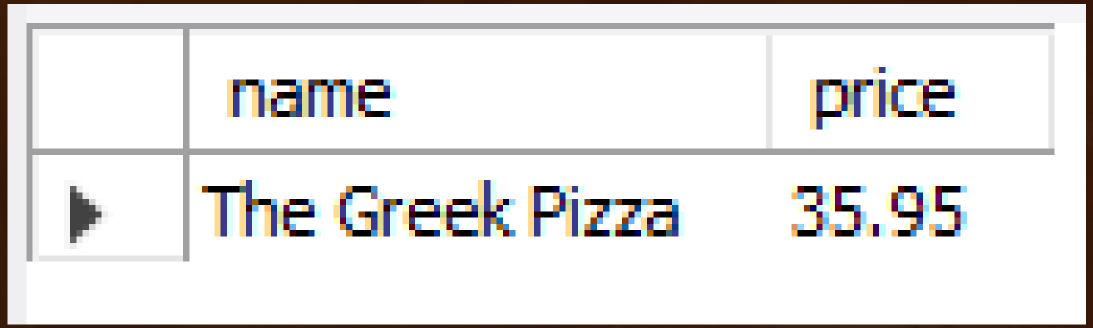


total_revenue
817860.05

IDENTIFY THE HIGHEST-PRICED PIZZA

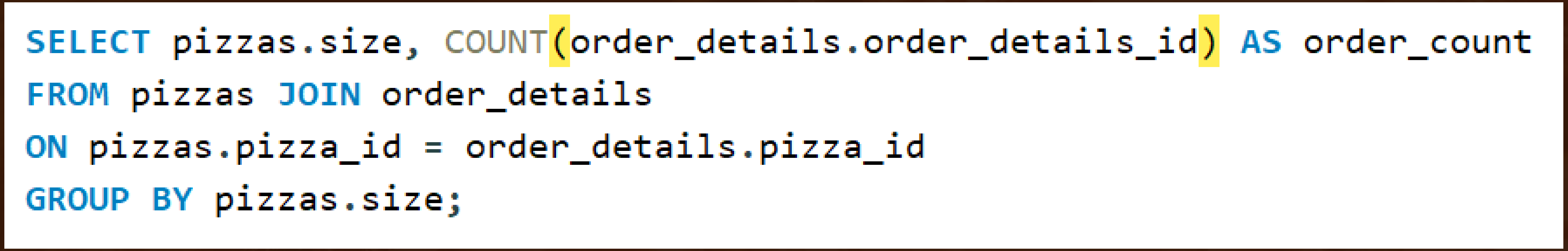


```
SELECT pizza_types.name, pizzas.price
FROM pizza_types JOIN pizzas
ON pizzas.pizza_type_id = pizza_types.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
```

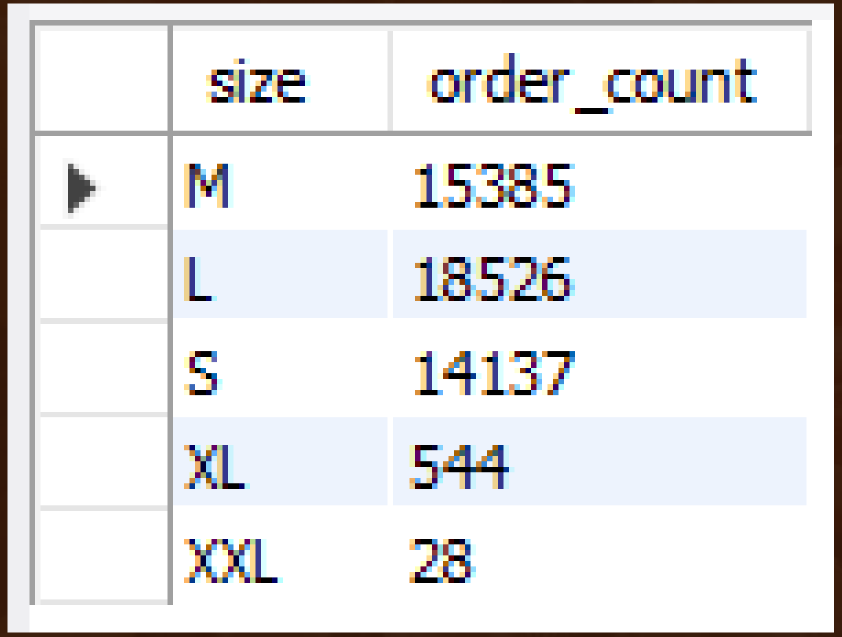


name	price
The Greek Pizza	35.95

IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED

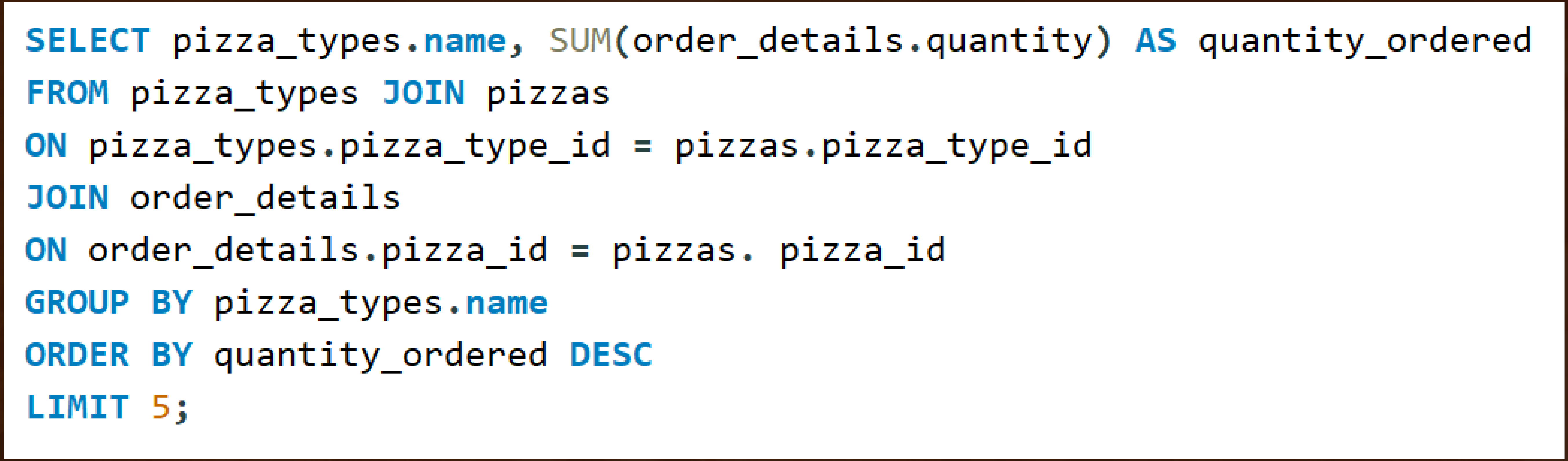


```
SELECT pizzas.size, COUNT(order_details.order_details_id) AS order_count
FROM pizzas JOIN order_details
ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size;
```

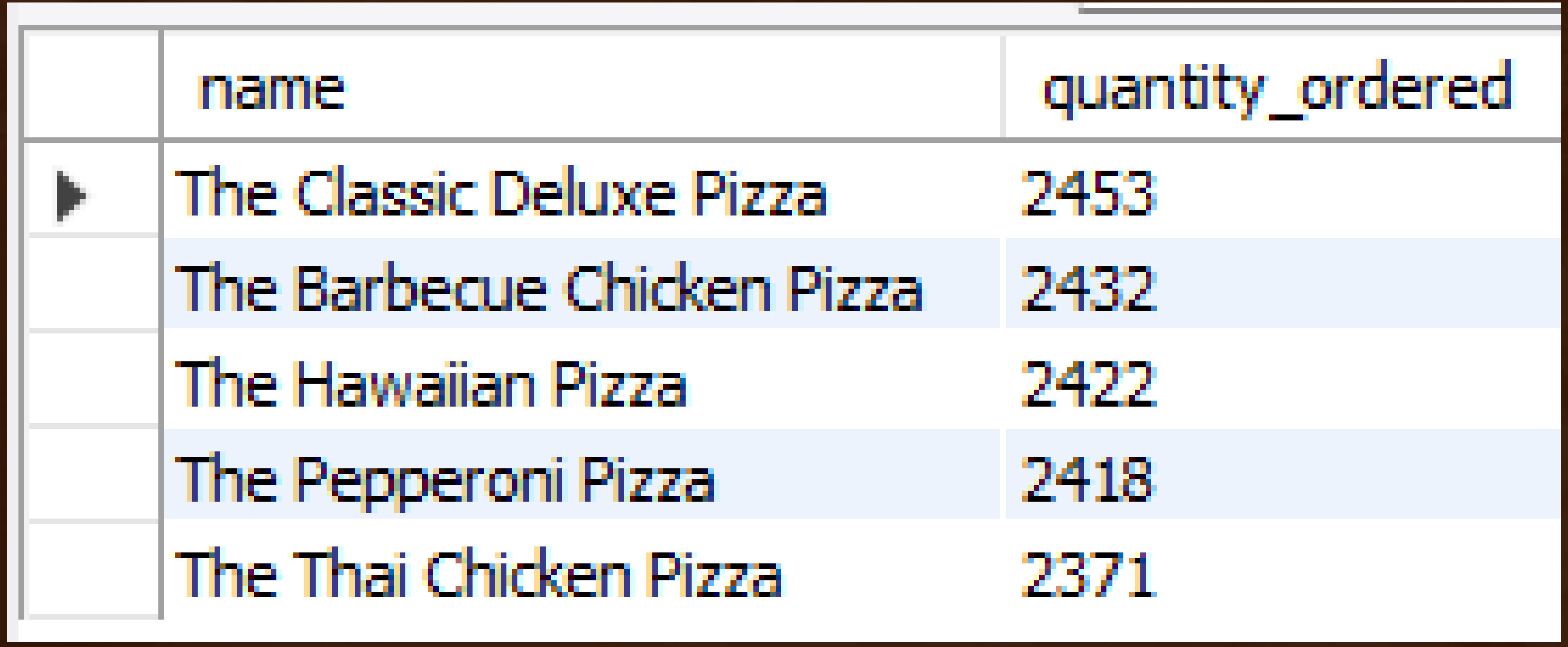


size	order_count
M	15385
L	18526
S	14137
XL	544
XXL	28

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

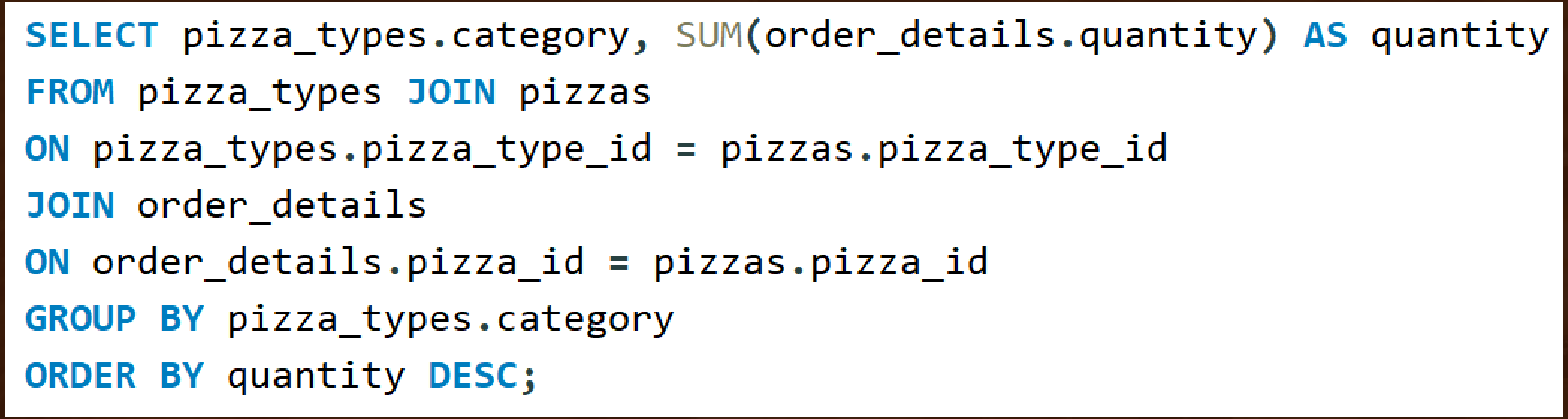


```
SELECT pizza_types.name, SUM(order_details.quantity) AS quantity_ordered
FROM pizza_types JOIN pizzas
ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN order_details
ON order_details.pizza_id = pizzas. pizza_id
GROUP BY pizza_types.name
ORDER BY quantity_ordered DESC
LIMIT 5;
```

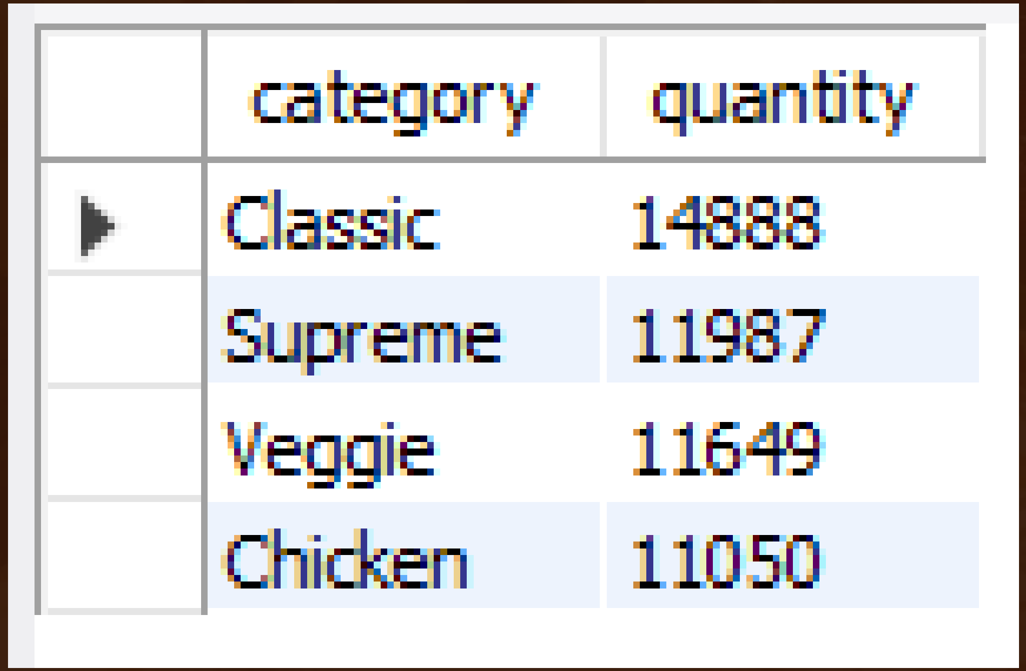


name	quantity_ordered
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

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

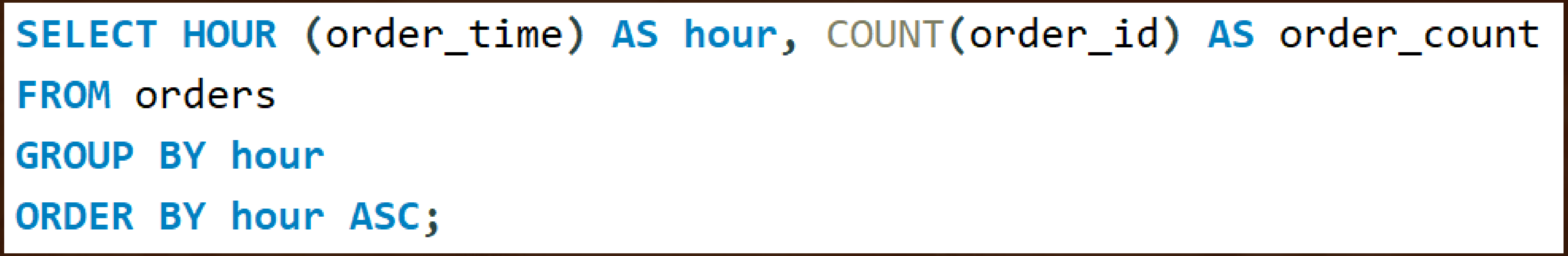


```sql
SELECT pizza_types.category, SUM(order_details.quantity) AS quantity
FROM pizza_types JOIN pizzas
ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN order_details
ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity DESC;
```

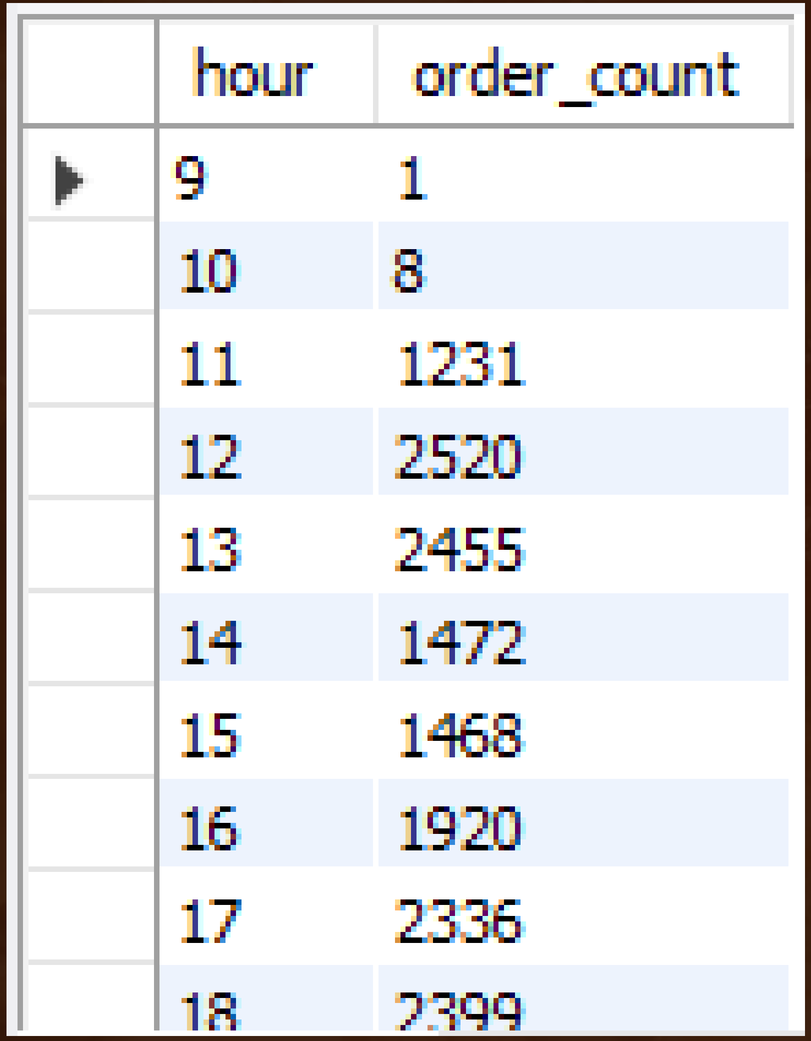


category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY

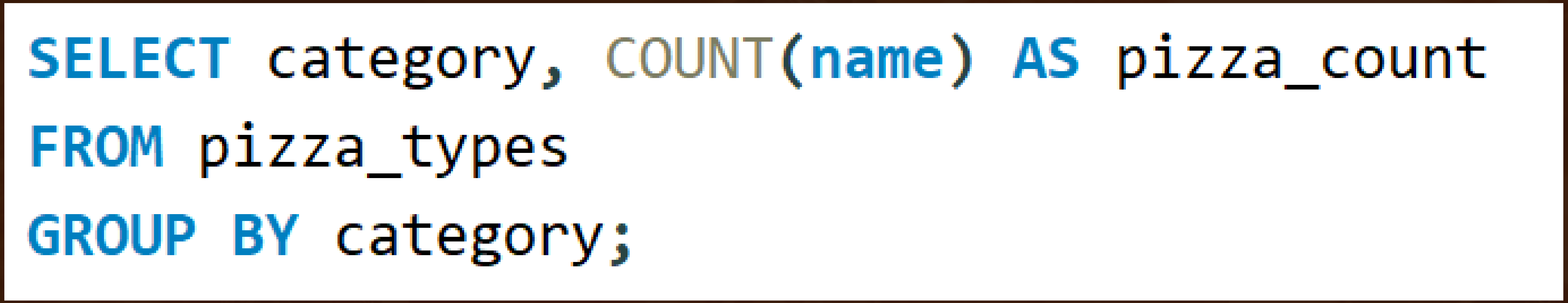


```sql
SELECT HOUR (order_time) AS hour, COUNT(order_id) AS order_count
FROM orders
GROUP BY hour
ORDER BY hour ASC;
```

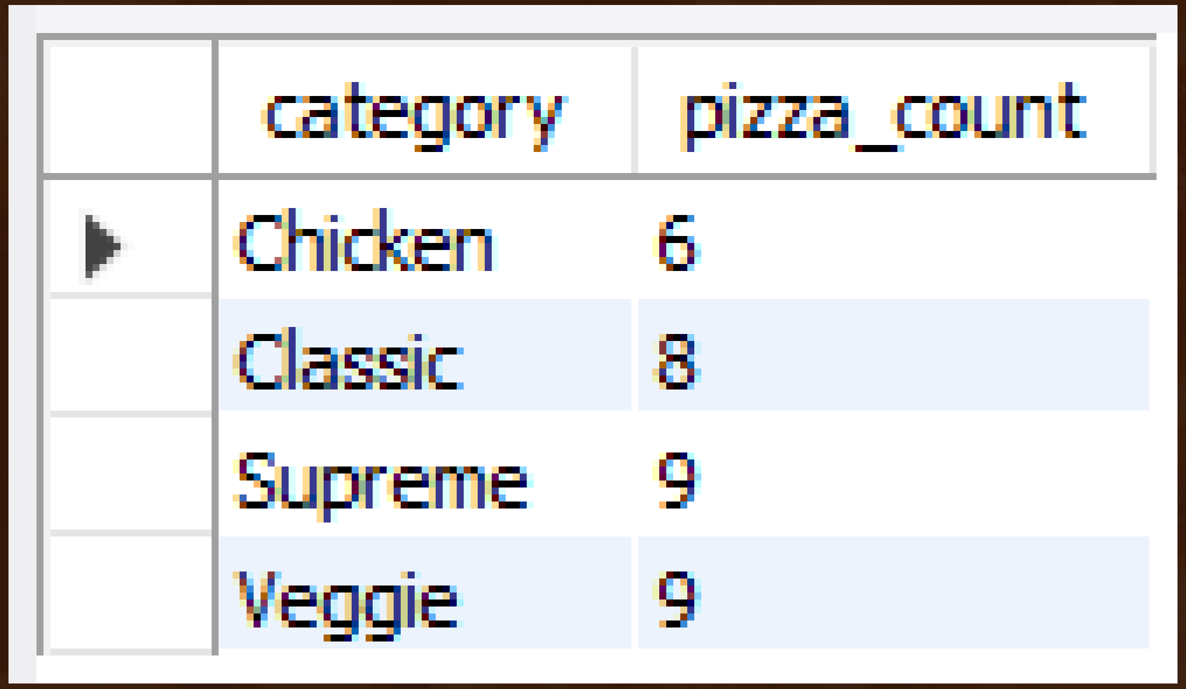


hour	order_count
9	1
10	8
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399

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

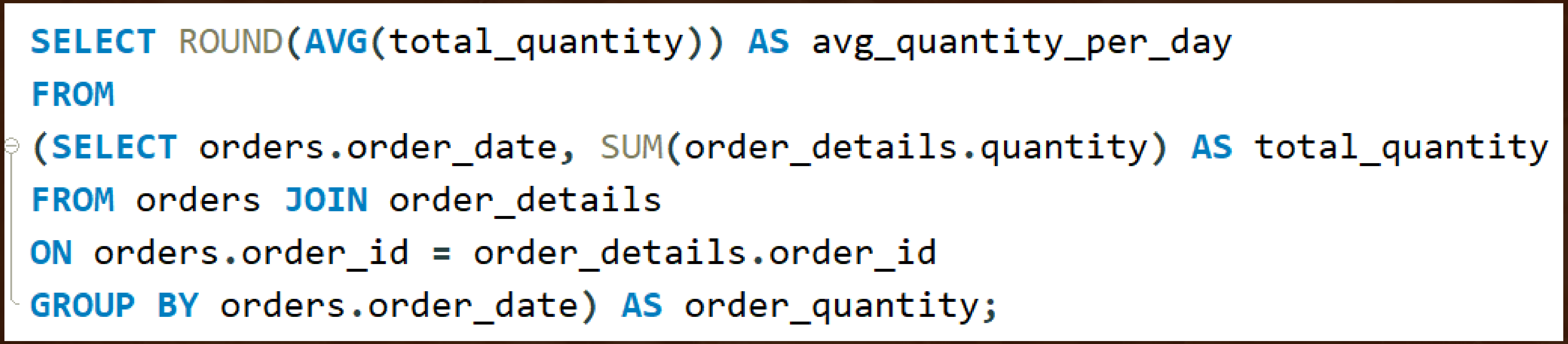


```
SELECT category, COUNT(name) AS pizza_count
FROM pizza_types
GROUP BY category;
```

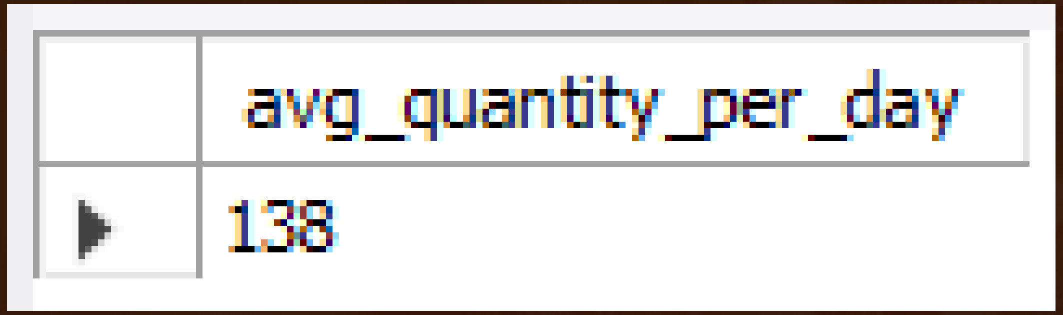


category	pizza_count
Chicken	6
Classic	8
Supreme	9
Veggie	9

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

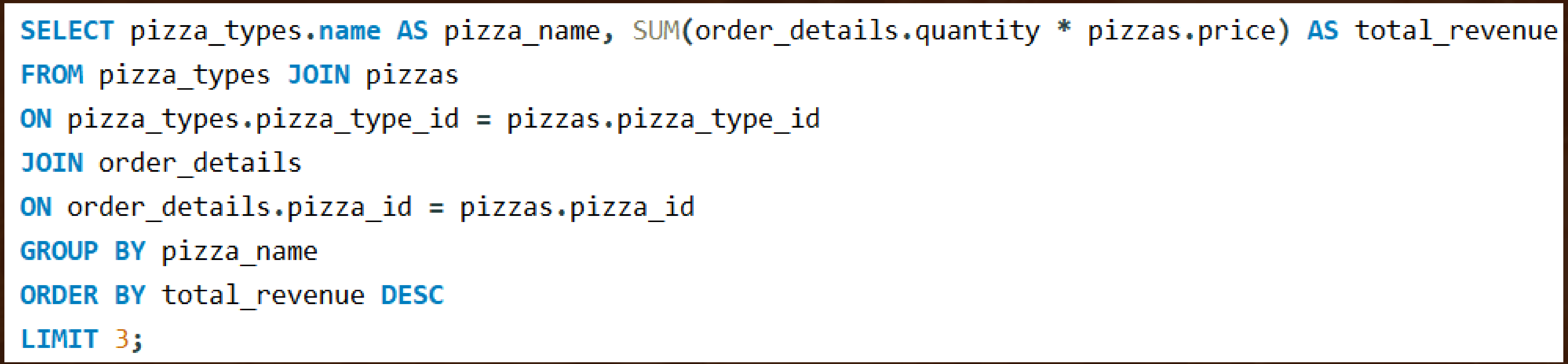


```
SELECT ROUND(AVG(total_quantity)) AS avg_quantity_per_day
FROM
(SELECT orders.order_date, SUM(order_details.quantity) AS total_quantity
FROM orders JOIN order_details
ON orders.order_id = order_details.order_id
GROUP BY orders.order_date) AS order_quantity;
```

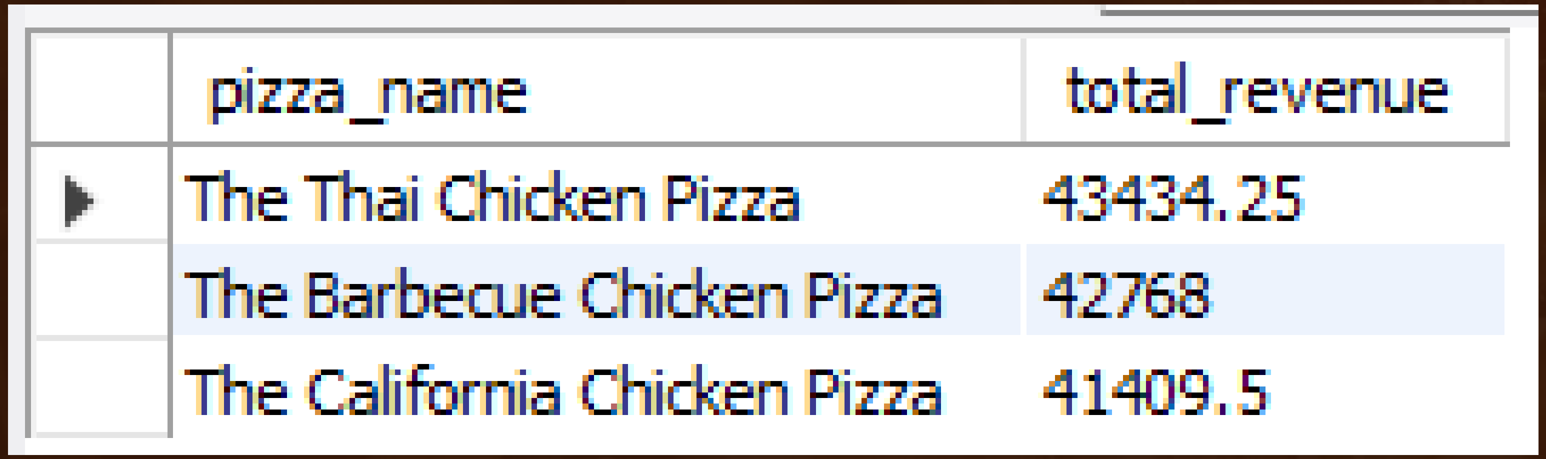


	avg_quantity_per_day
▶	138

DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE

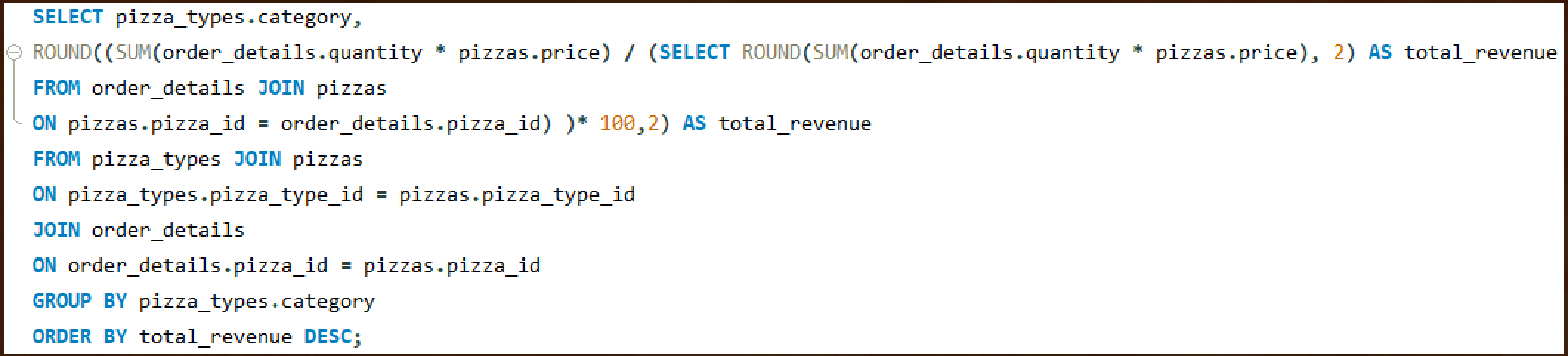


```sql
SELECT pizza_types.name AS pizza_name, SUM(order_details.quantity * pizzas.price) AS total_revenue
FROM pizza_types JOIN pizzas
ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN order_details
ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_name
ORDER BY total_revenue DESC
LIMIT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

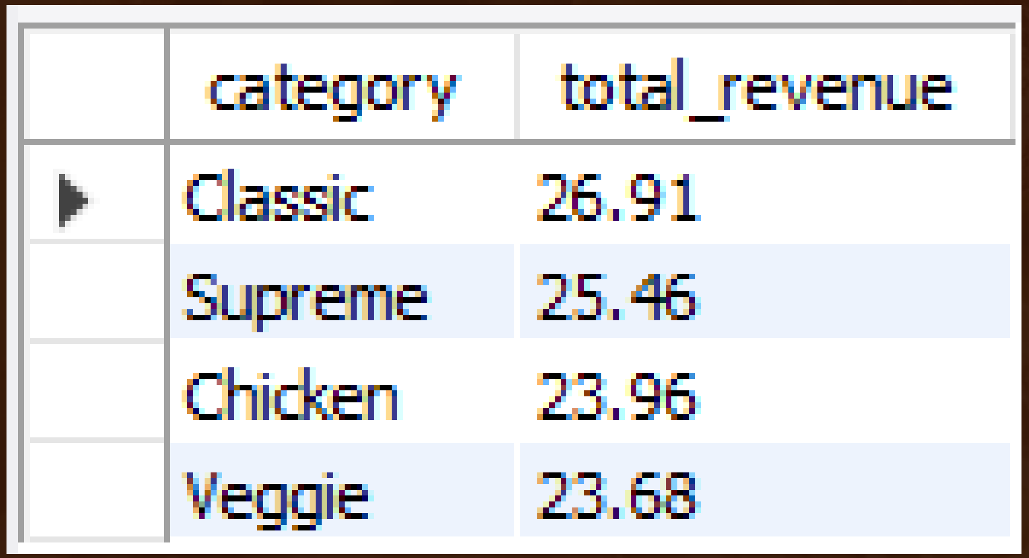


pizza_name	total_revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE

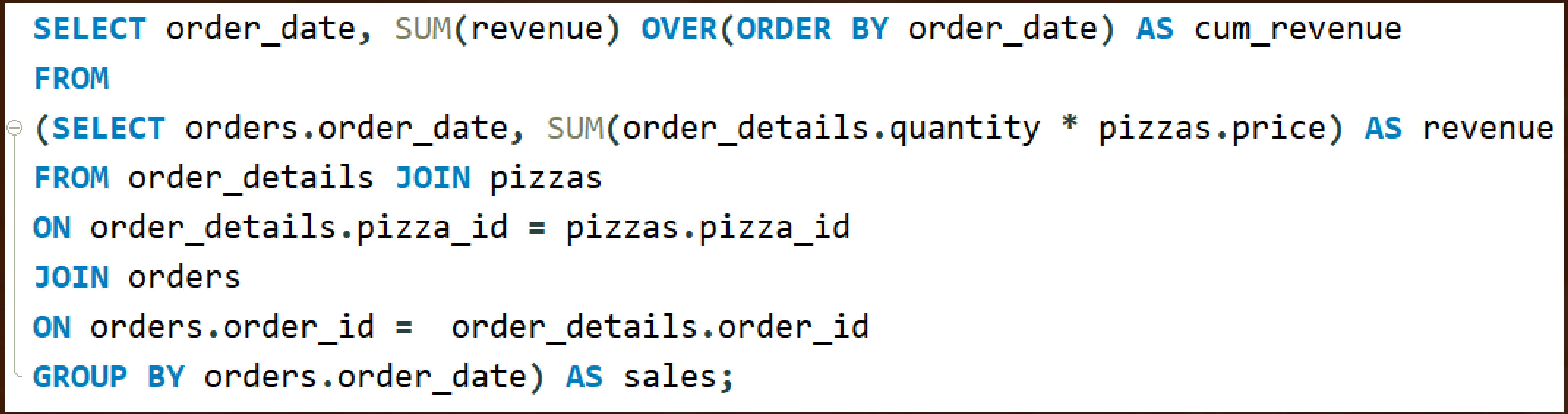


```
SELECT pizza_types.category,
ROUND((SUM(order_details.quantity * pizzas.price) / (SELECT ROUND(SUM(order_details.quantity * pizzas.price), 2) AS total_revenue
FROM order_details JOIN pizzas
ON pizzas.pizza_id = order_details.pizza_id) )* 100,2) AS total_revenue
FROM pizza_types JOIN pizzas
ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN order_details
ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY total_revenue DESC;
```

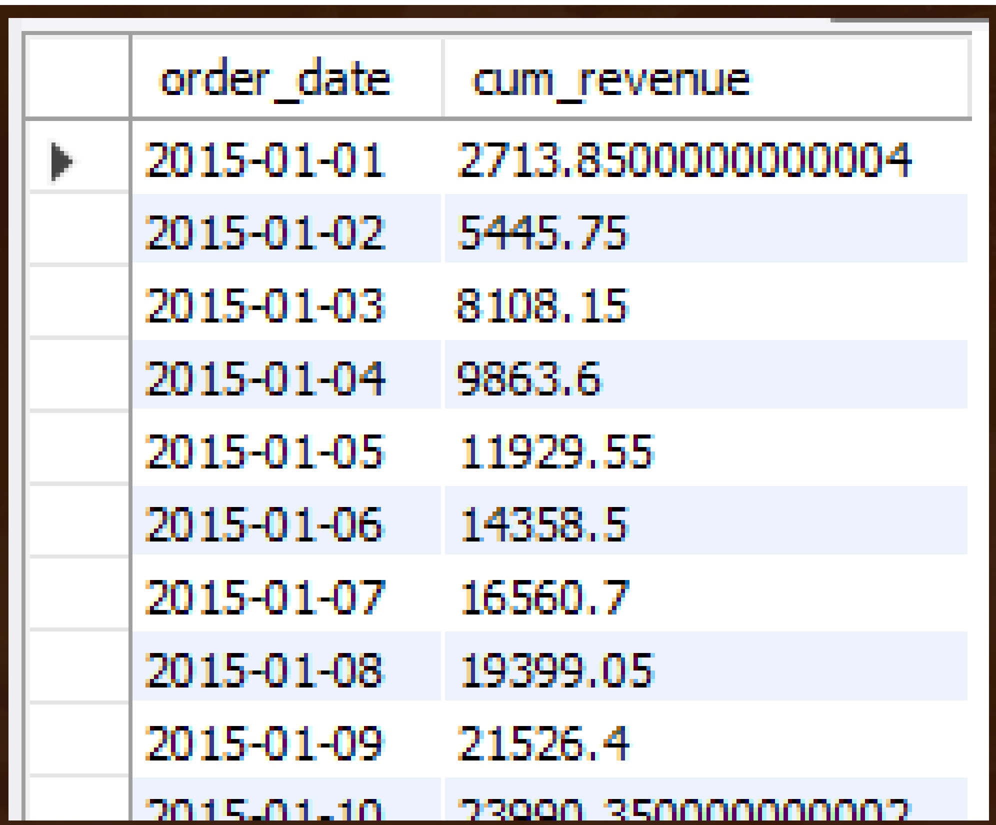


category	total_revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME



```sql
SELECT order_date, SUM(revenue) OVER(ORDER BY order_date) AS cum_revenue
FROM
(SELECT orders.order_date, SUM(order_details.quantity * pizzas.price) AS revenue
FROM order_details JOIN pizzas
ON order_details.pizza_id = pizzas.pizza_id
JOIN orders
ON orders.order_id =  order_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
2015-01-10	23990.350000000002

La Pizzeria Sales Analysis Presentation

THANK YOU FOR ATTENTION

Presented by Shivani K