‘Pizza Sales’ Case Study

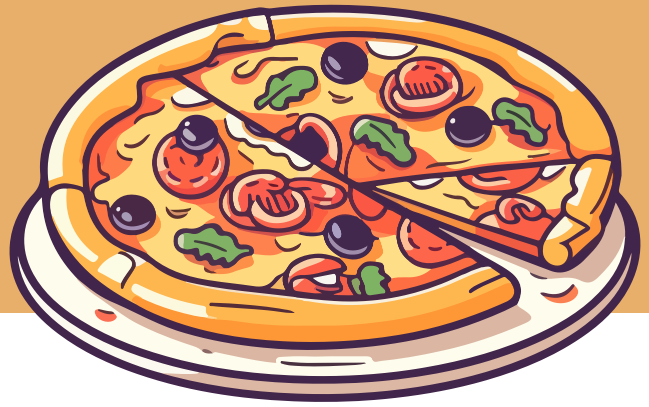
Using SQL

Case study from WsCube Tech

Made by:
Adamya Shula

Solved using SQL

OVERVIEW



This project analyzes pizza sales data using SQL to answer real-world business questions.

The goal was to explore sales performance, customer behavior, and product trends to provide insights that can help improve business decisions.
The dataset contains information about orders, pizzas, customers, and transactions.

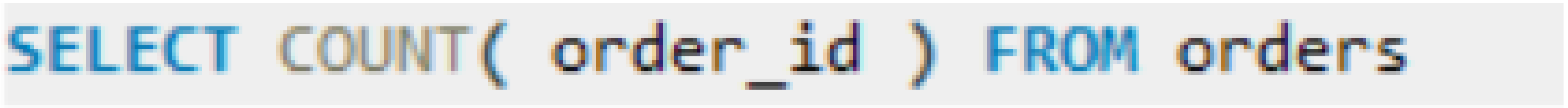
Using SQL, I wrote queries to calculate key metrics such as:

- Total revenue generated
- Best-selling pizzas and categories
- Average order value
- Top spending customers
- Sales trends over time

By turning raw sales data into insights, this project demonstrates how SQL can be used not only for querying databases but also for making data-driven business recommendations.

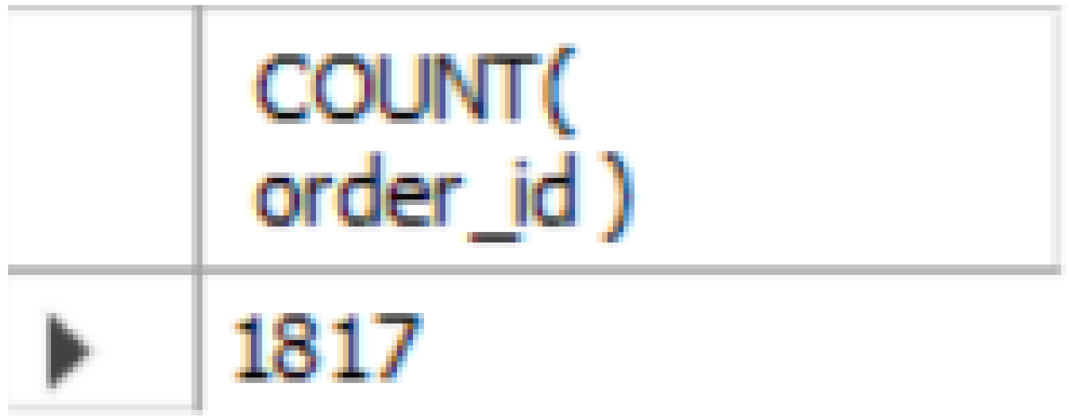
Retrieve the total number of orders placed

Query:

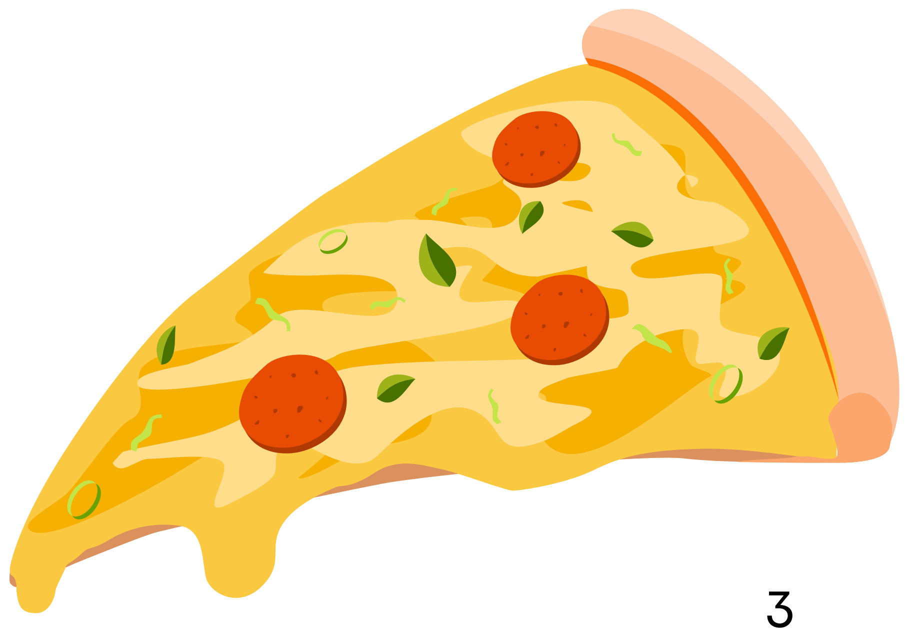


```
SELECT COUNT( order_id ) FROM orders
```

Solution:

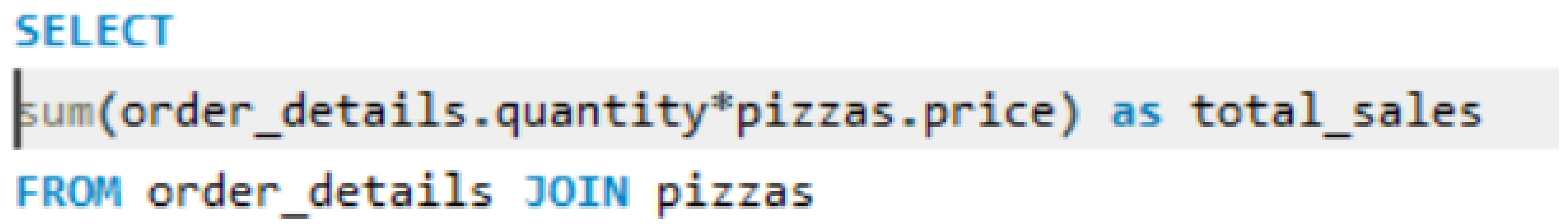


	COUNT(order_id)
▶	1817



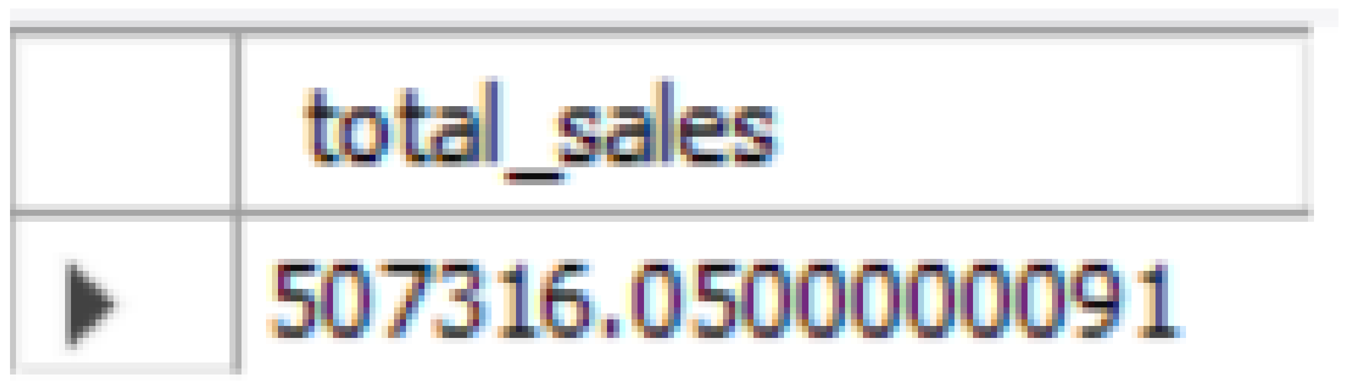
Calculate the total revenue generated from pizza sales.

Query:



```sql
SELECT
sum(order_details.quantity*pizzas.price) as total_sales
FROM order_details JOIN pizzas
```

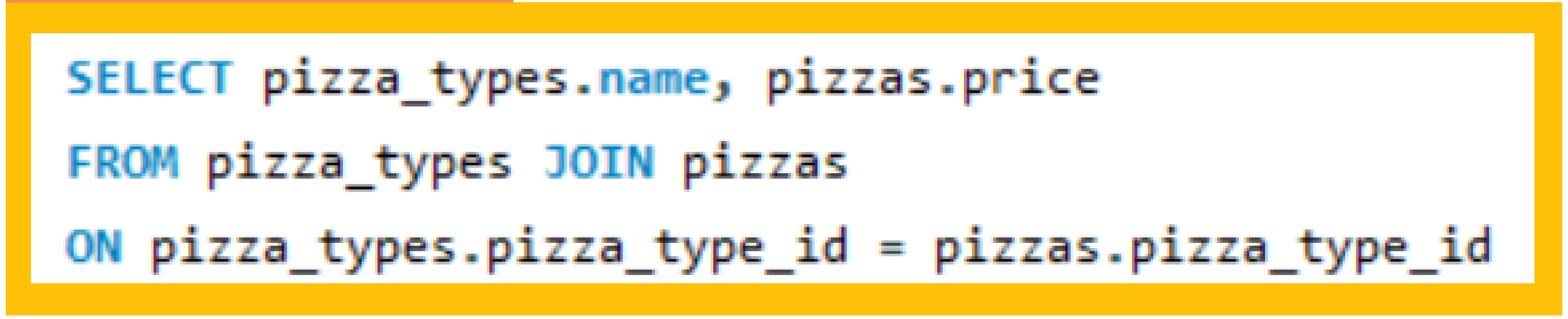
Solution:



	total_sales
▶	507316.0500000091

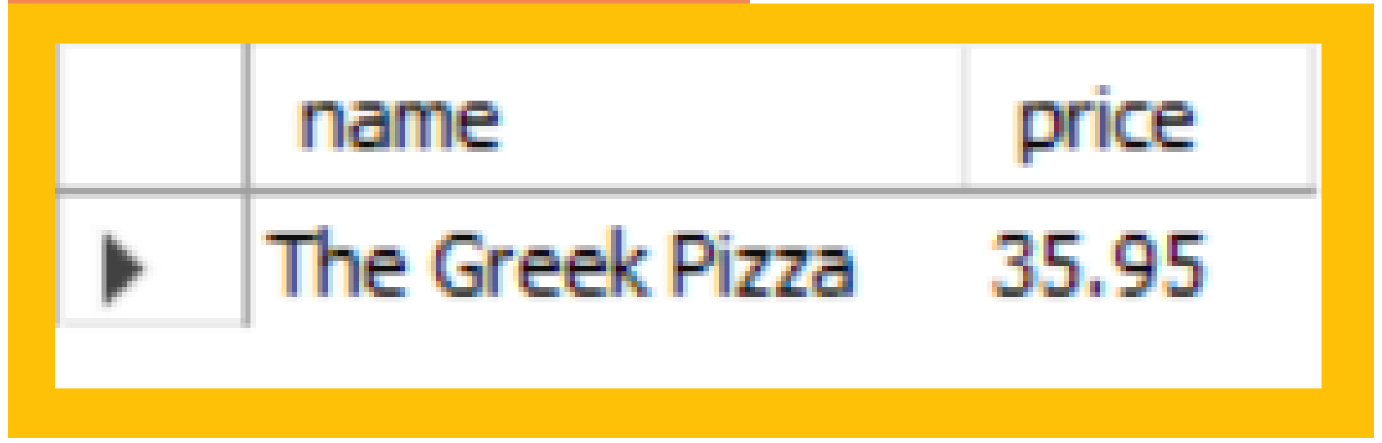
Identify the highest-priced pizza

Query:



```
SELECT pizza_types.name, pizzas.price
FROM pizza_types JOIN pizzas
ON pizza_types.pizza_type_id = pizzas.pizza_type_id
```

Solution:



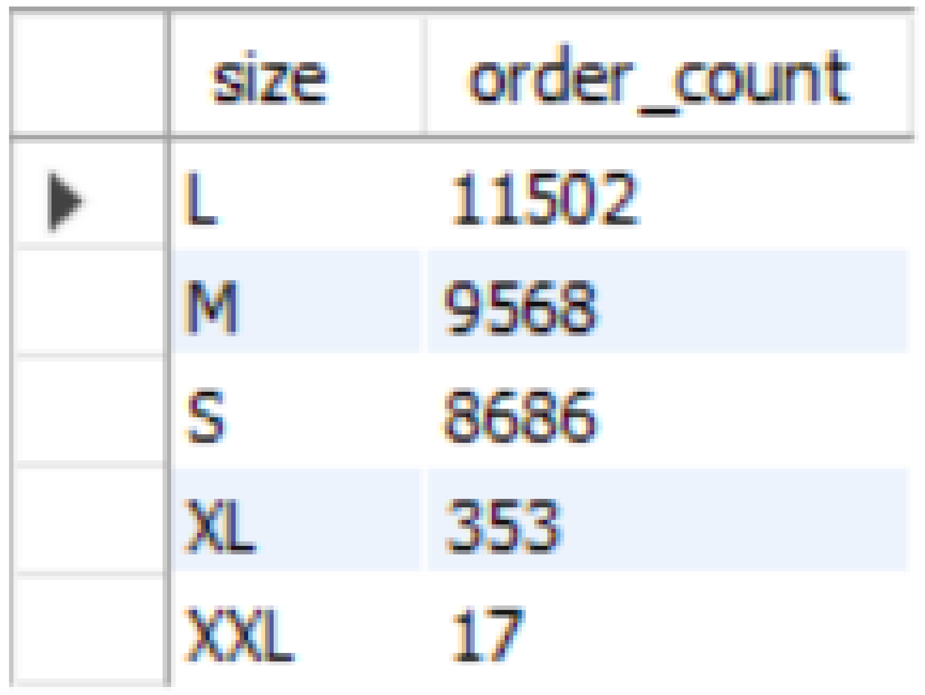
	name	price
▶	The Greek Pizza	35.95

Identify the most common pizza size ordered

Query:

```
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 ORDER BY order_count desc;
```

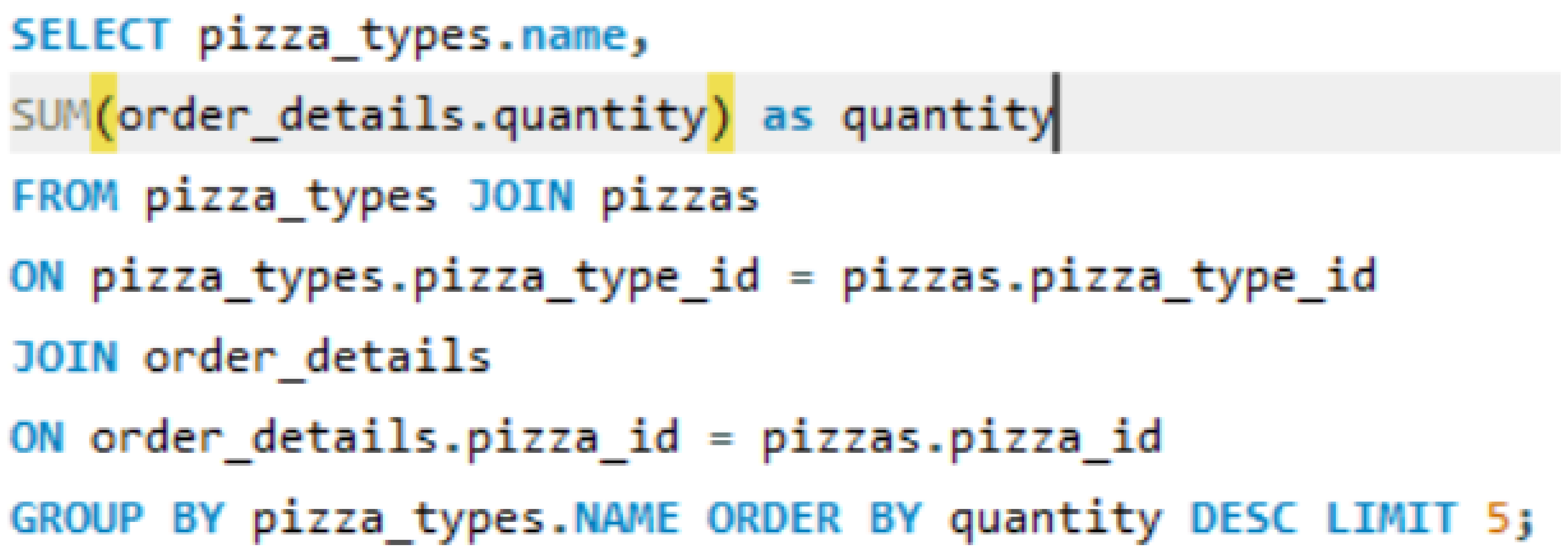
Solution:



size	order_count
L	11502
M	9568
S	8686
XL	353
XXL	17

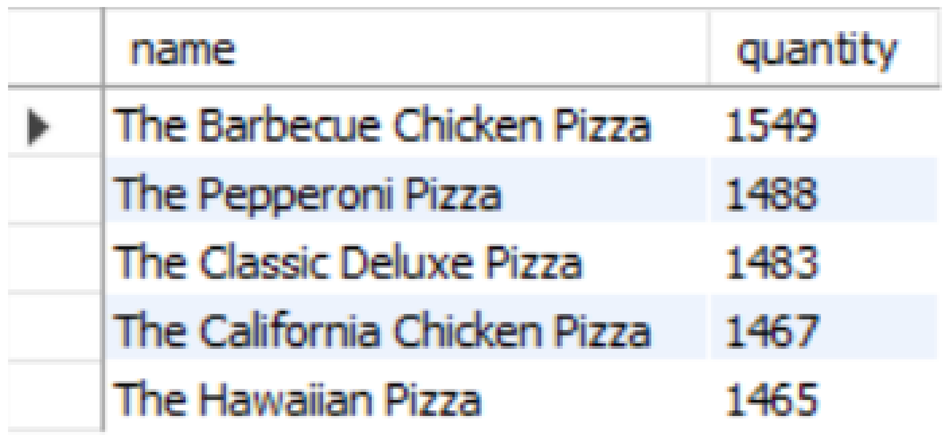
List the top 5 most ordered pizza types along with their quantities

Query:

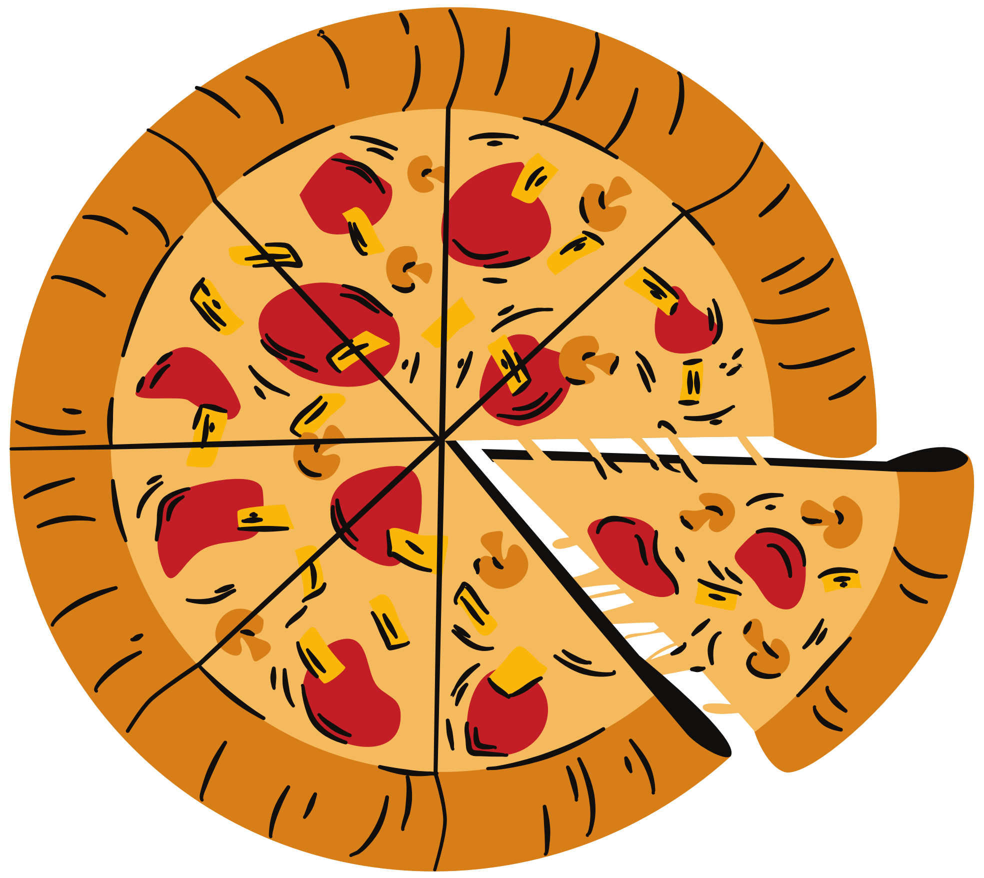


```
SELECT pizza_types.name,
SUM(order_details.quantity) as quantity
FROM pizza_types JOIN pizzas
ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN order_details
ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.NAME ORDER BY quantity DESC LIMIT 5;
```

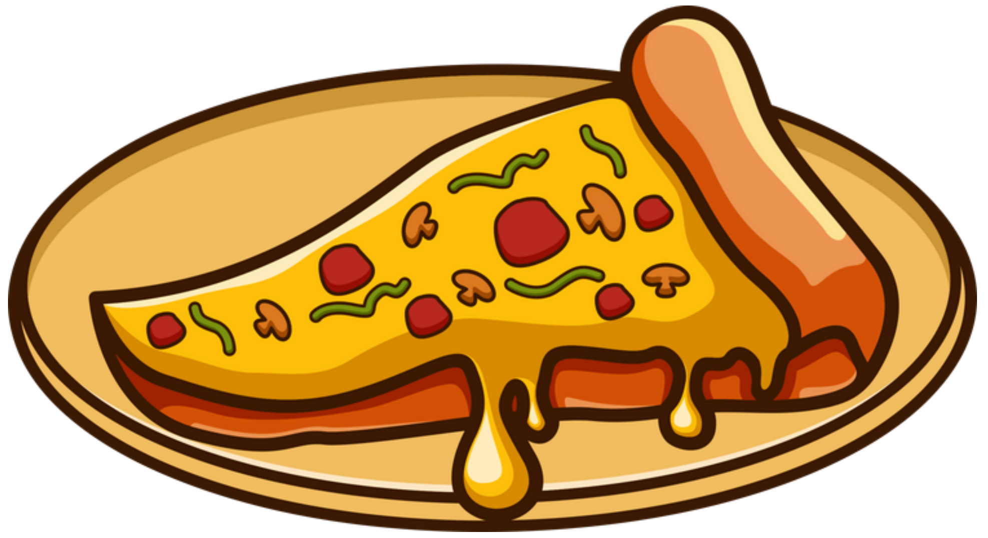
Solution:

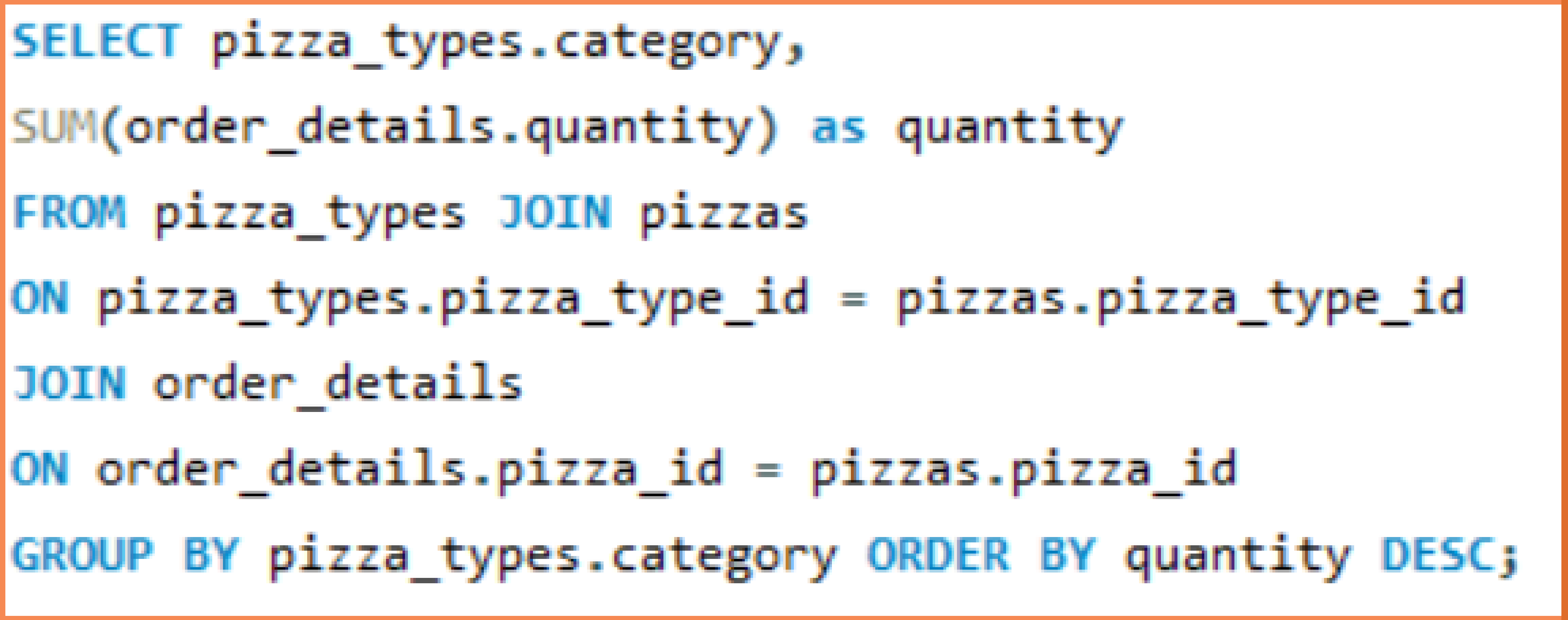


name	quantity
The Barbecue Chicken Pizza	1549
The Pepperoni Pizza	1488
The Classic Deluxe Pizza	1483
The California Chicken Pizza	1467
The Hawaiian Pizza	1465



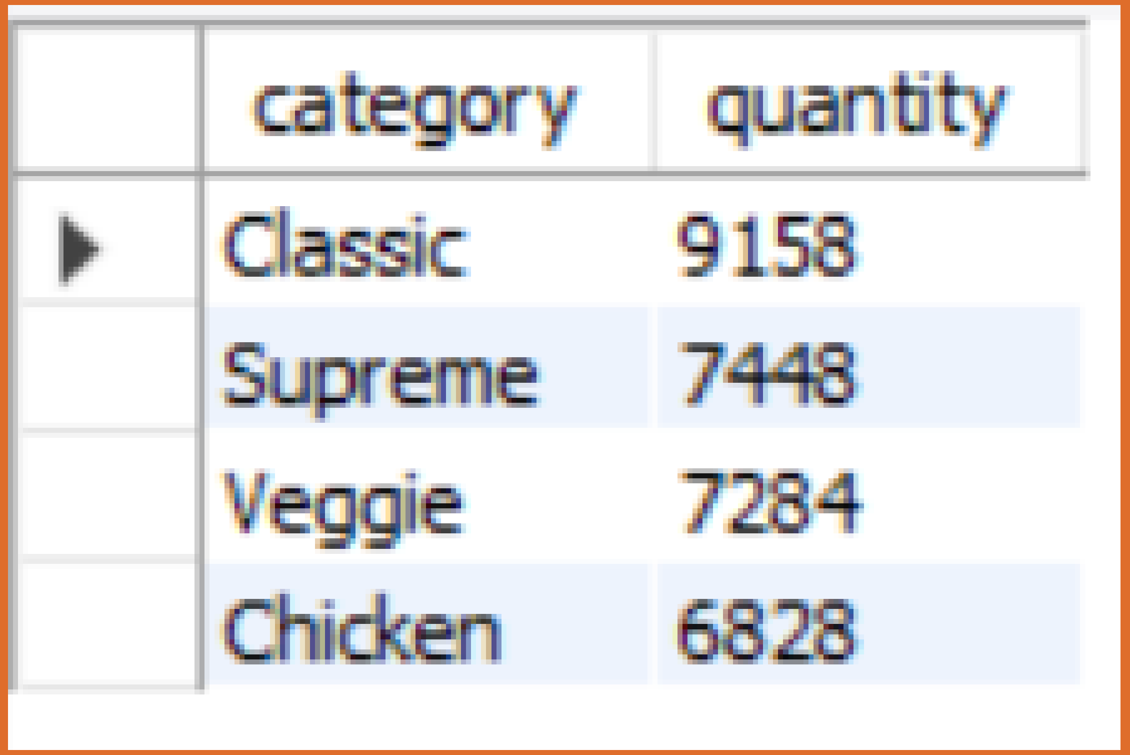
Join the necessary tables to find the total quantity of each pizza category ordered





```
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;
```

Query:

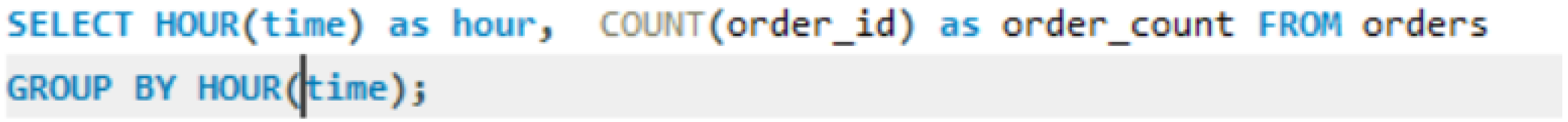


category	quantity
Classic	9158
Supreme	7448
Veggie	7284
Chicken	6828

Solution:

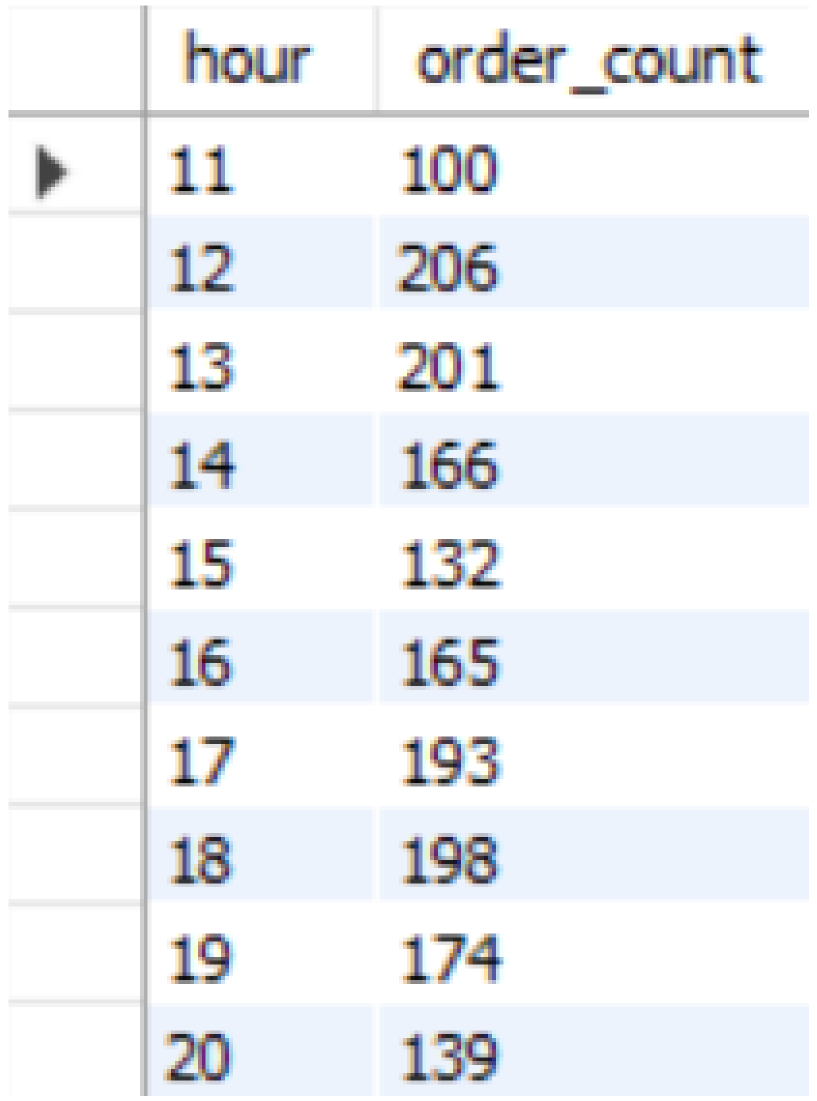
Determine the distribution of orders by hour of the day

Query:

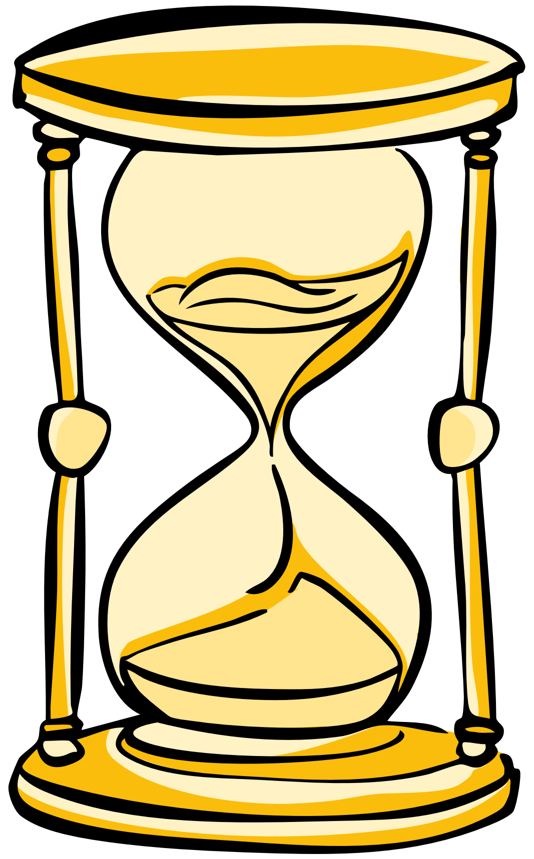


```
SELECT HOUR(time) as hour,  COUNT(order_id) as order_count FROM orders
GROUP BY HOUR(time);
```

Solution:

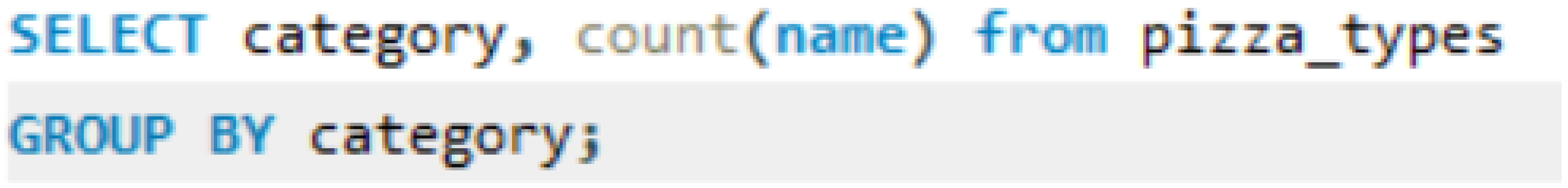


hour	order_count
11	100
12	206
13	201
14	166
15	132
16	165
17	193
18	198
19	174
20	139



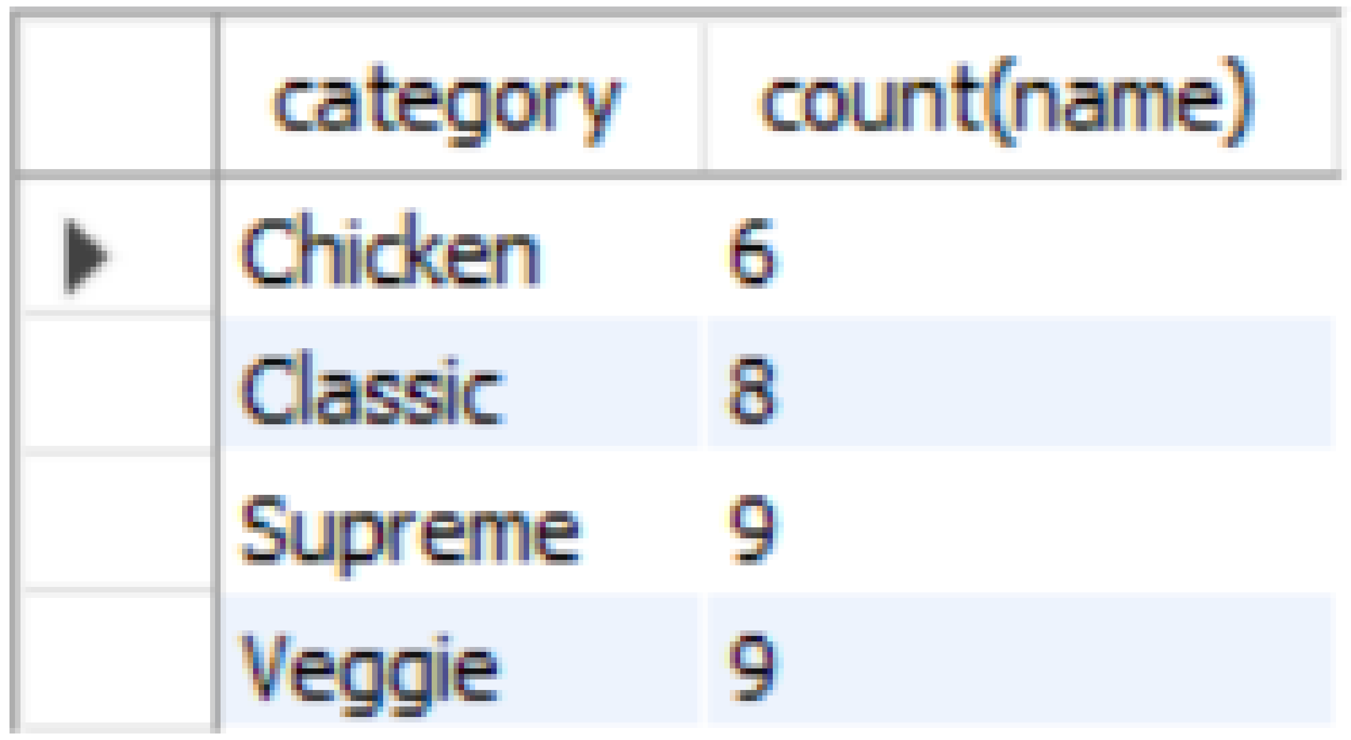
Join relevant tables to find the category-wise distribution of pizzas

Query:



```
SELECT category, count(name) from pizza_types
GROUP BY category;
```

Solution:



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

Recommendations

Promotions on Fridays → Target customers with discounts or bundles to maximize high-traffic days.

Focus on 'Large' size Pizzas → Create family-size combo deals since 'Large' generates most revenue

Customer Loyalty Program → Reward top spenders to improve retention.

Category Optimization → Expand Classic pizza options since they dominate sales.