FINAL DATA ANALYST PROJECT 2025

PIZZA SALES ANALYSIS USING SQL

PRESENTED BY: D S HARSHIT | TOOLS: MYSQL WORKBENCH

PROJECT OBJECTIVE

- Use MySQL to explore and analyze pizza sales data.
- Answer real-world business questions from basic to advanced.
- Gain insights into revenue, top products, customer behavior, and sales patterns.

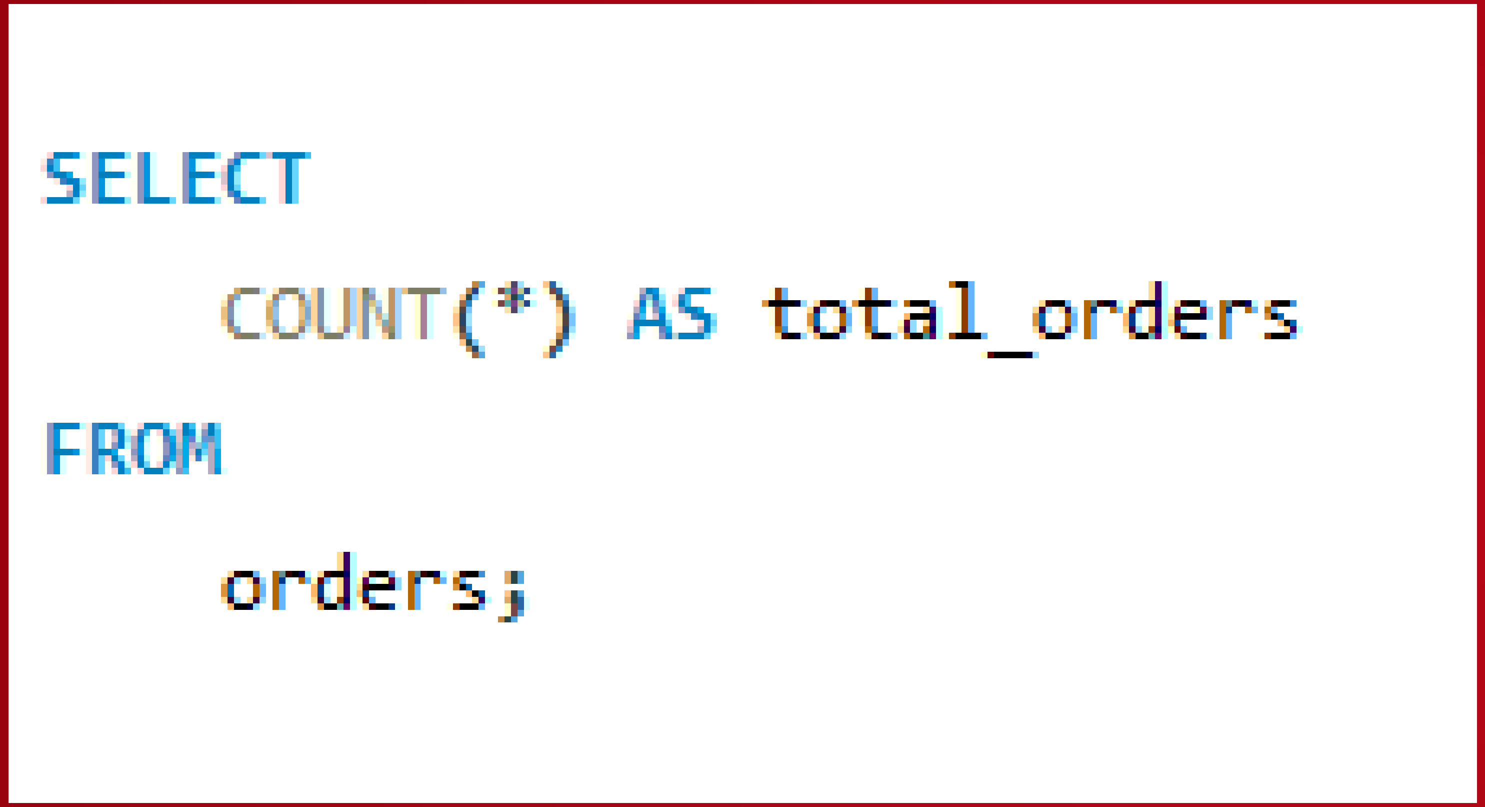
DATASET COLUMNS

TABLE	COLUMNS
Orders	order_id, order_date, order_time
OrderDetails	order_details_id, order_id, pizza_id, quantity
Pizzas	pizza_id, pizza_type_id, size, price
PizzaTypes	pizza_type_id, name, category, ingredients

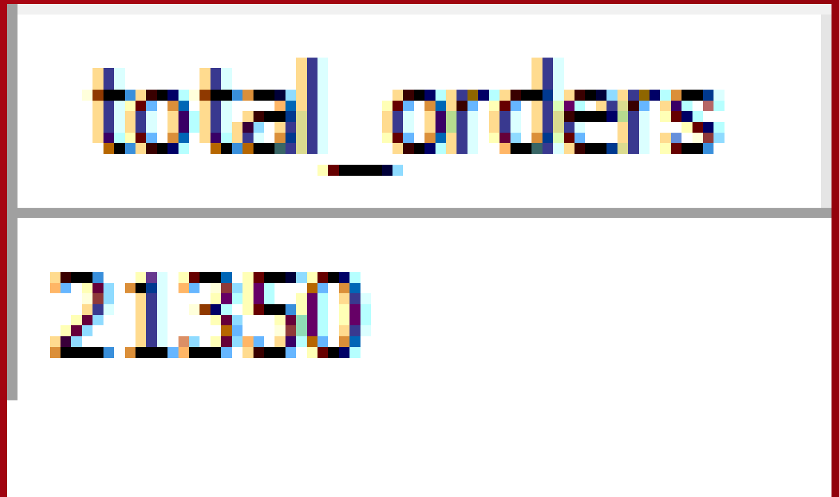
SQL QUESTIONS SOLVED

Basics	5
Intermediate	5
Advanced	3
Total	**13**

RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

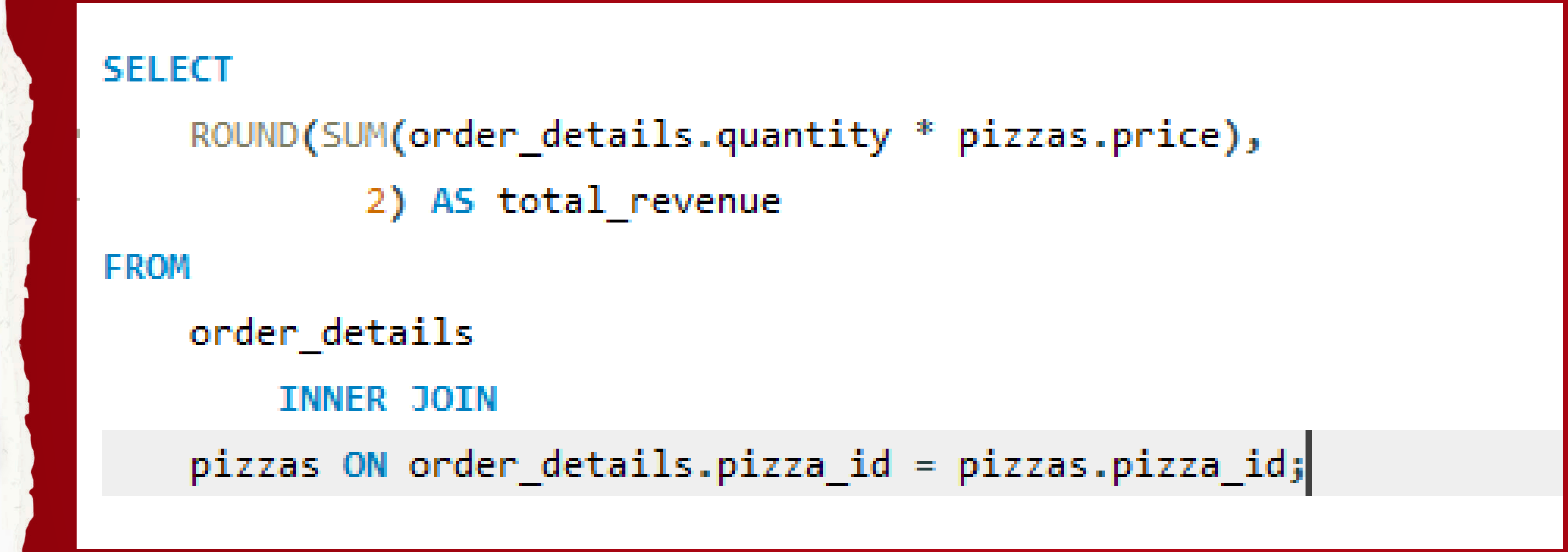


```
SELECT
    COUNT(*) AS total_orders
FROM
    orders;
```

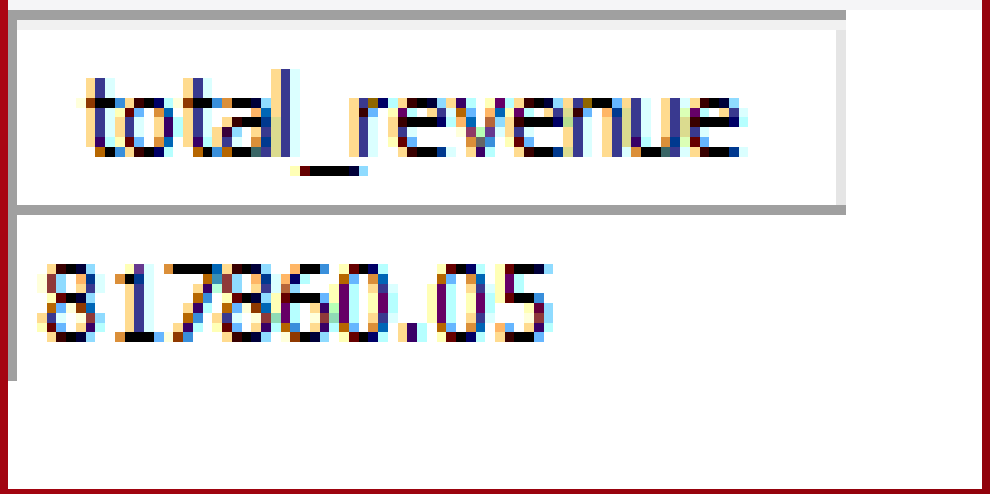


total_orders
21350

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

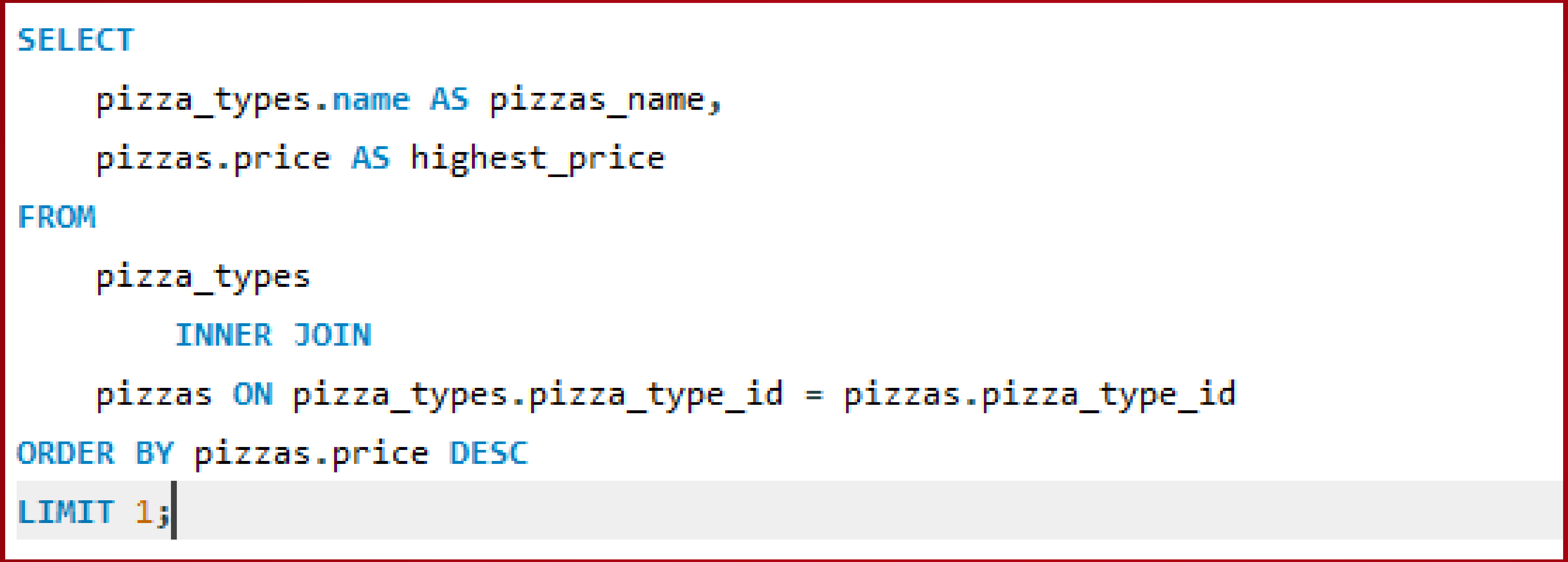


```sql
SELECT
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS total_revenue
FROM
    order_details
        INNER JOIN
    pizzas ON order_details.pizza_id = pizzas.pizza_id;
```

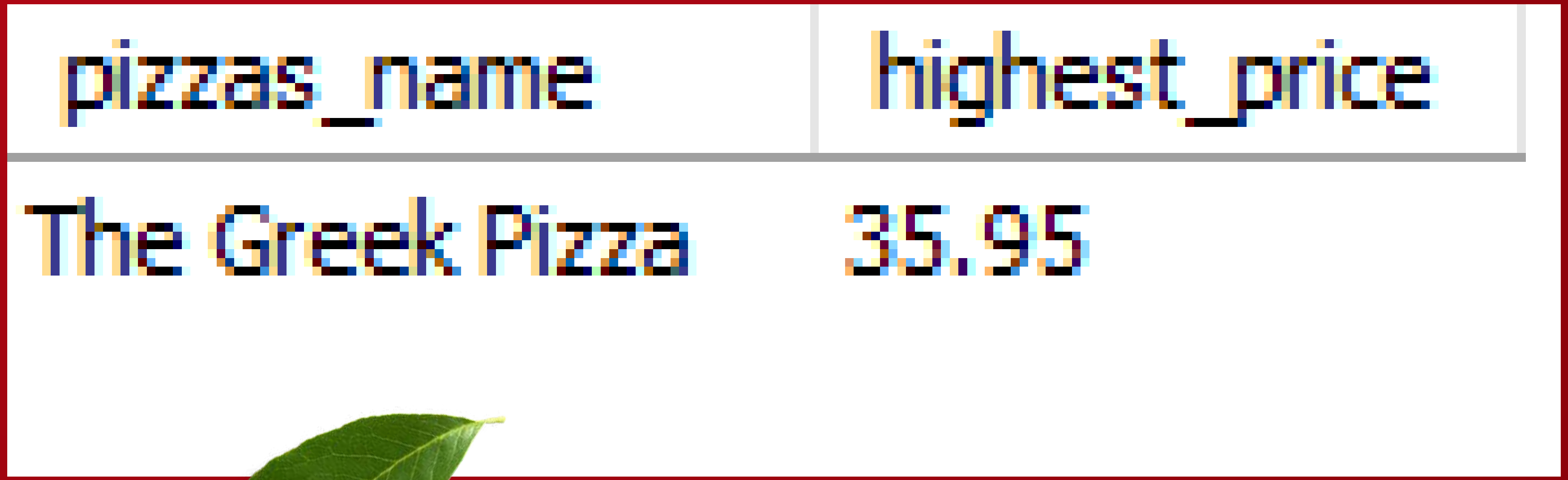


total_revenue
817860.05

IDENTIFY THE HIGHEST-PRICED PIZZA.

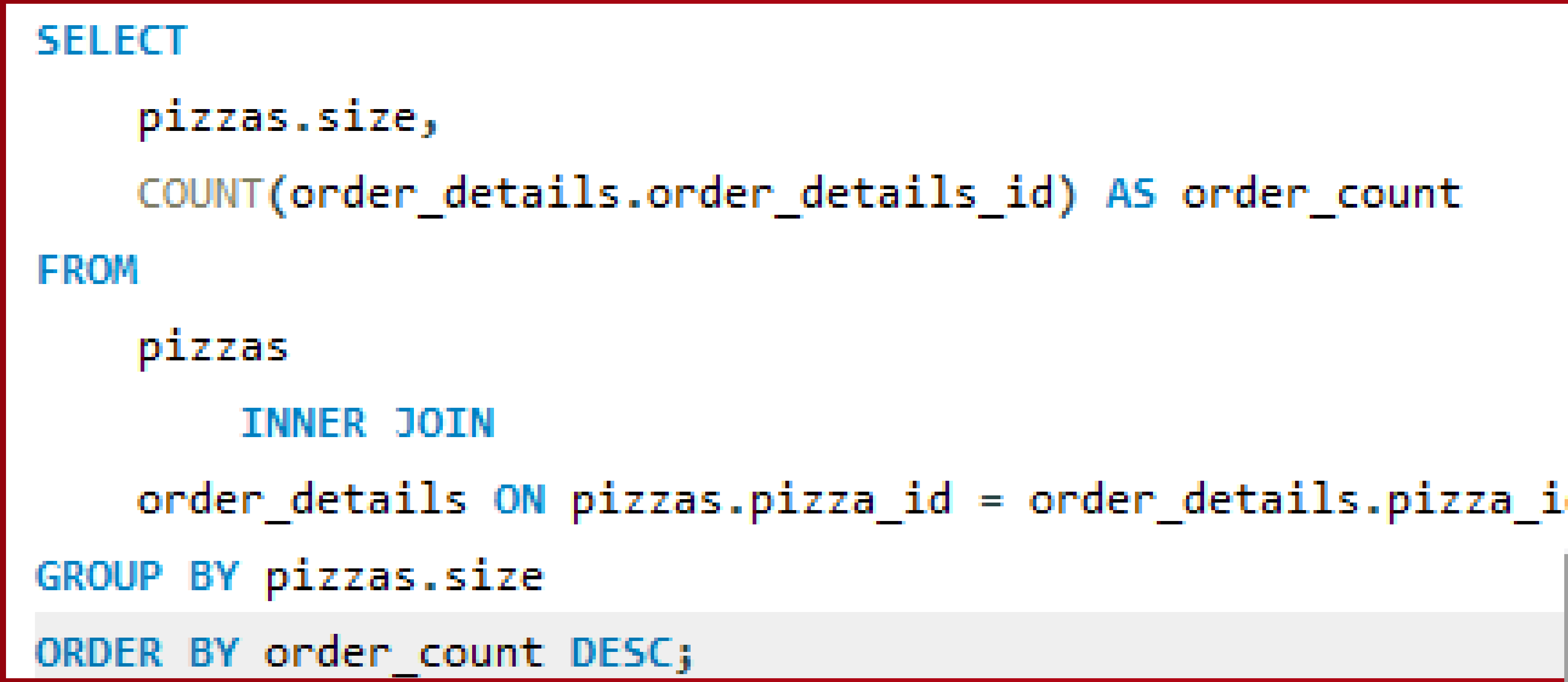


```sql
SELECT
    pizza_types.name AS pizzas_name,
    pizzas.price AS highest_price
FROM
    pizza_types
        INNER JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
```

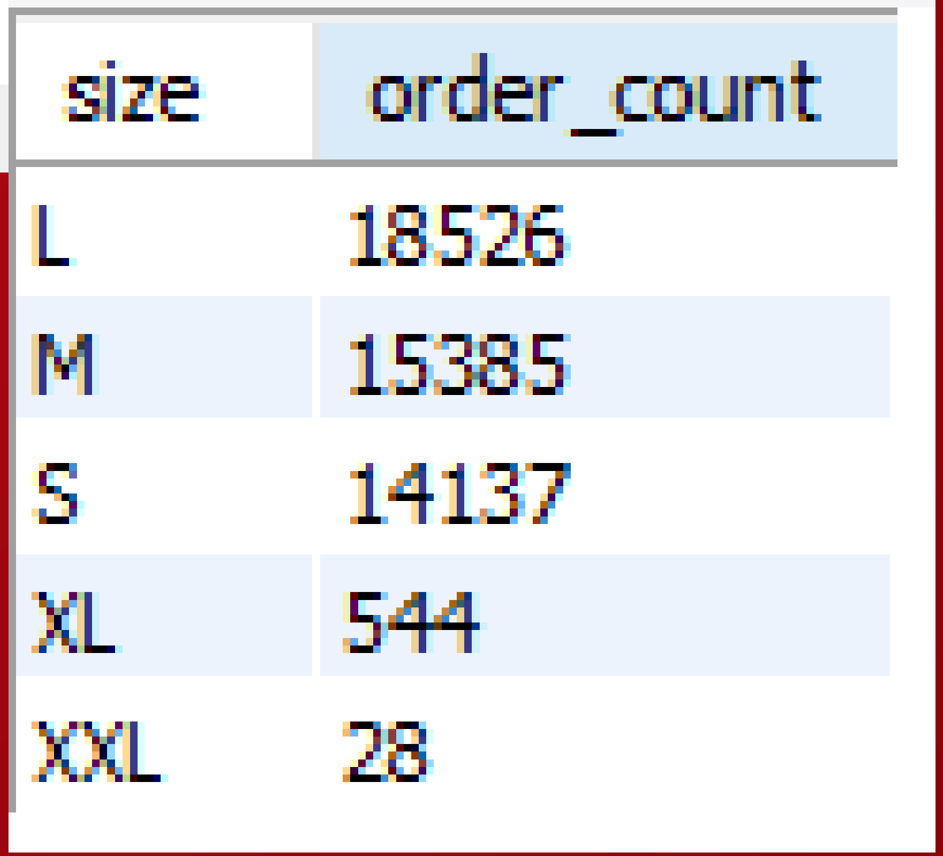


pizzas_name	highest_price
The Greek Pizza	35.95

IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

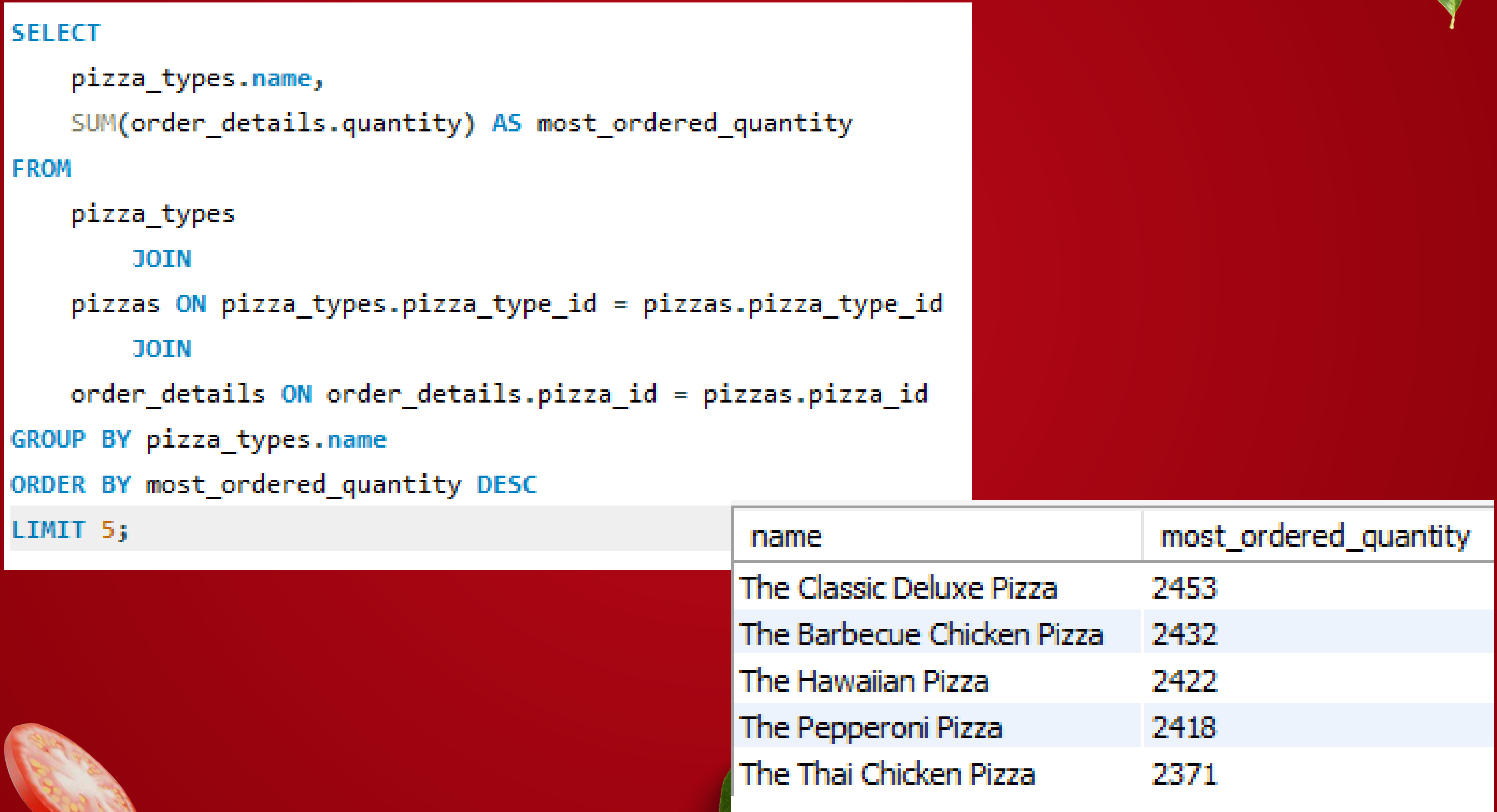


```
SELECT
    pizzas.size,
    COUNT(order_details.order_details_id) AS order_count
FROM
    pizzas
        INNER JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC;
```



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

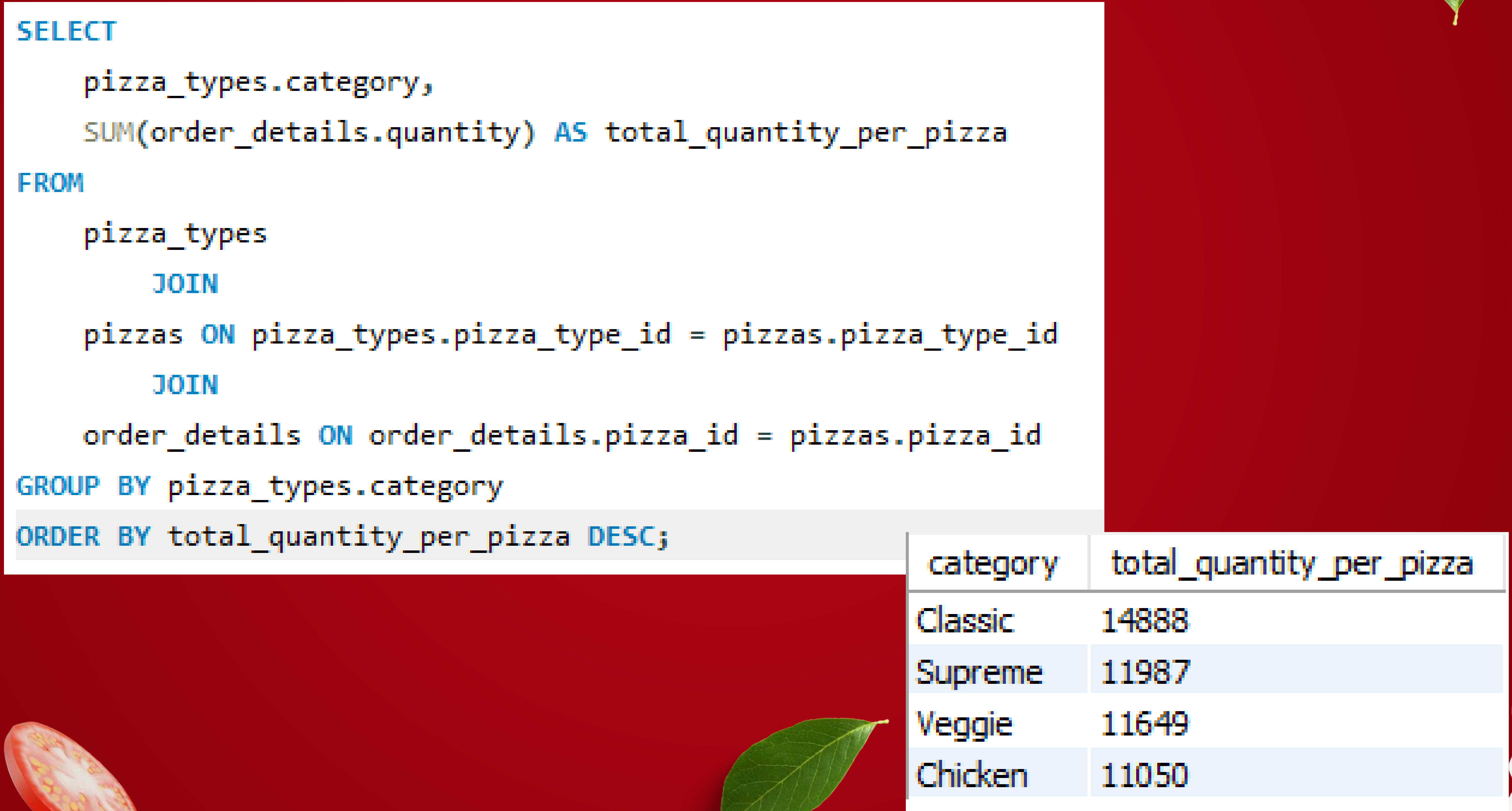
LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.



```
SELECT
    pizza_types.name,
    SUM(order_details.quantity) AS most_ordered_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 most_ordered_quantity DESC
LIMIT 5;
```

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

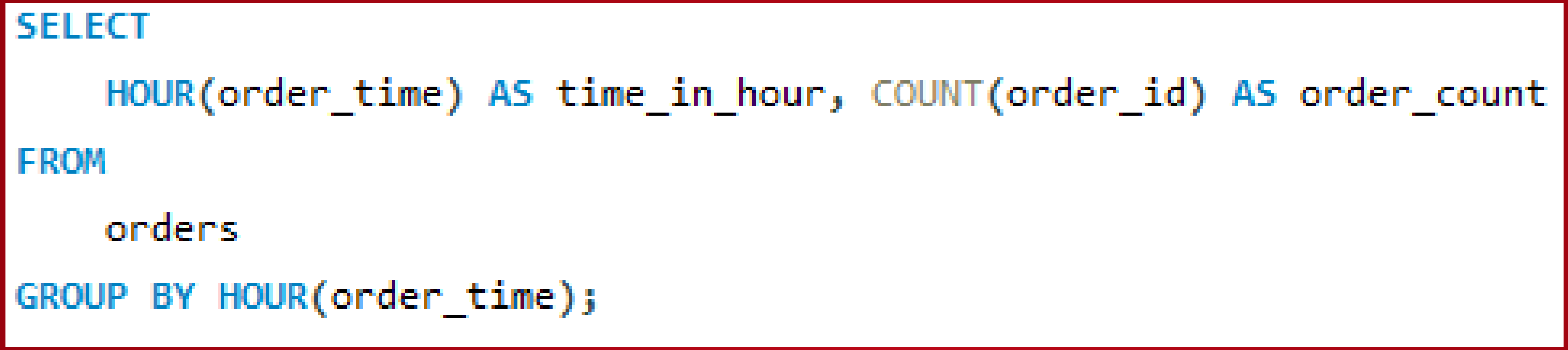
JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.



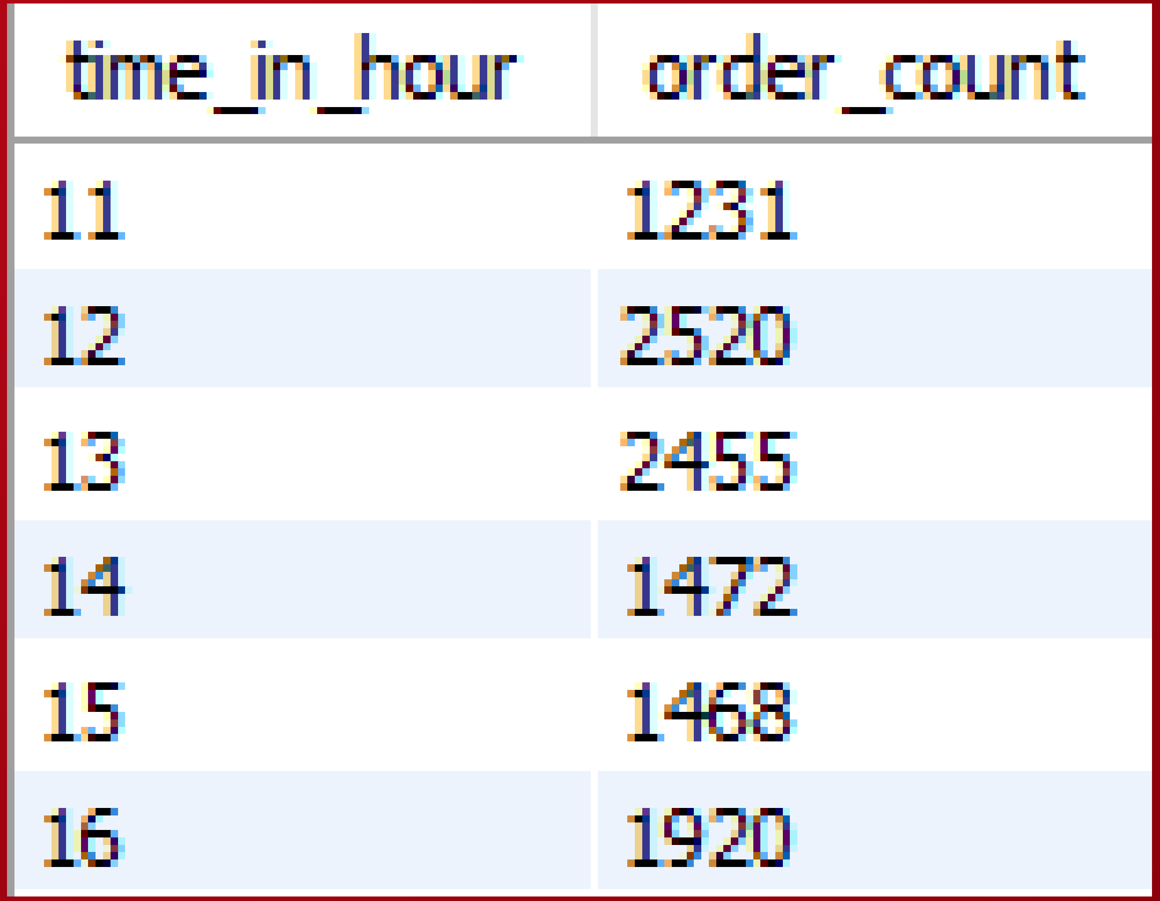
```
SELECT
    pizza_types.category,
    SUM(order_details.quantity) AS total_quantity_per_pizza
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER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

category	total_quantity_per_pizza
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

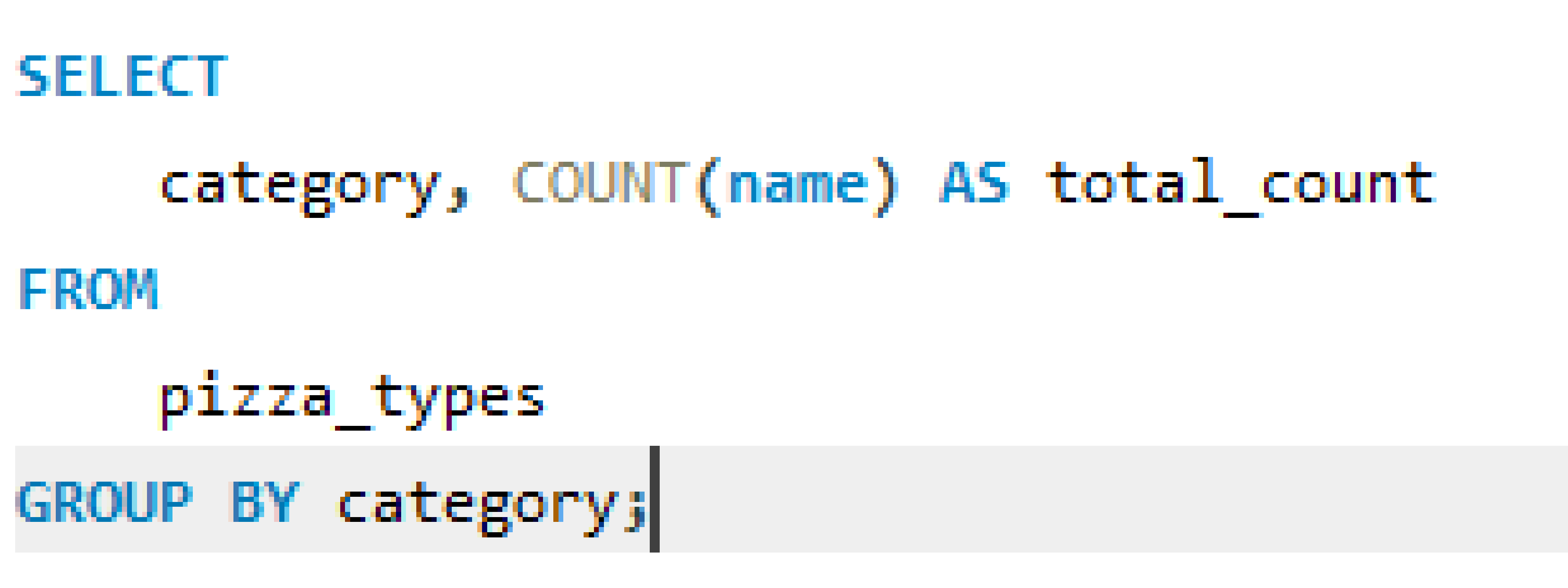


```
SELECT
    HOUR(order_time) AS time_in_hour, COUNT(order_id) AS order_count
FROM
    orders
GROUP BY HOUR(order_time);
```

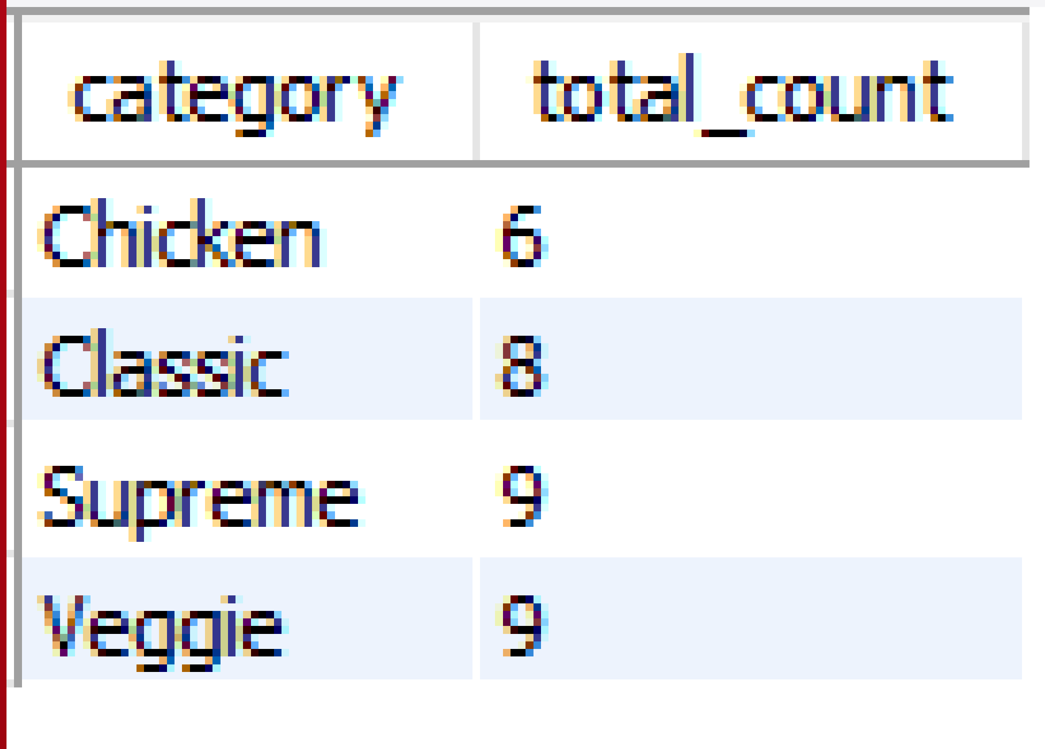


time_in_hour	order_count
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920

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

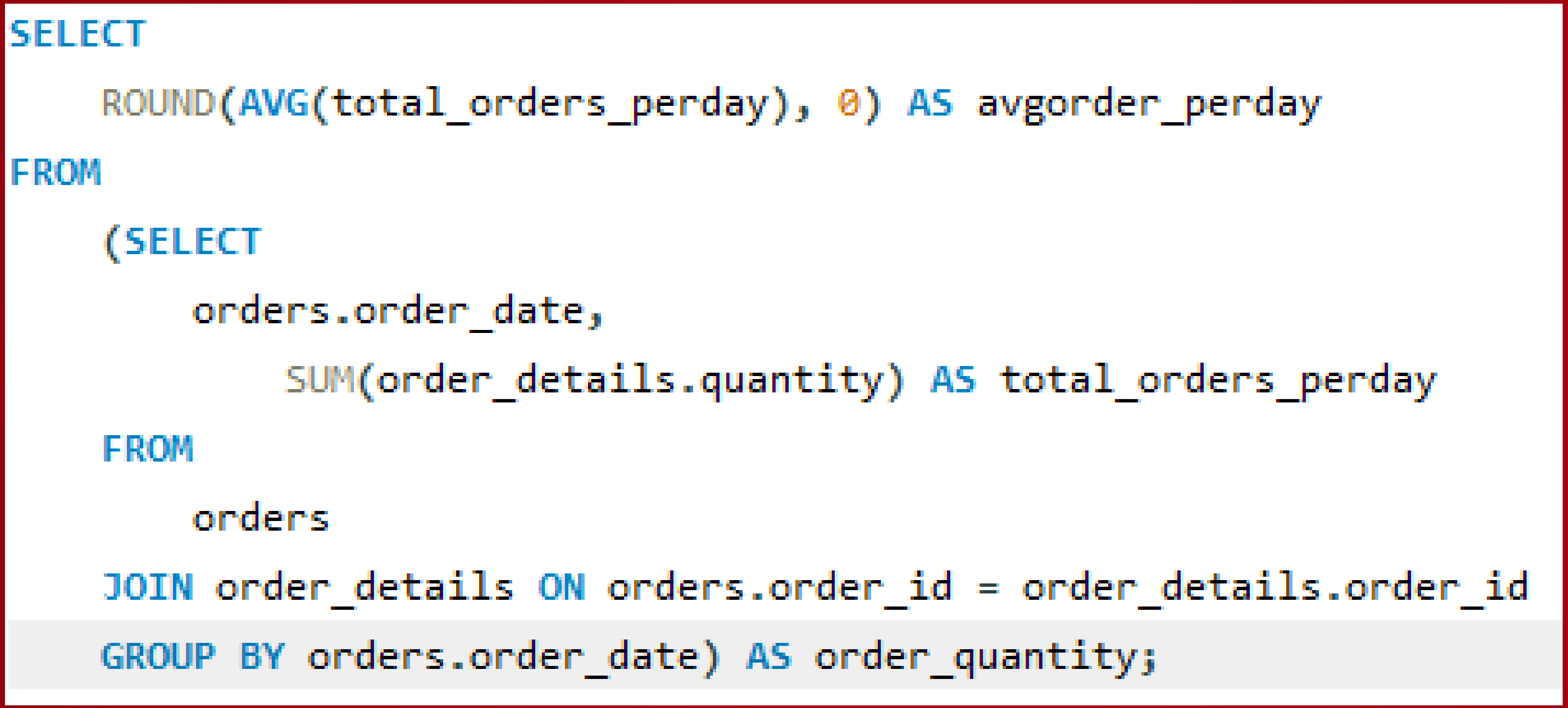


```
SELECT
    category, COUNT(name) AS total_count
FROM
    pizza_types
GROUP BY category;
```

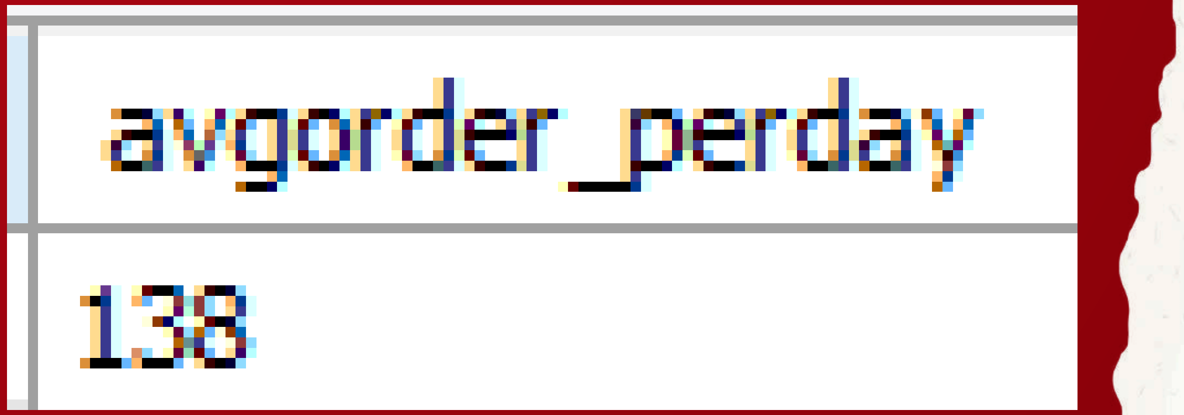


category	total_count
Chicken	6
Classic	8
Supreme	9
Veggie	9

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

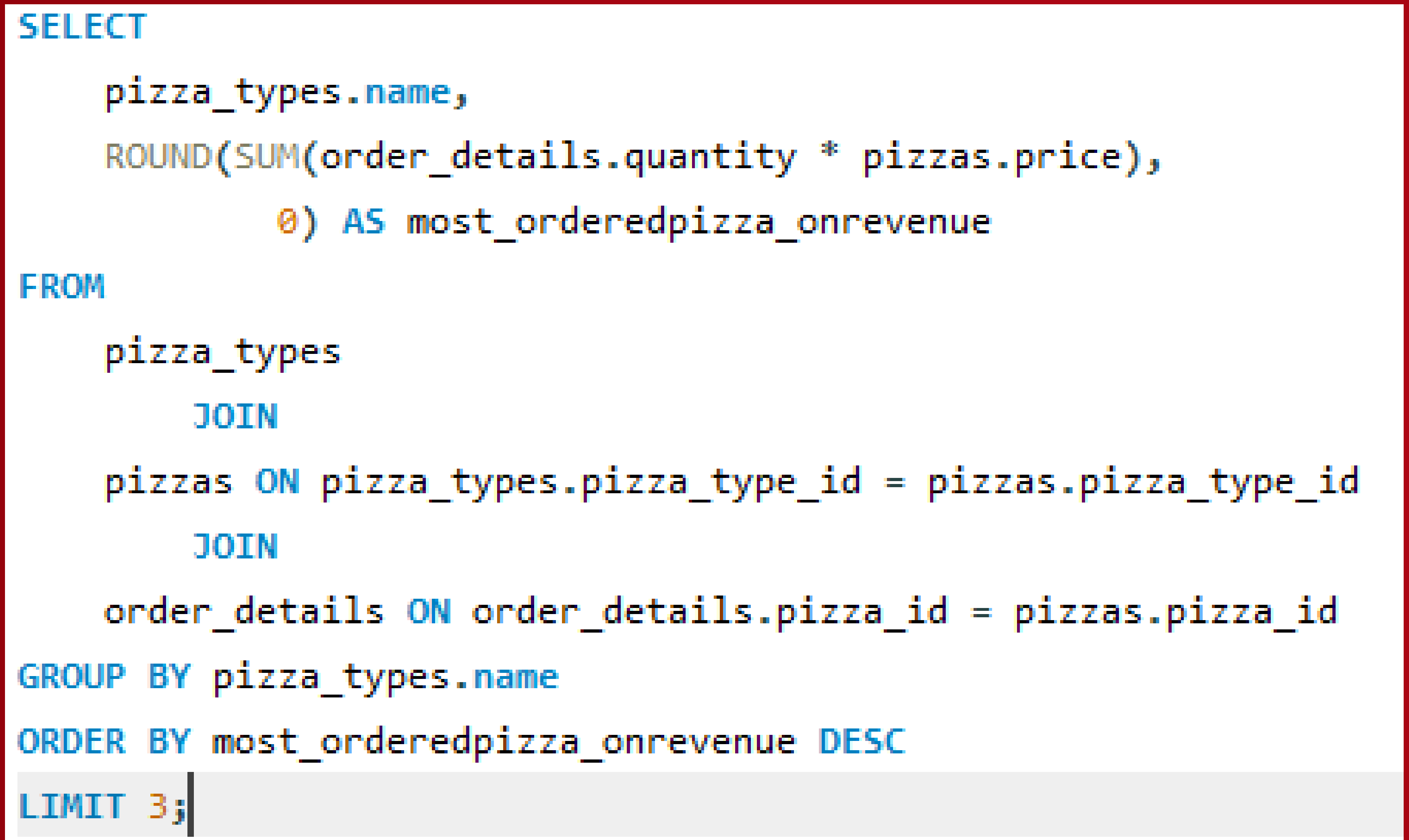


```
SELECT
    ROUND(AVG(total_orders_perday), 0) AS avgorder_perday
FROM
    (SELECT
        orders.order_date,
            SUM(order_details.quantity) AS total_orders_perday
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY orders.order_date) AS order_quantity;
```

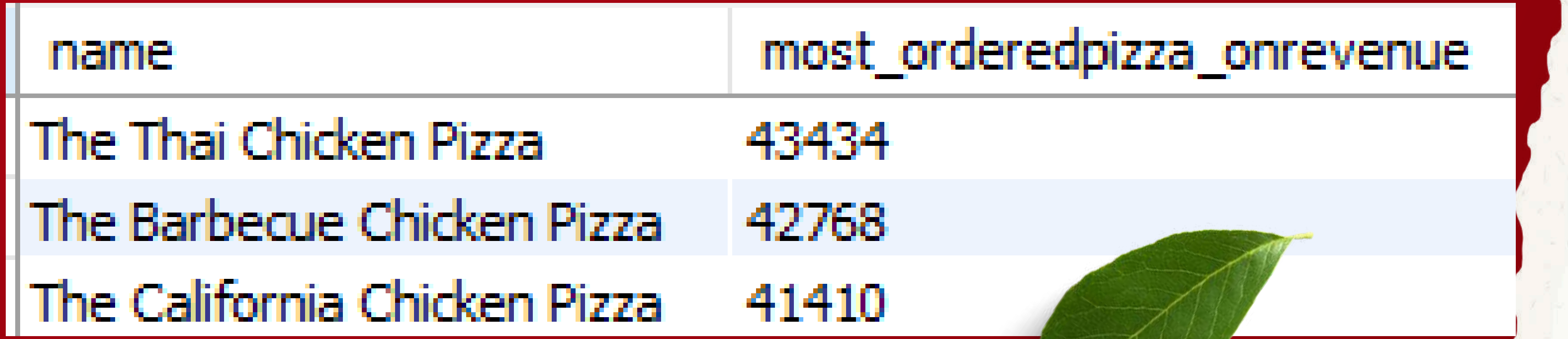


avgorder_perday
138

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

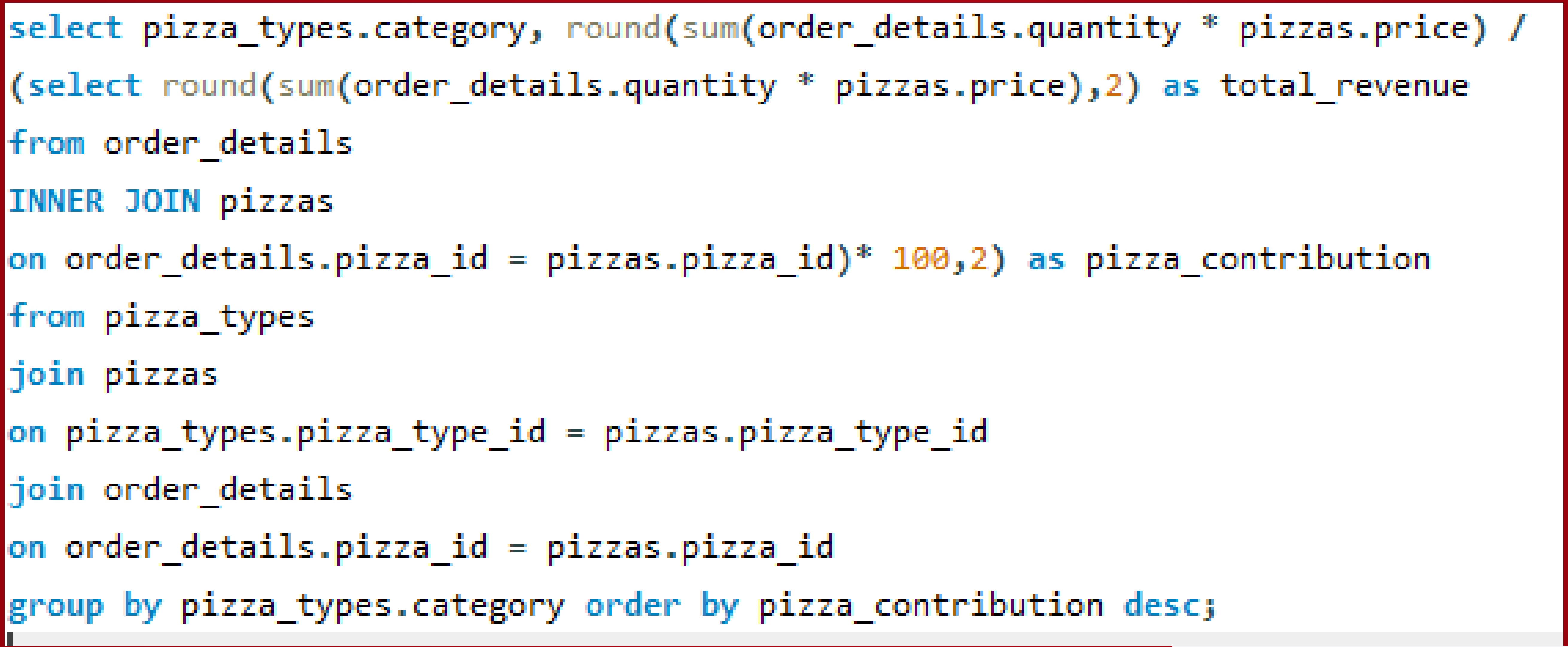


```
SELECT
    pizza_types.name,
    ROUND(SUM(order_details.quantity * pizzas.price),
            0) AS most_orderedpizza_onrevenue
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id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

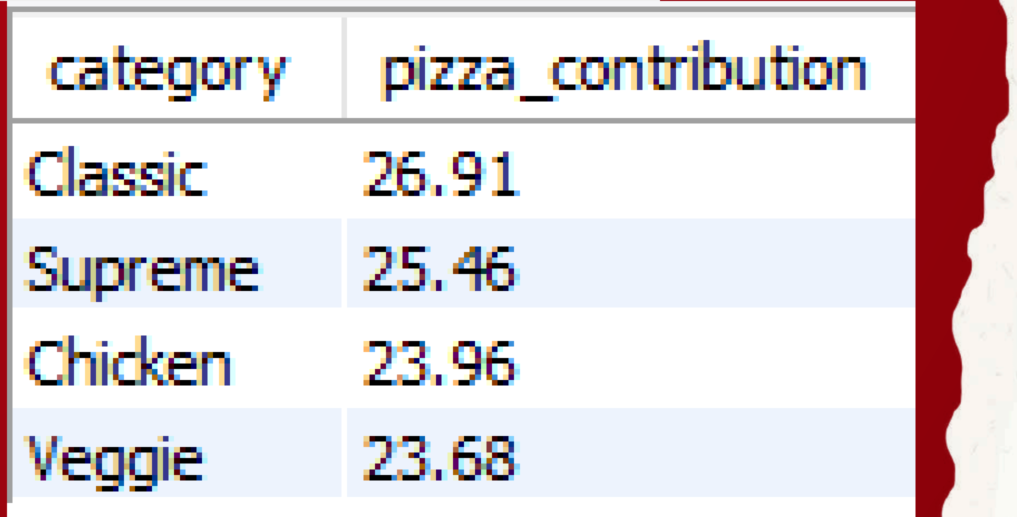


name	most_orderedpizza_onrevenue
The Thai Chicken Pizza	43434
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41410

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

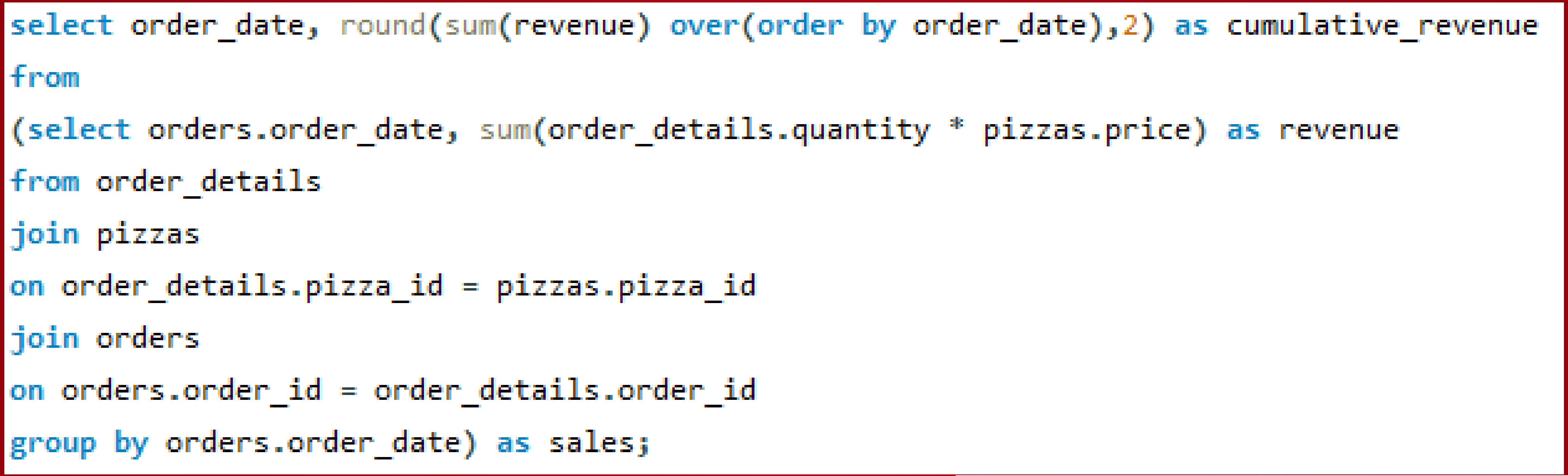


```
select pizza_types.category, round(sum(order_details.quantity * pizzas.price) /
(select round(sum(order_details.quantity * pizzas.price),2) as total_revenue
from order_details
INNER JOIN pizzas
on order_details.pizza_id = pizzas.pizza_id)* 100,2) as pizza_contribution
from pizza_types
join pizzas
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 pizza_contribution desc;
```

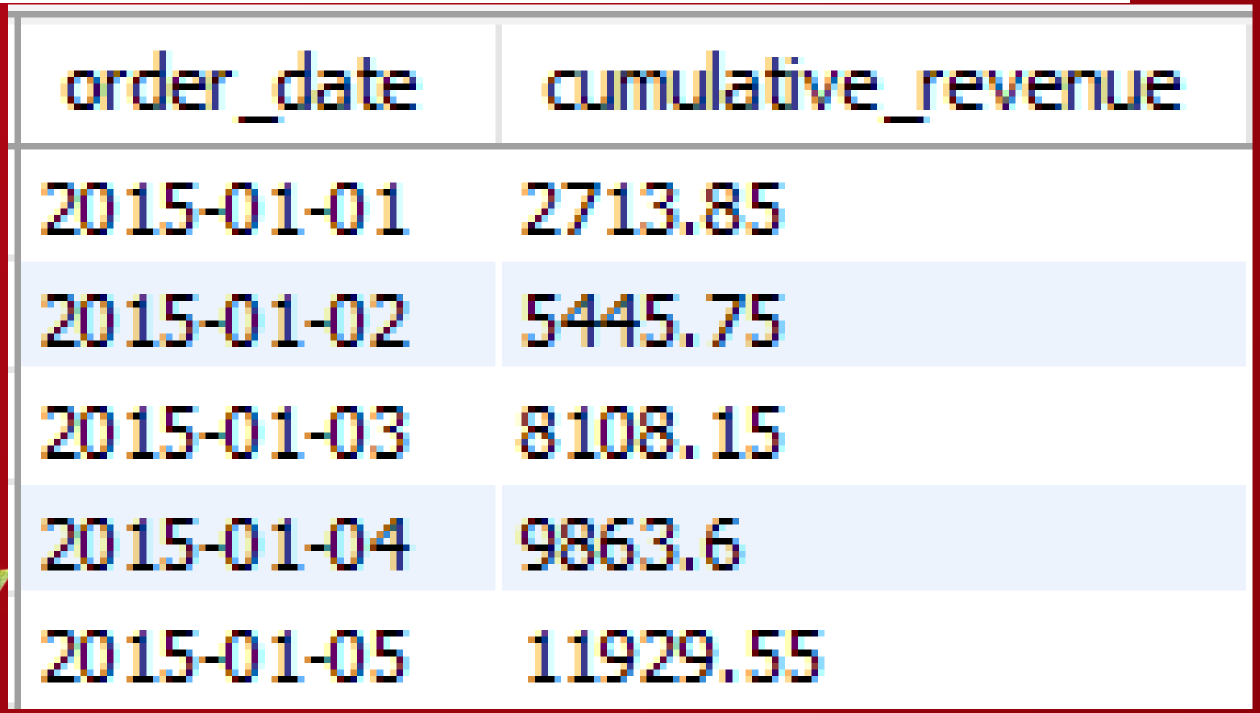


category	pizza_contribution
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

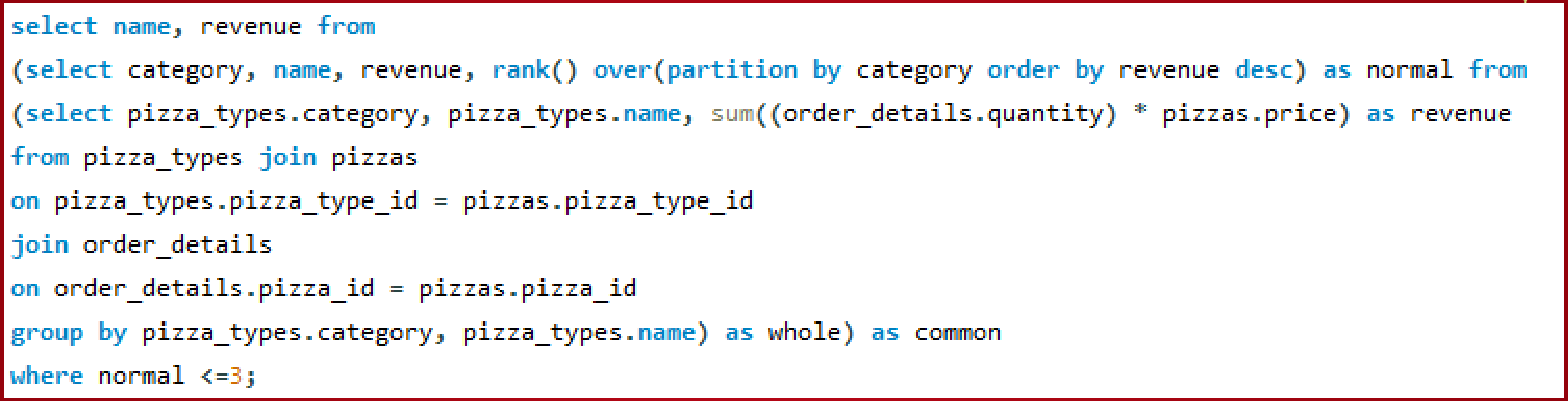


```
select order_date, round(sum(revenue) over(order by order_date),2) as cumulative_revenue
from
(select orders.order_date, sum(order_details.quantity * pizzas.price) as revenue
from order_details
join pizzas
on order_details.pizza_id = pizzas.pizza_id
join orders
on orders.order_id = order_details.order_id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

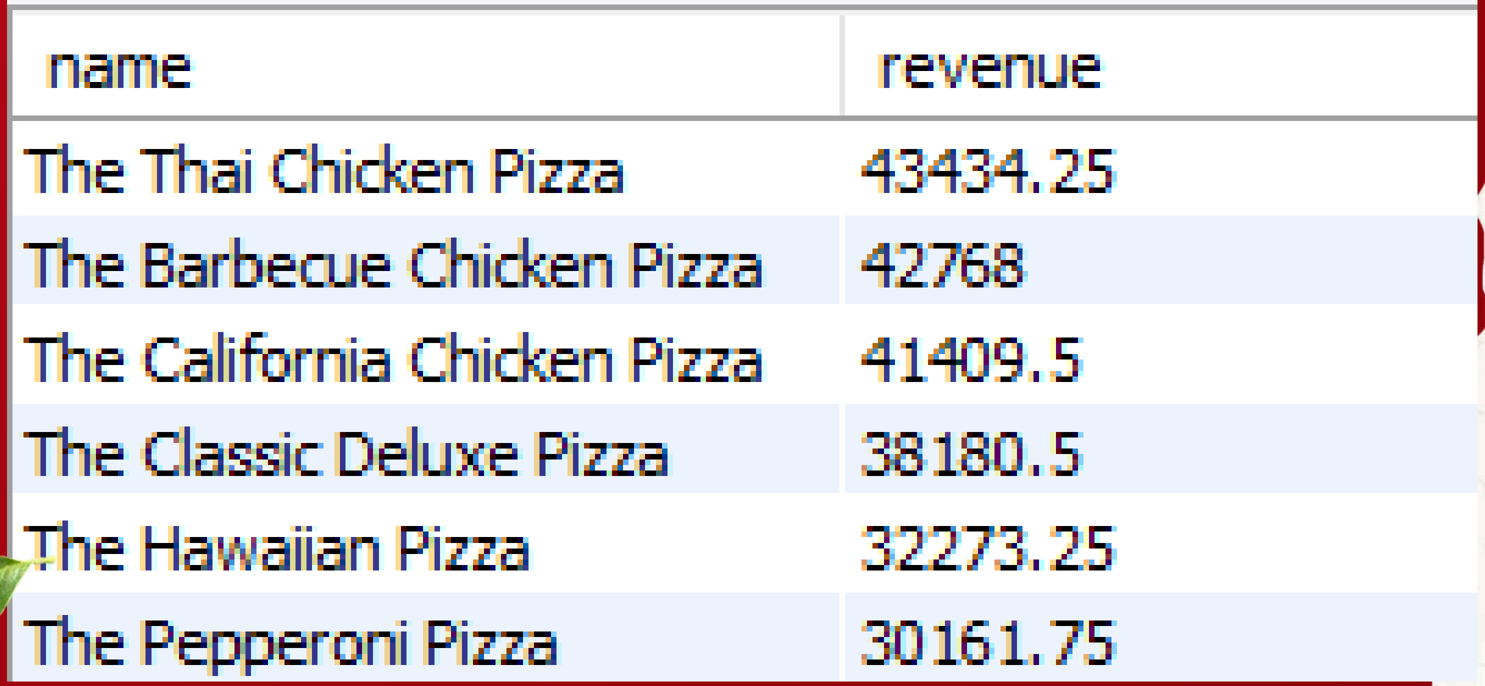


order_date	cumulative_revenue
2015-01-01	2713.85
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55

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



```
select name, revenue from
(select category, name, revenue, rank() over(partition by category order by revenue desc) as normal from
(select pizza_types.category, pizza_types.name, sum((order_details.quantity) * pizzas.price) as 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, pizza_types.name) as whole) as common
where normal <=3;
```



name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
The Classic Deluxe Pizza	38180.5
The Hawaiian Pizza	32273.25
The Pepperoni Pizza	30161.75

BUSINESS INSIGHTS

- Most orders came between 12–1 PM.
- Large size pizzas were most ordered.
- Thai Chicken Pizza was top revenue generator.
- The Greek Pizza has the highest Price.

TOOLS & SQL TECHNIQUES USED

- SQL Joins (INNER, LEFT).
- Aggregate Functions (SUM, COUNT, AVG).
- GROUP BY, ORDER BY, LIMIT.
- Subqueries, CTEs.
- Window Functions (for cumulative revenue).

CONCLUSION

- SQL is powerful tool for analyzing transactional data.
- Actionable insights help drive business decisions.
- Future: Real-time dashboards using Power BI.

THANK YOU

D S Harshit | Aspiring Data Analyst | ds-harshit | d-s-harshit