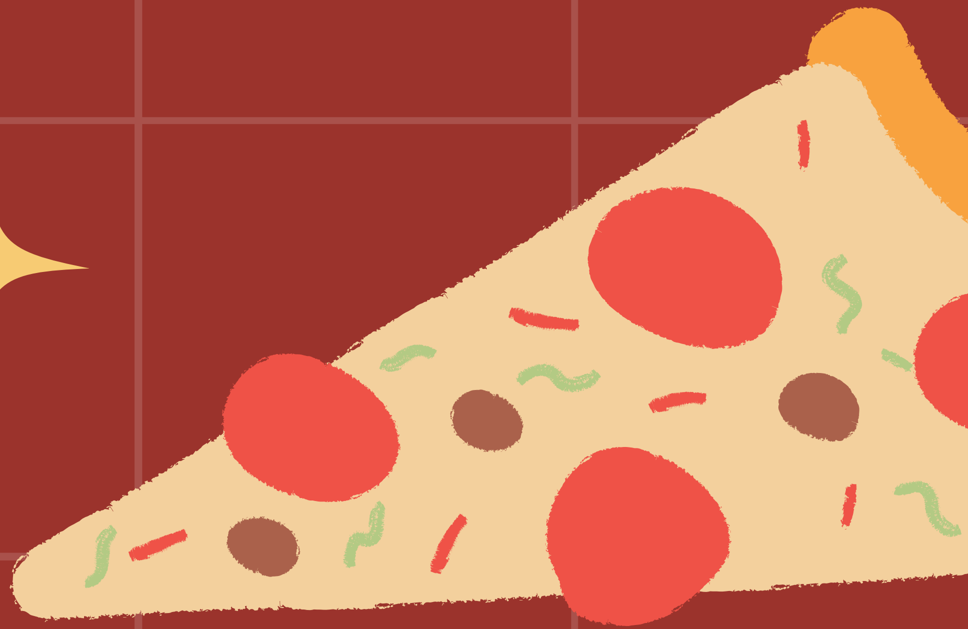
# PIZZA SALES REPORT

@Md Zishan Khan



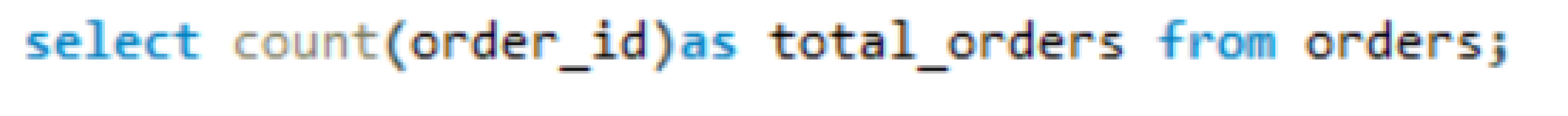
# HELLO!

Hello I am Md Zishan Khan. In this Project, I am using SQl to Slove SQL Query on related to pizza Sales Report

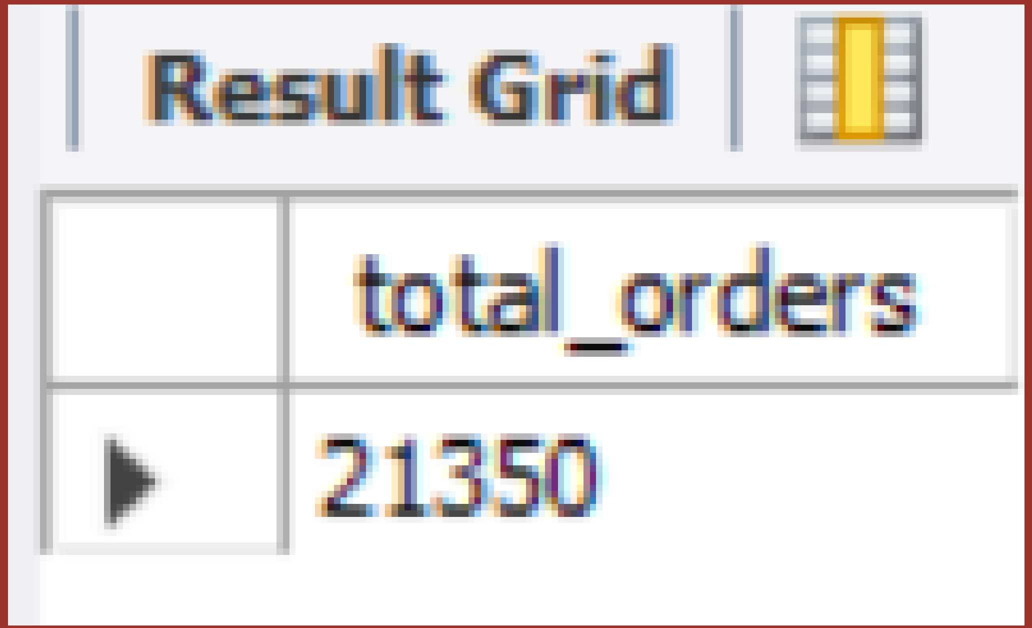
# OBJECTIVES

This SQL project on sales pizza reports aims to leverage data analysis techiques to extract valuable insights from a database , enabling stakeholders to make informed decisions and drive business growth in the competitive pizza industry. By solving a series of progressively complex queries, the project aims to understand sales patterns, revenue generation, and customer preferences, which can help in making data-driven decisions for business improvements.

# Q1. RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

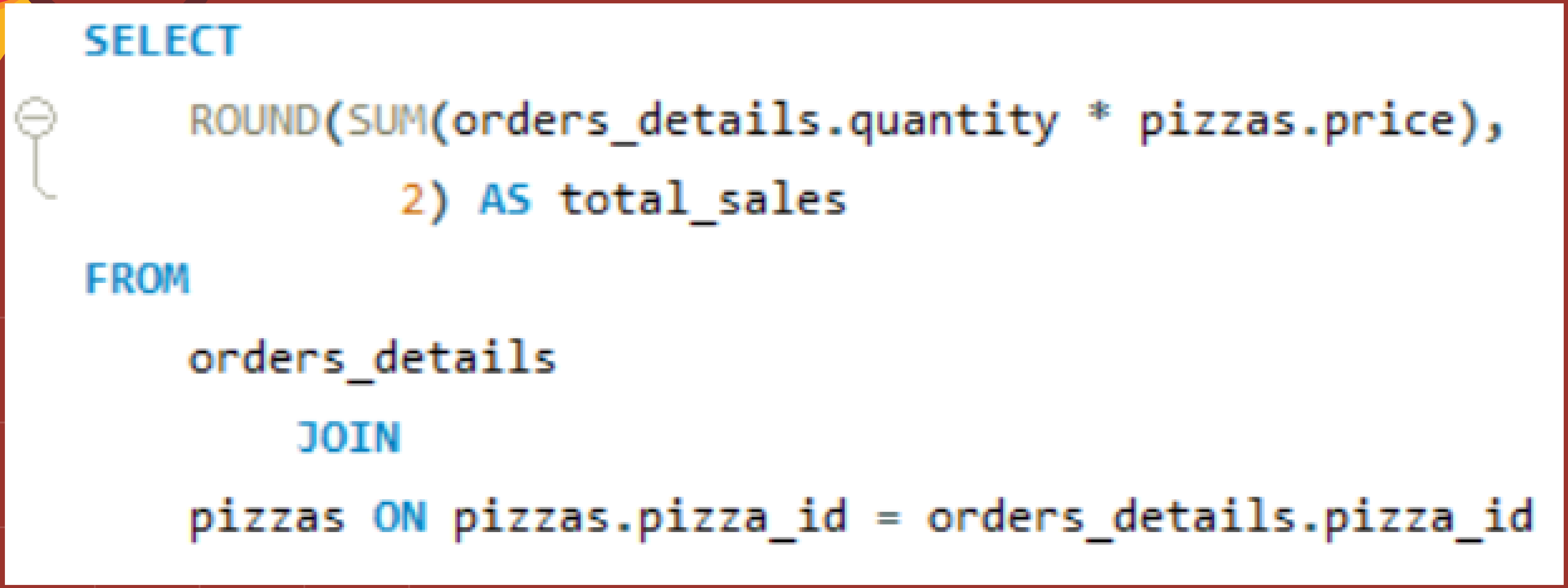


```
select count(order_id)as total_orders from orders;
```

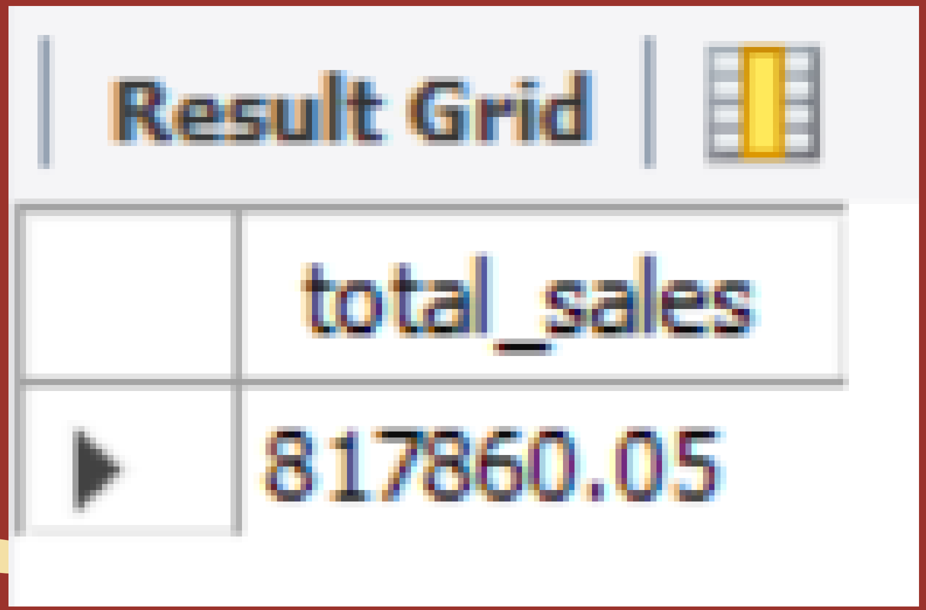


```
  total_orders
▶ 21350
```

# Q2. CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.ORDERS_DETAILS

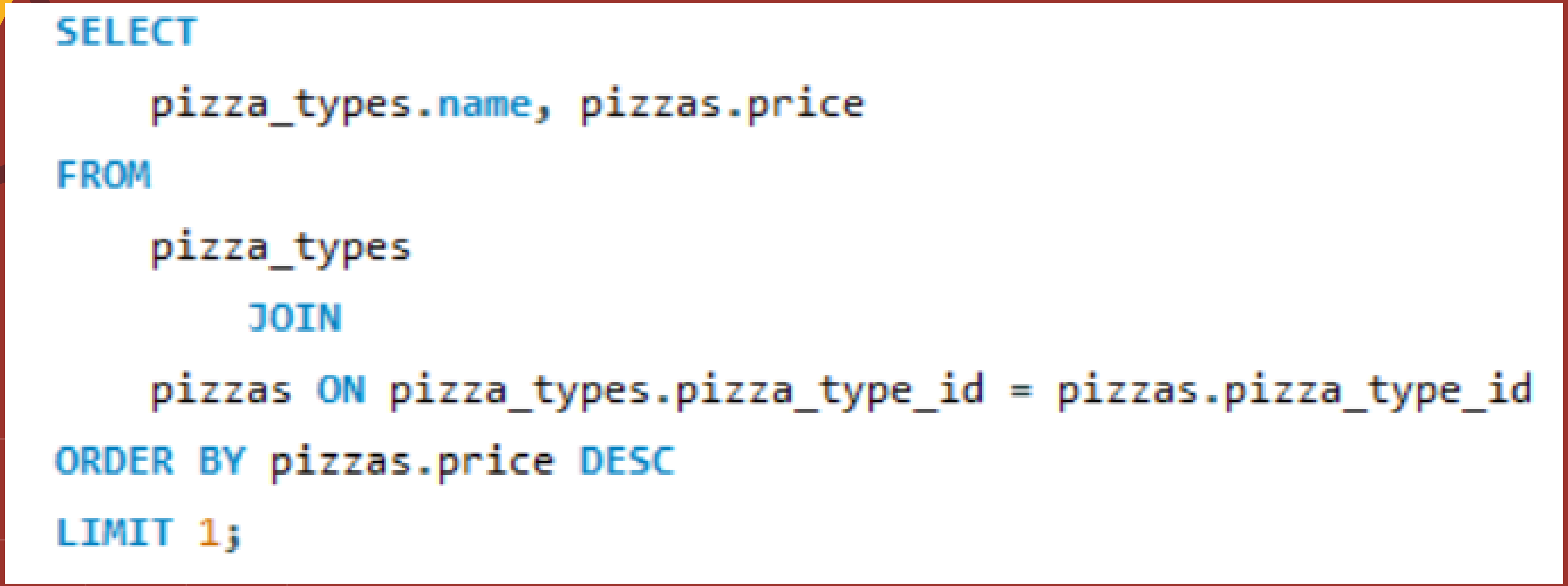


```
SELECT
    ROUND(SUM(orders_details.quantity * pizzas.price),
            2) AS total_sales
FROM
    orders_details
        JOIN
    pizzas ON pizzas.pizza_id = orders_details.pizza_id
```

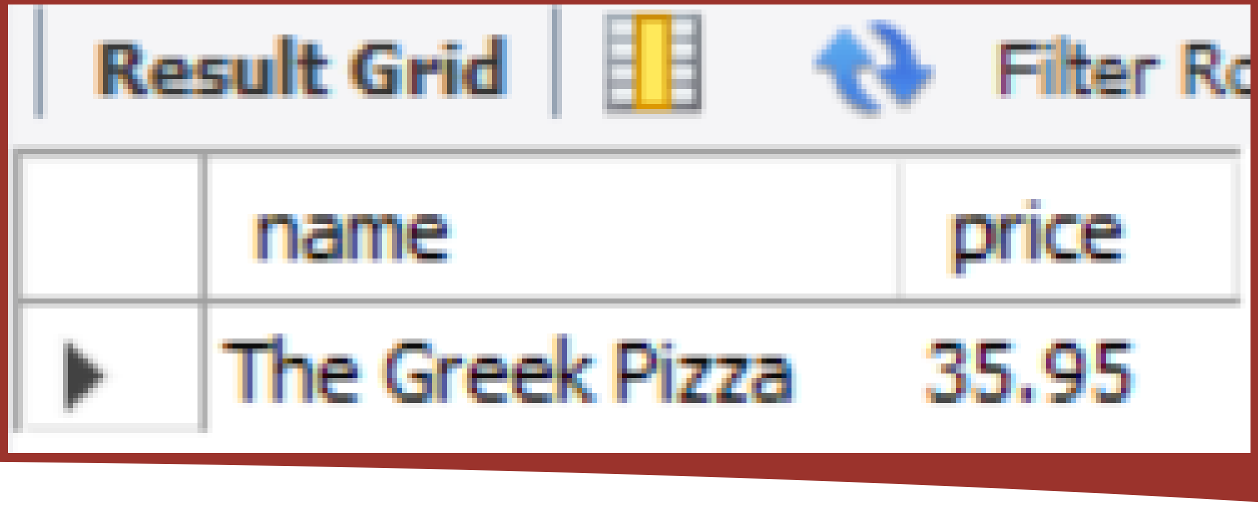


| | total_sales |
|---|---|
| ▶ | 817860.05 |

# Q3. IDENTIFY THE HIGHEST-PRICED PIZZA.

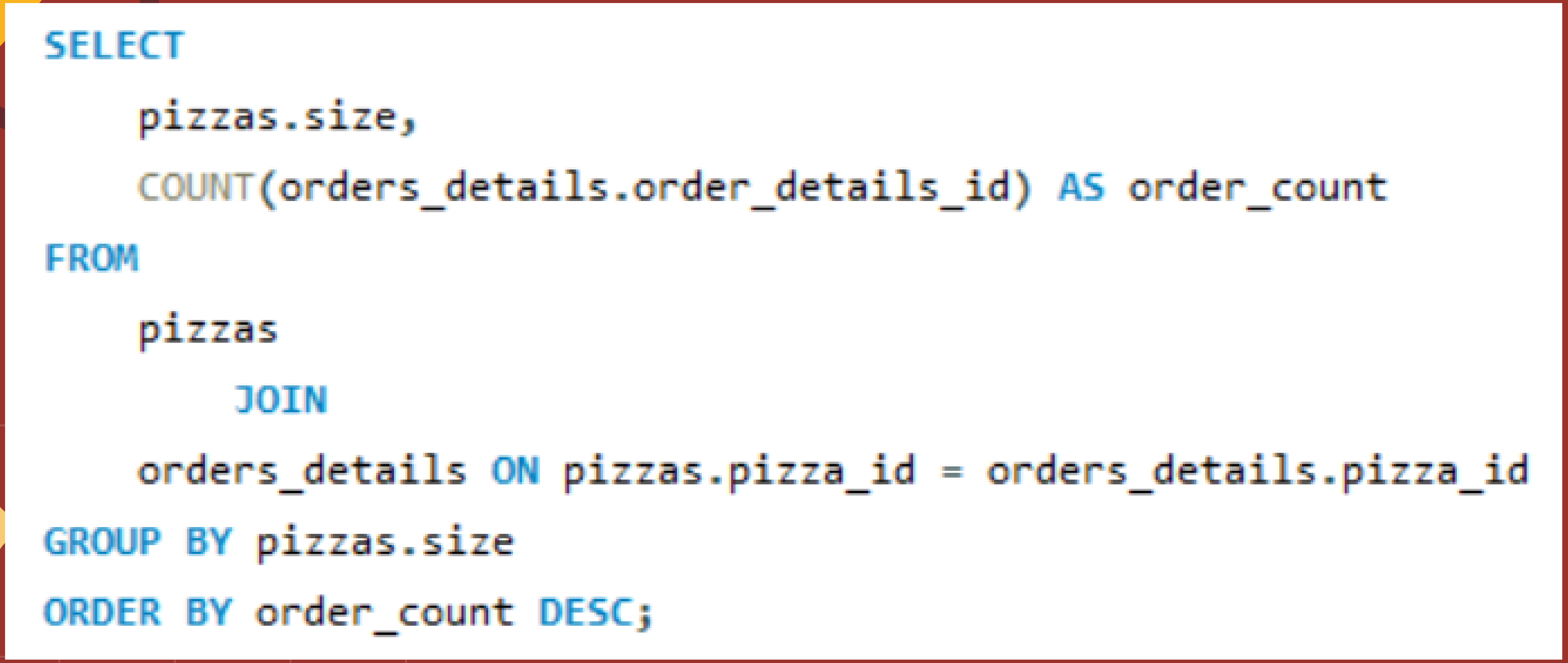


```
SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
```

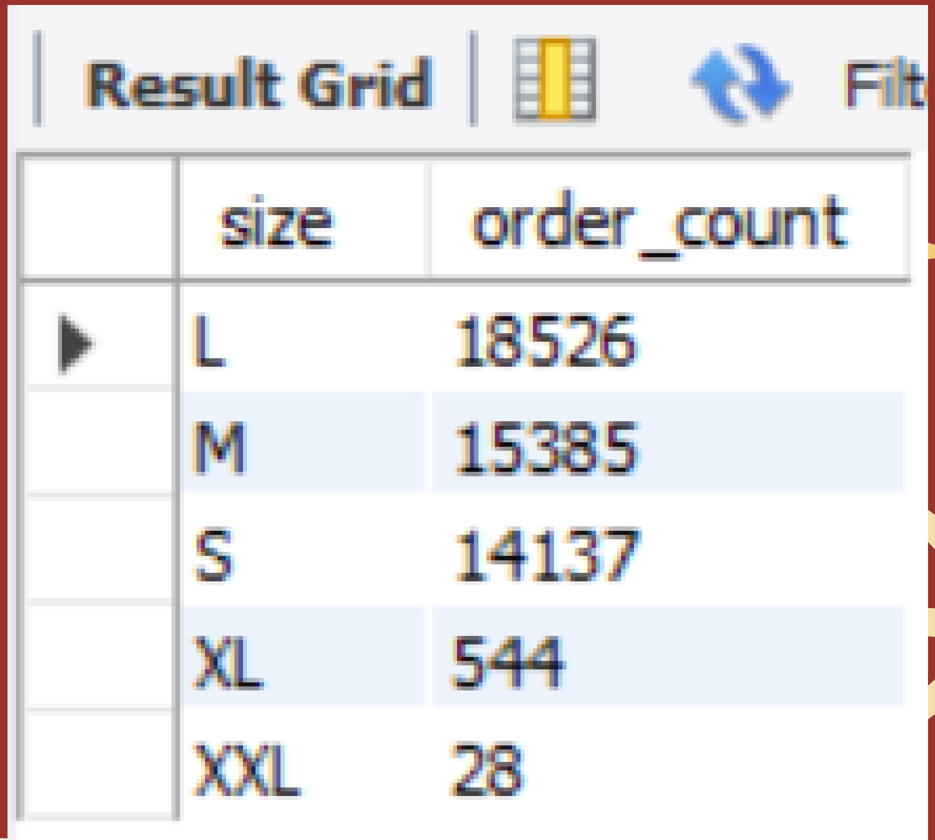


| name | price |
|---|---|
| The Greek Pizza | 35.95 |

# Q4. IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

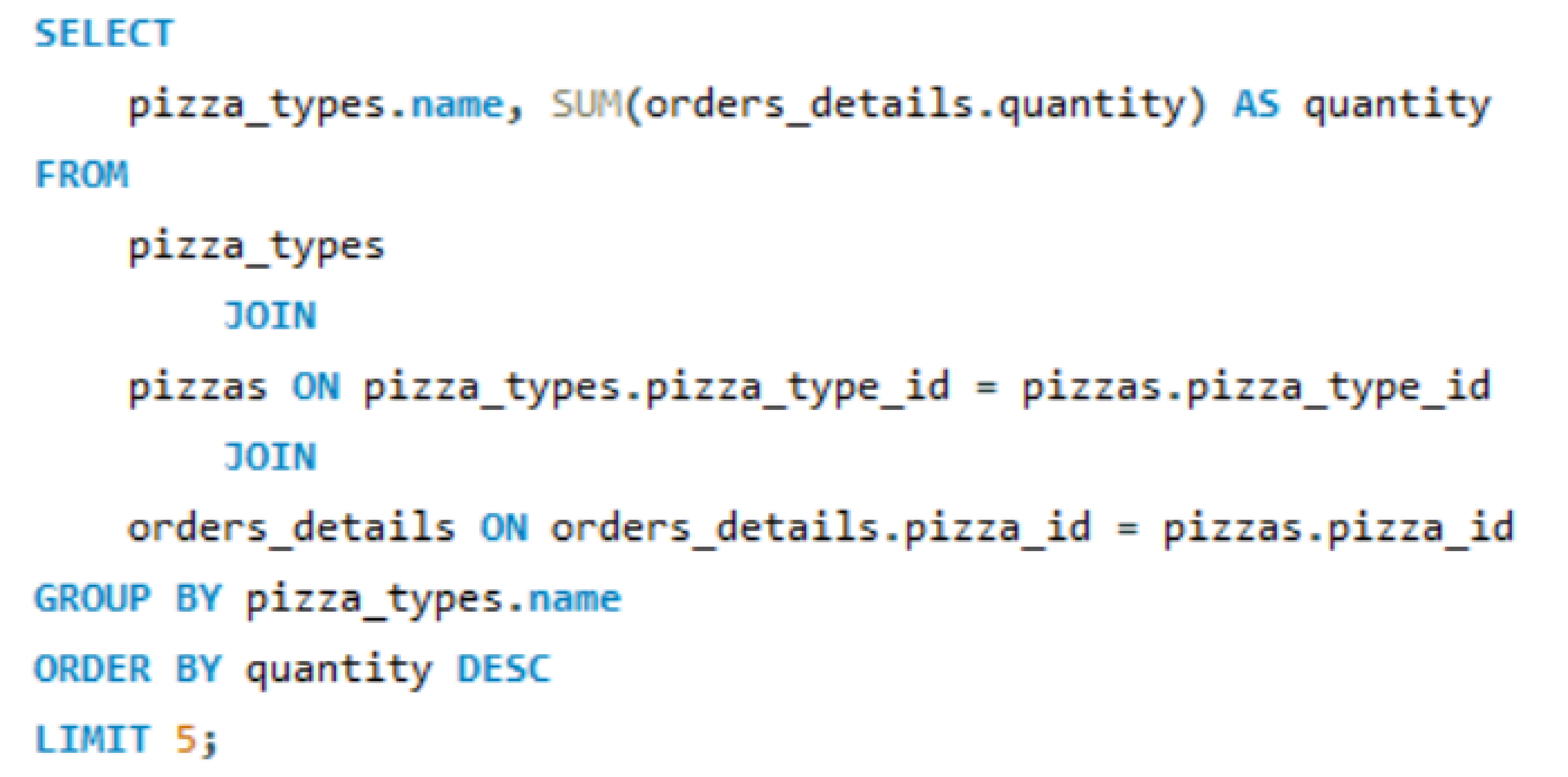


```
SELECT
    pizzas.size,
    COUNT(orders_details.order_details_id) AS order_count
FROM
    pizzas
        JOIN
    orders_details ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC;
```

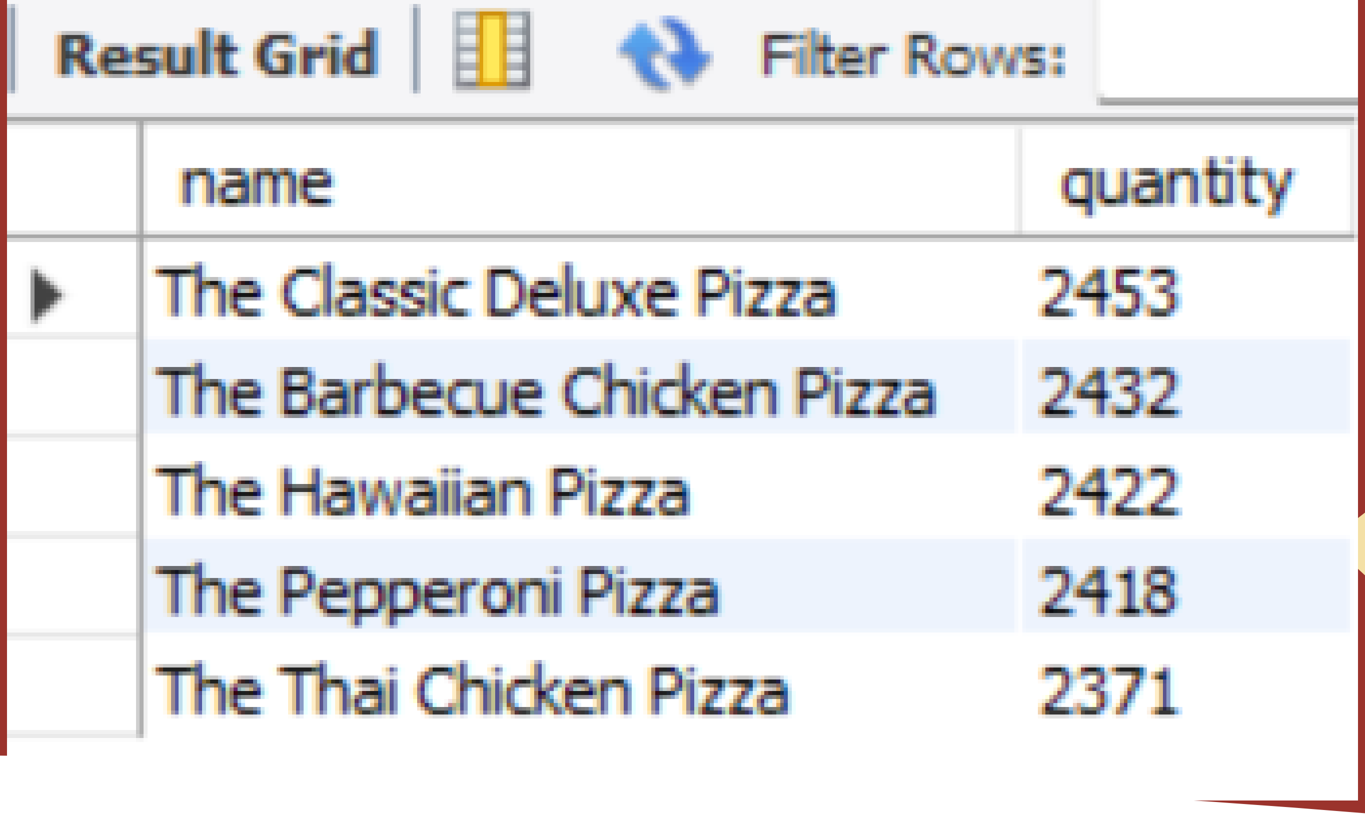


| size | order_count |
|---|---|
| L | 18526 |
| M | 15385 |
| S | 14137 |
| XL | 544 |
| XXL | 28 |

# Q5. LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.

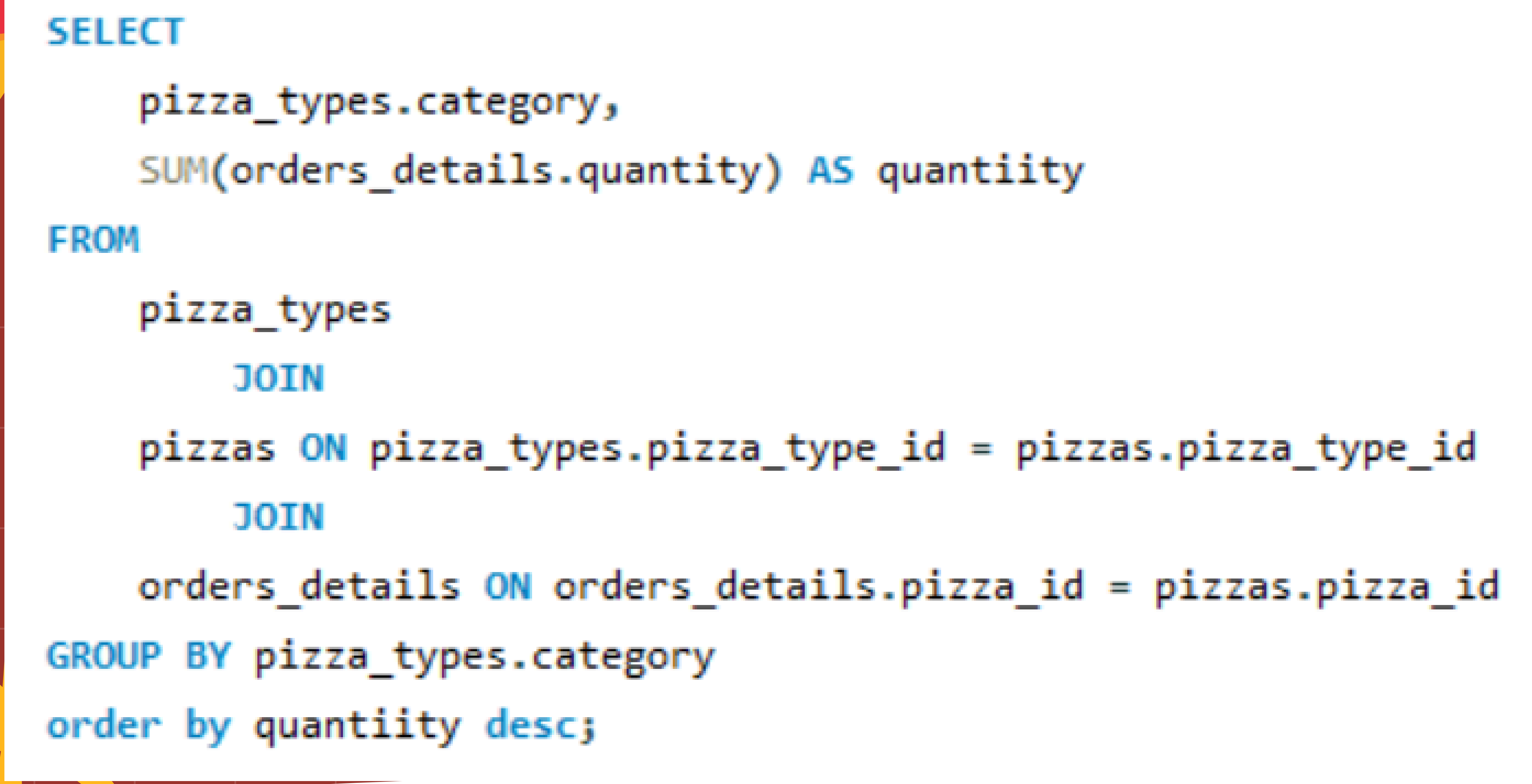


```
SELECT
    pizza_types.name, SUM(orders_details.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY quantity DESC
LIMIT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

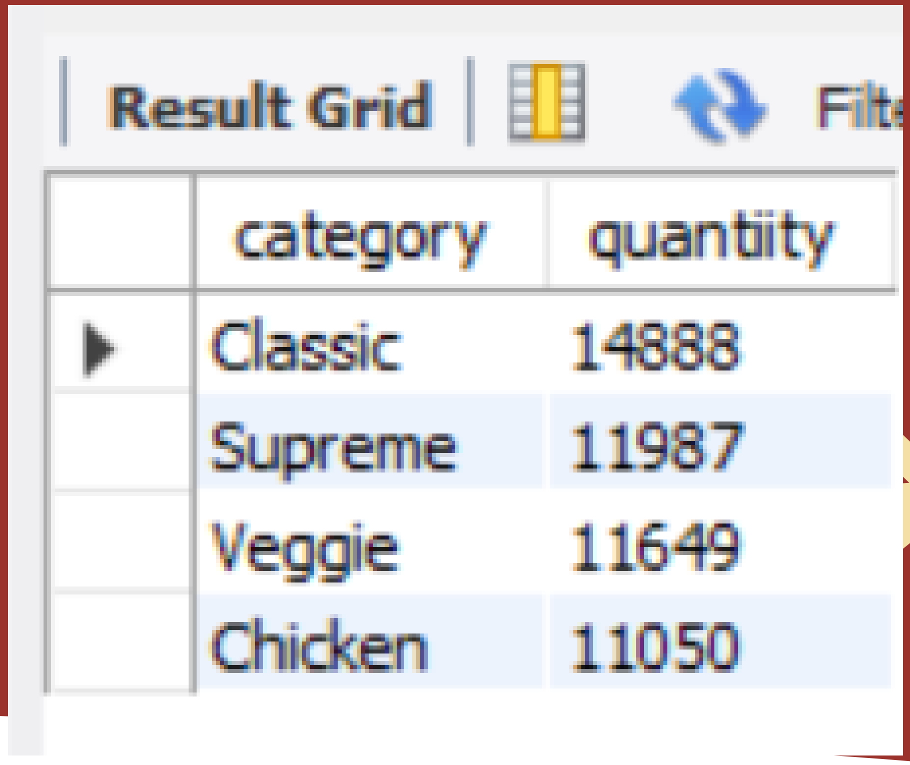


| name | quantity |
|---|---|
| The Classic Deluxe Pizza | 2453 |
| The Barbecue Chicken Pizza | 2432 |
| The Hawaiian Pizza | 2422 |
| The Pepperoni Pizza | 2418 |
| The Thai Chicken Pizza | 2371 |

# Q6. JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

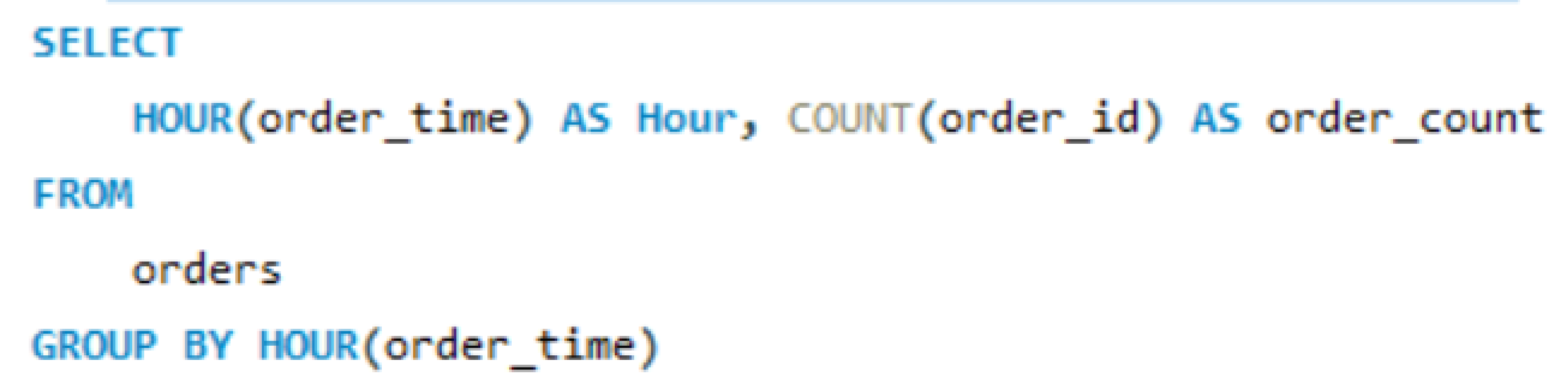


```
SELECT
    pizza_types.category,
    SUM(orders_details.quantity) AS quantiity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
order by quantiity desc;
```

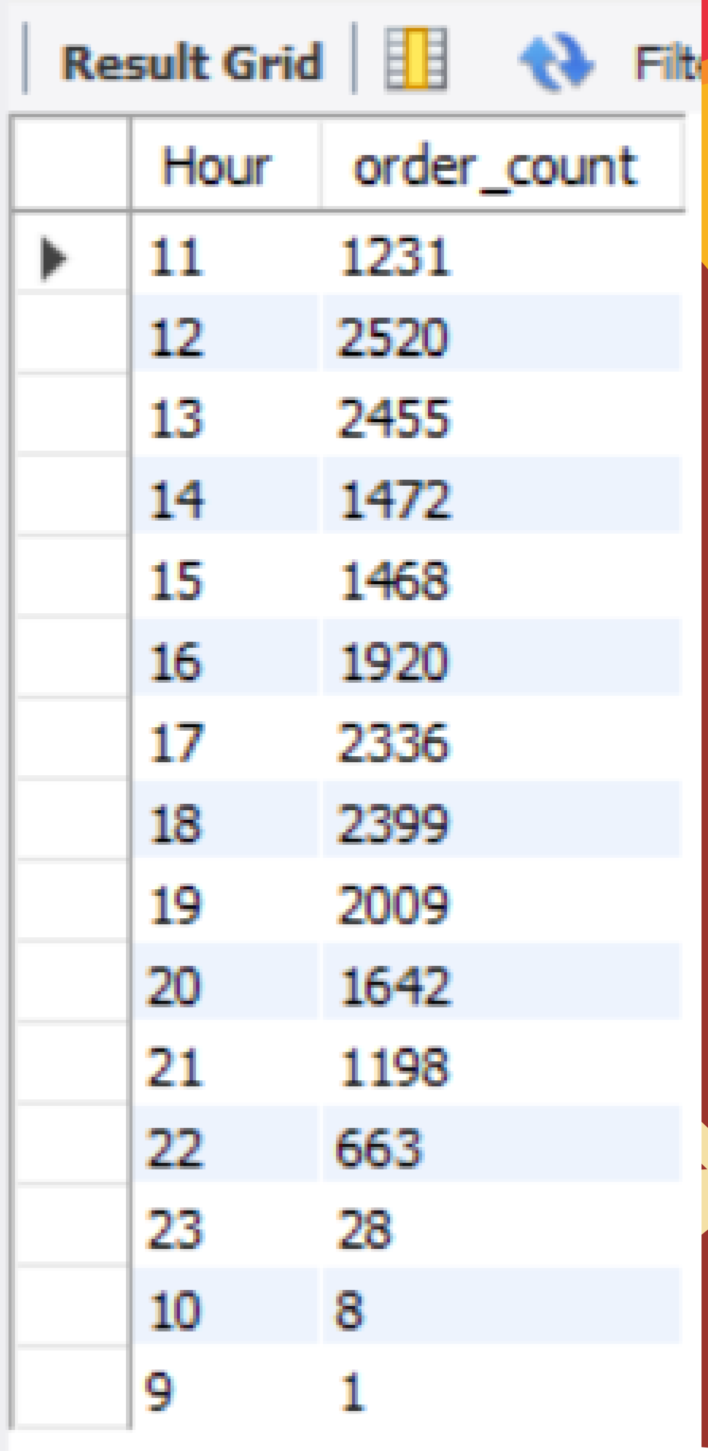


| category | quantiity |
|---|---|
| Classic | 14888 |
| Supreme | 11987 |
| Veggie | 11649 |
| Chicken | 11050 |

# Q7. DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

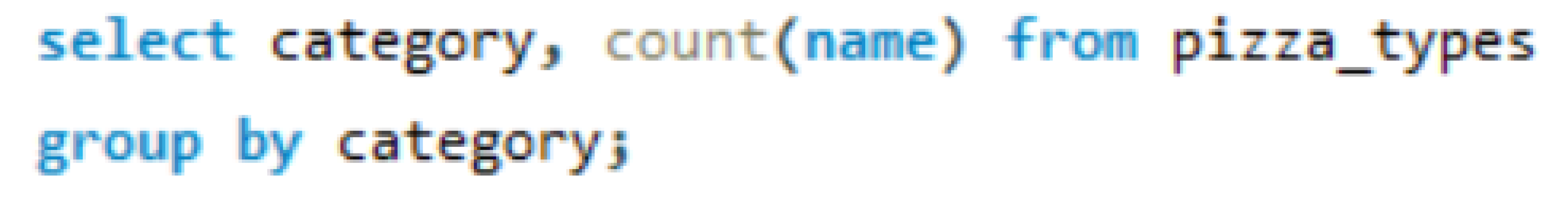


```
SELECT
    HOUR(order_time) AS Hour, COUNT(order_id) AS order_count
FROM
    orders
GROUP BY HOUR(order_time)
```



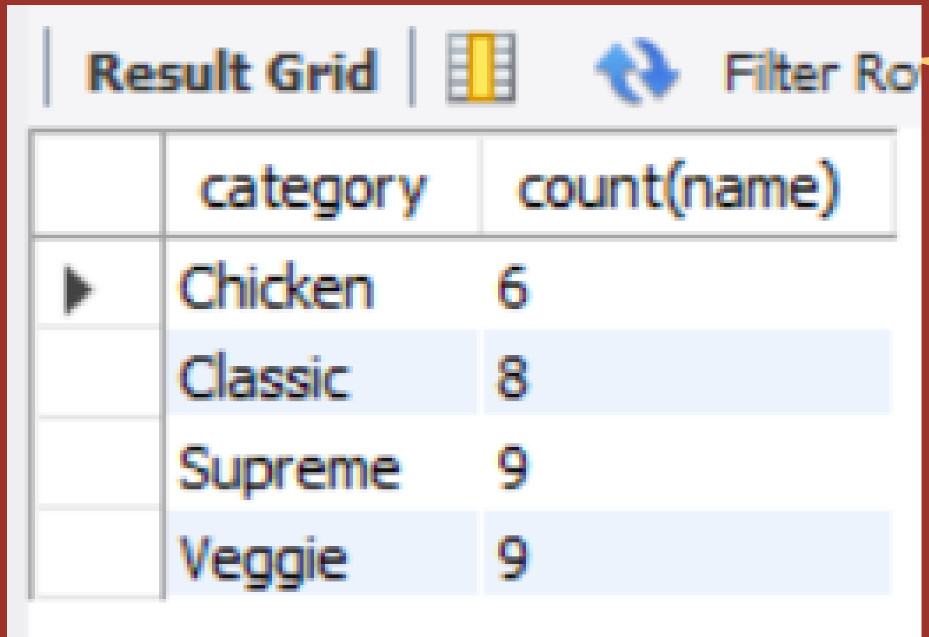
| Hour | order_count |
|---|---|
| 11 | 1231 |
| 12 | 2520 |
| 13 | 2455 |
| 14 | 1472 |
| 15 | 1468 |
| 16 | 1920 |
| 17 | 2336 |
| 18 | 2399 |
| 19 | 2009 |
| 20 | 1642 |
| 21 | 1198 |
| 22 | 663 |
| 23 | 28 |
| 10 | 8 |
| 9 | 1 |

# Q8. JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.



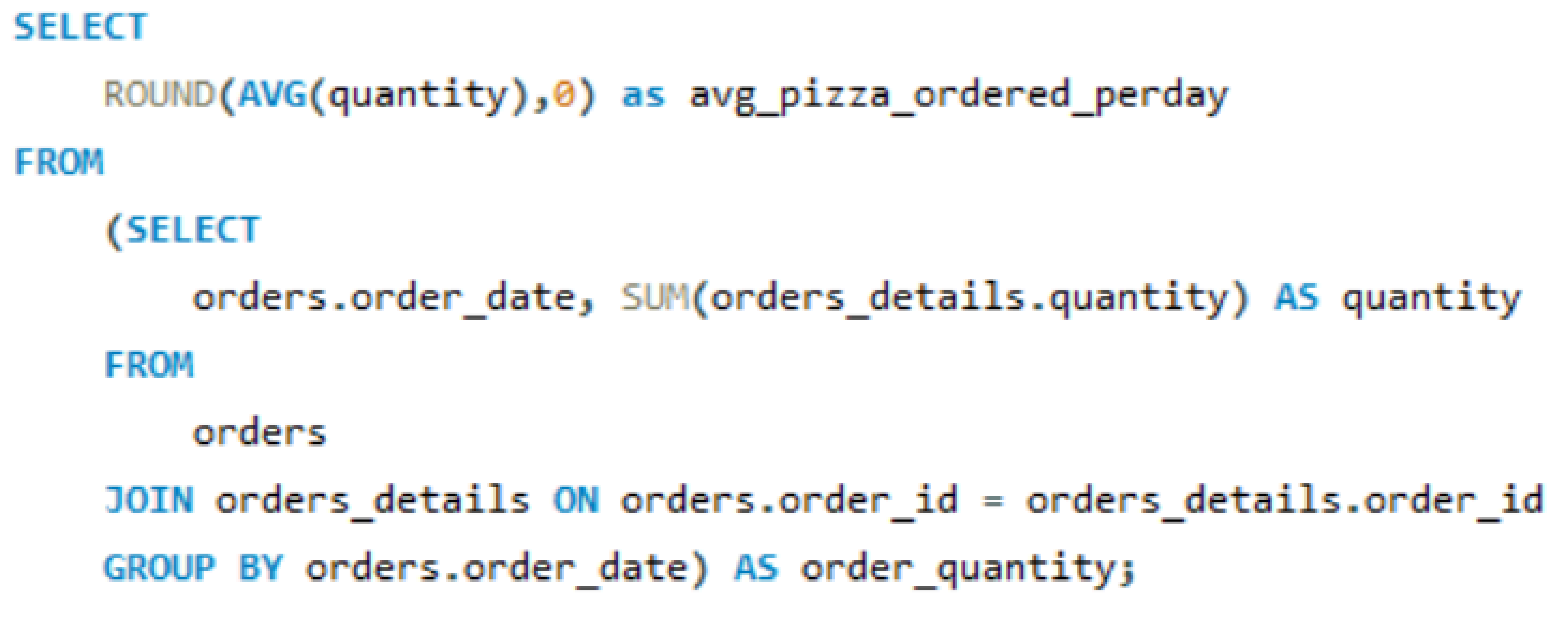
```
select category, count(name) from pizza_types

group by category;
```

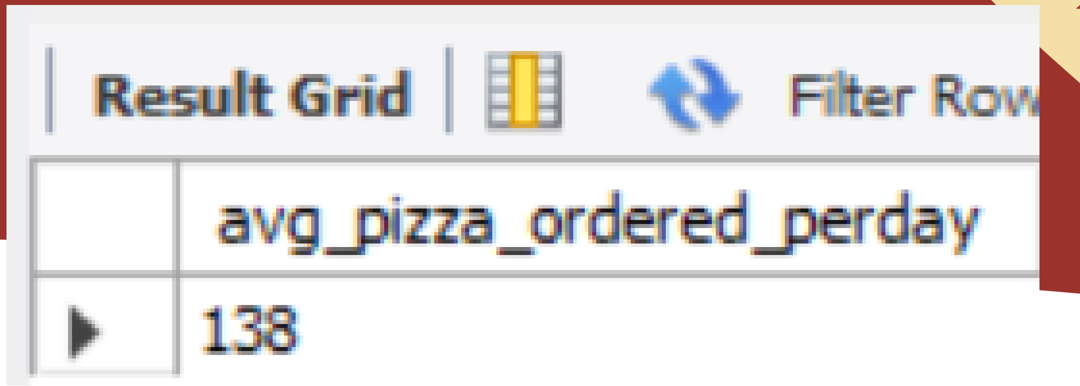


| category | count(name) |
|---|---|
| Chicken | 6 |
| Classic | 8 |
| Supreme | 9 |
| Veggie | 9 |

# Q9. GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAG NUMBER OF PIZZAS ORDERED PER DAY.

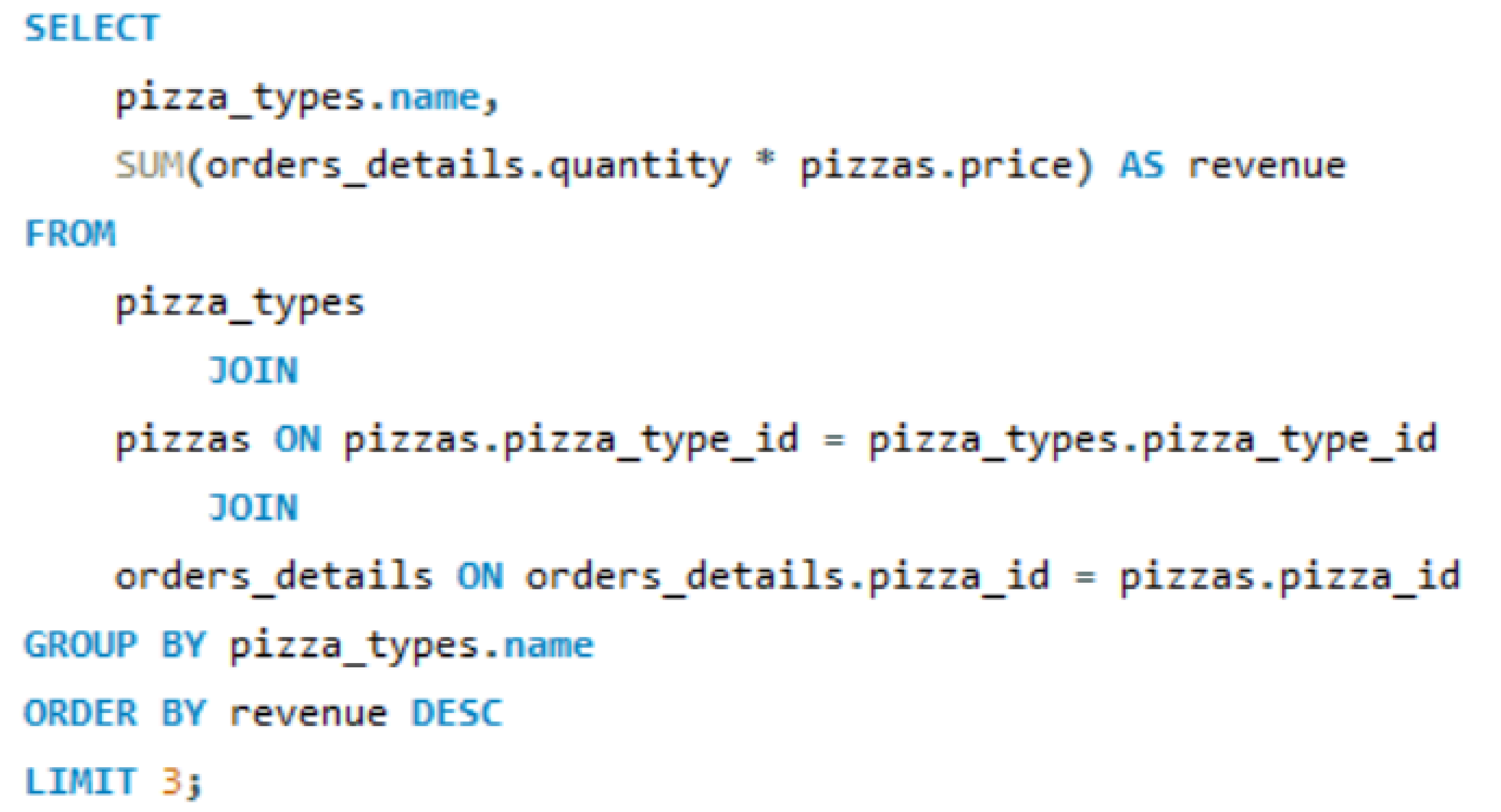


```
SELECT
    ROUND(AVG(quantity),0) as avg_pizza_ordered_perday
FROM
    (SELECT
        orders.order_date, SUM(orders_details.quantity) AS quantity
    FROM
        orders
    JOIN orders_details ON orders.order_id = orders_details.order_id
    GROUP BY orders.order_date) AS order_quantity;
```

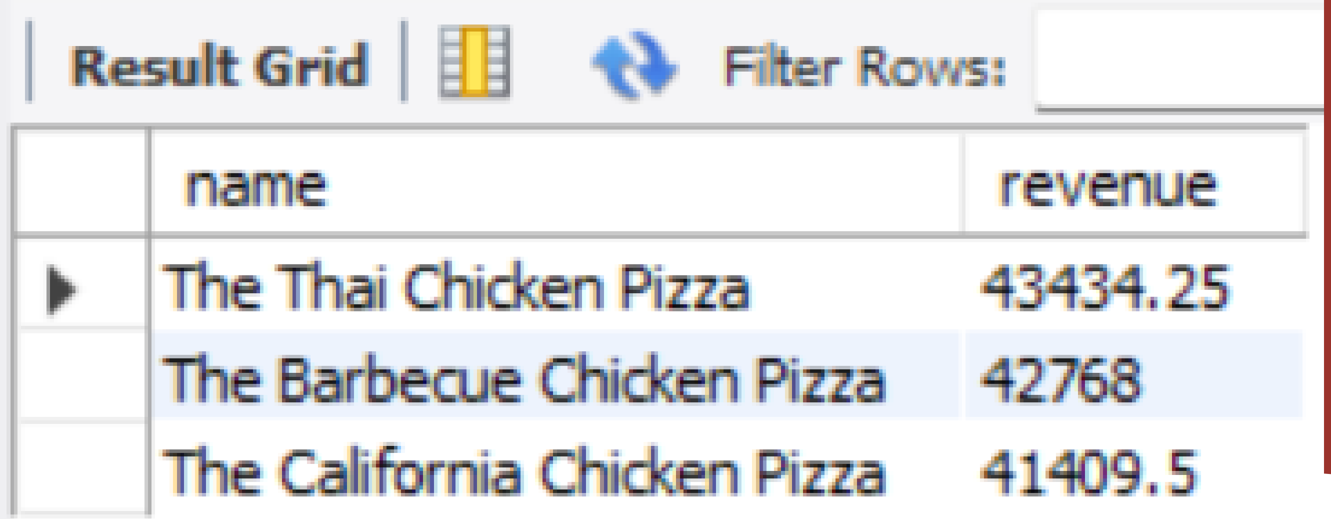


```
avg_pizza_ordered_perday
138
```

# Q10. DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

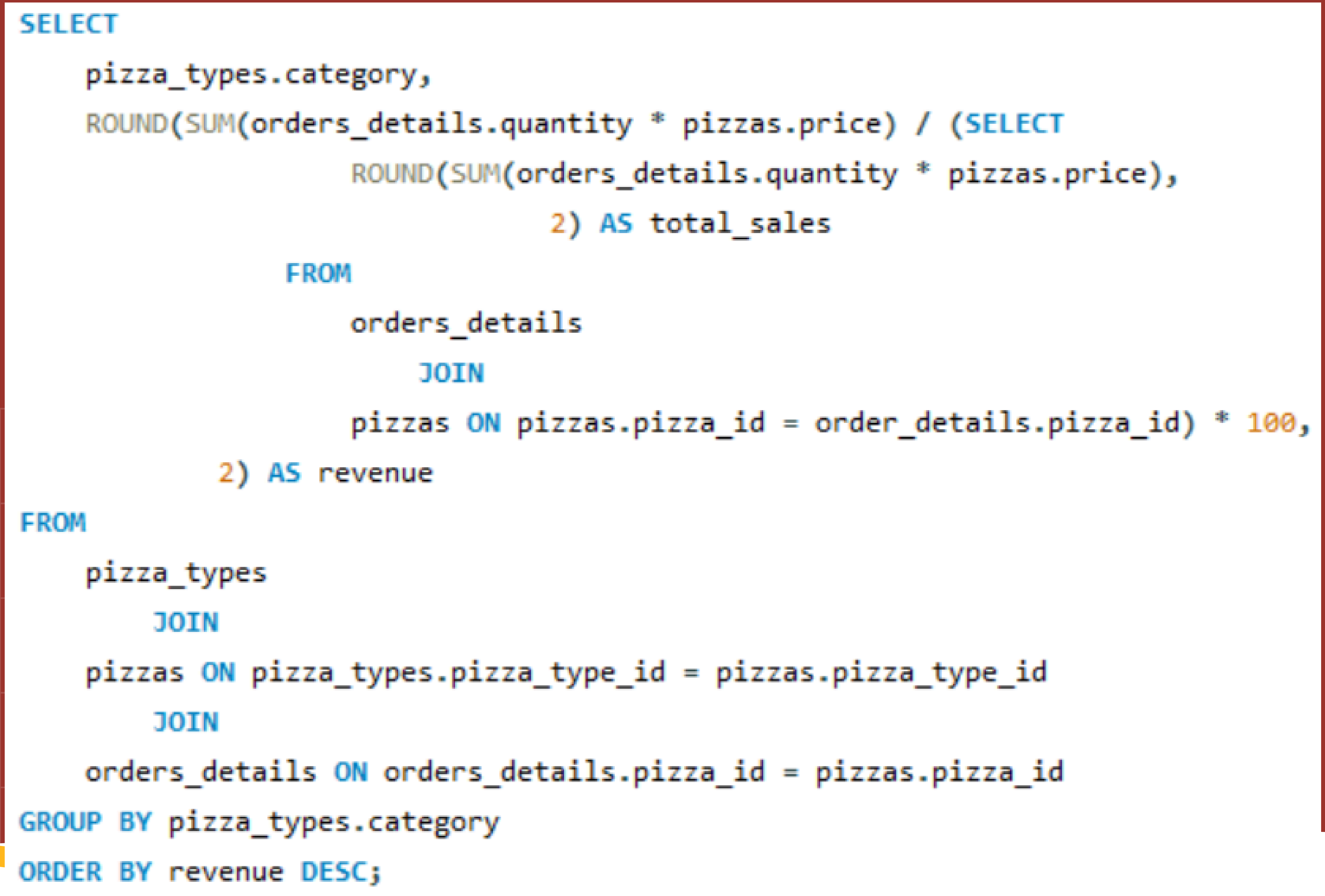


```
SELECT
    pizza_types.name,
    SUM(orders_details.quantity * pizzas.price) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY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

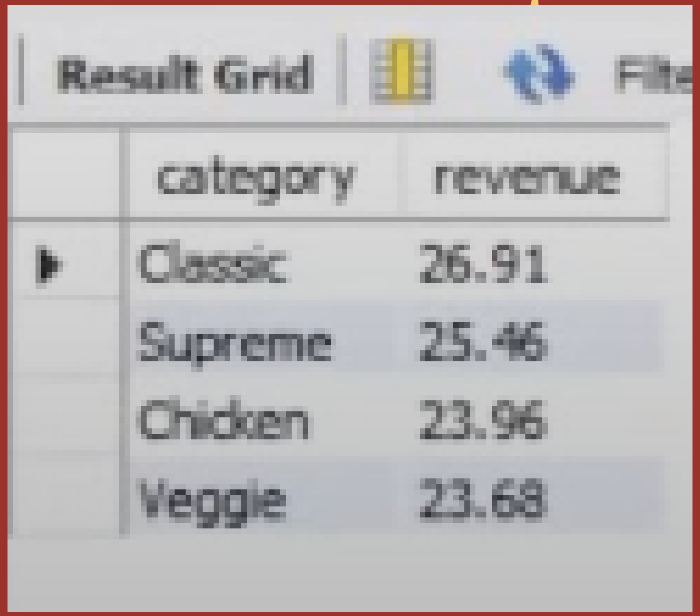


| name | revenue |
|---|---|
| The Thai Chicken Pizza | 43434.25 |
| The Barbecue Chicken Pizza | 42768 |
| The California Chicken Pizza | 41409.5 |

# Q11. CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

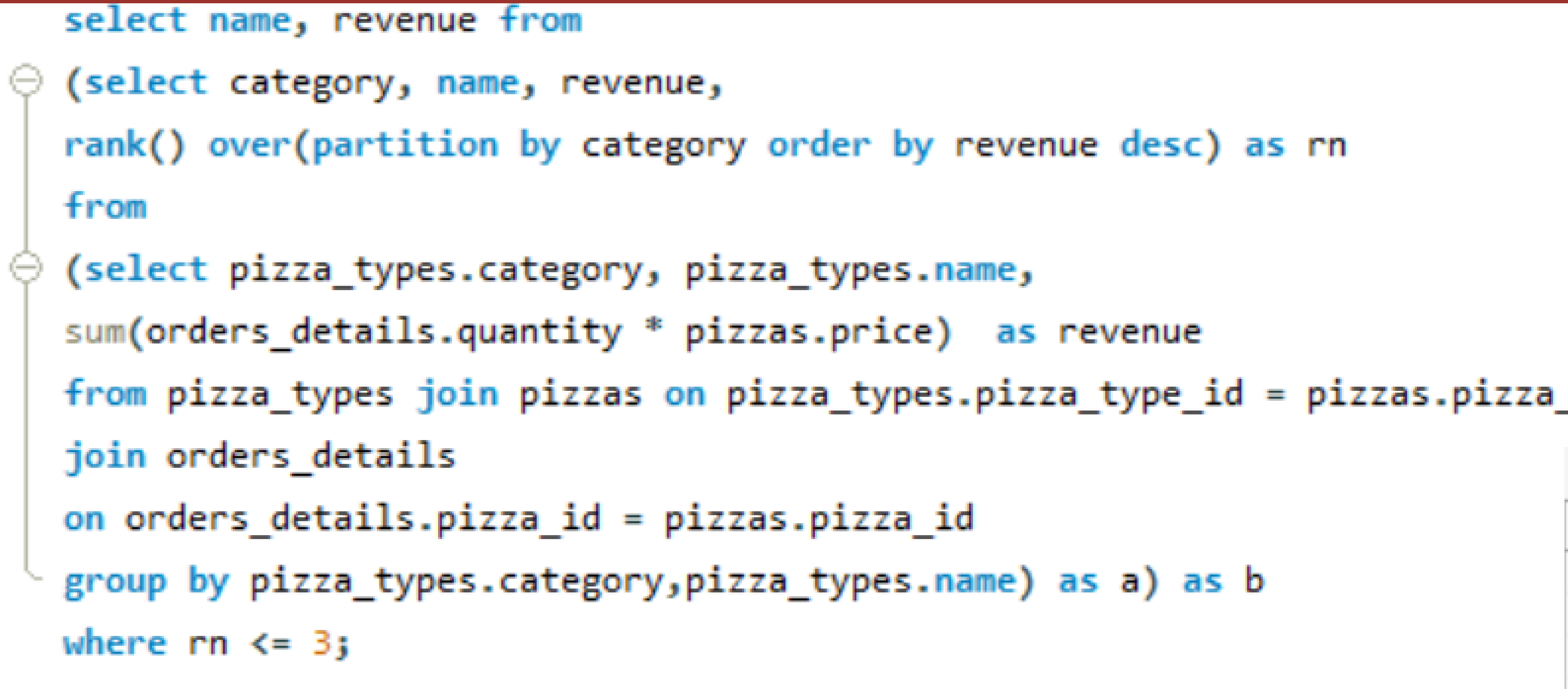


```
SELECT
    pizza_types.category,
    ROUND(SUM(orders_details.quantity * pizzas.price) / (SELECT
                ROUND(SUM(orders_details.quantity * pizzas.price),
                            2) AS total_sales
            FROM
                orders_details
                    JOIN
             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

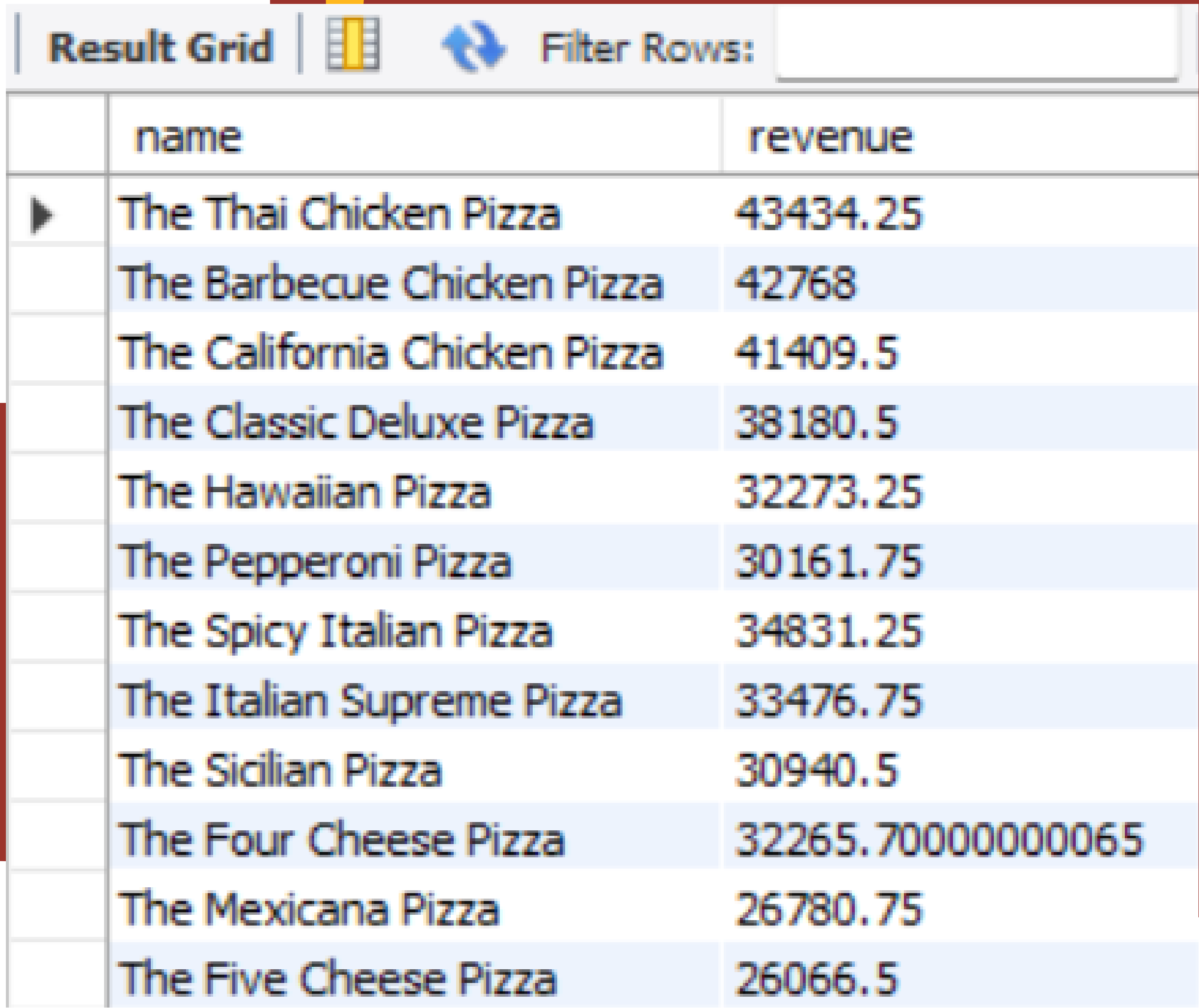


| category | revenue |
|---|---|
| Classic | 26.91 |
| Supreme | 25.46 |
| Chicken | 23.96 |
| Veggie | 23.68 |

# Q12. DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.

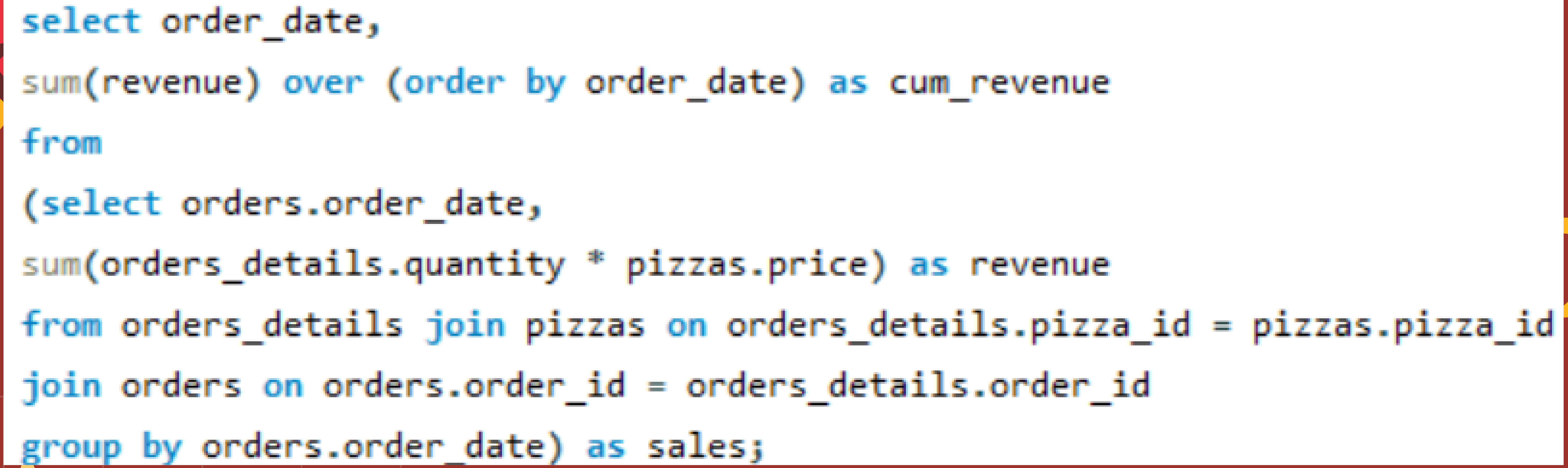


```
select name, revenue from
(select category, name, revenue,
rank() over(partition by category order by revenue desc) as rn
from
(select pizza_types.category, pizza_types.name,
sum(orders_details.quantity * pizzas.price)  as revenue
from pizza_types join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id
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;
```

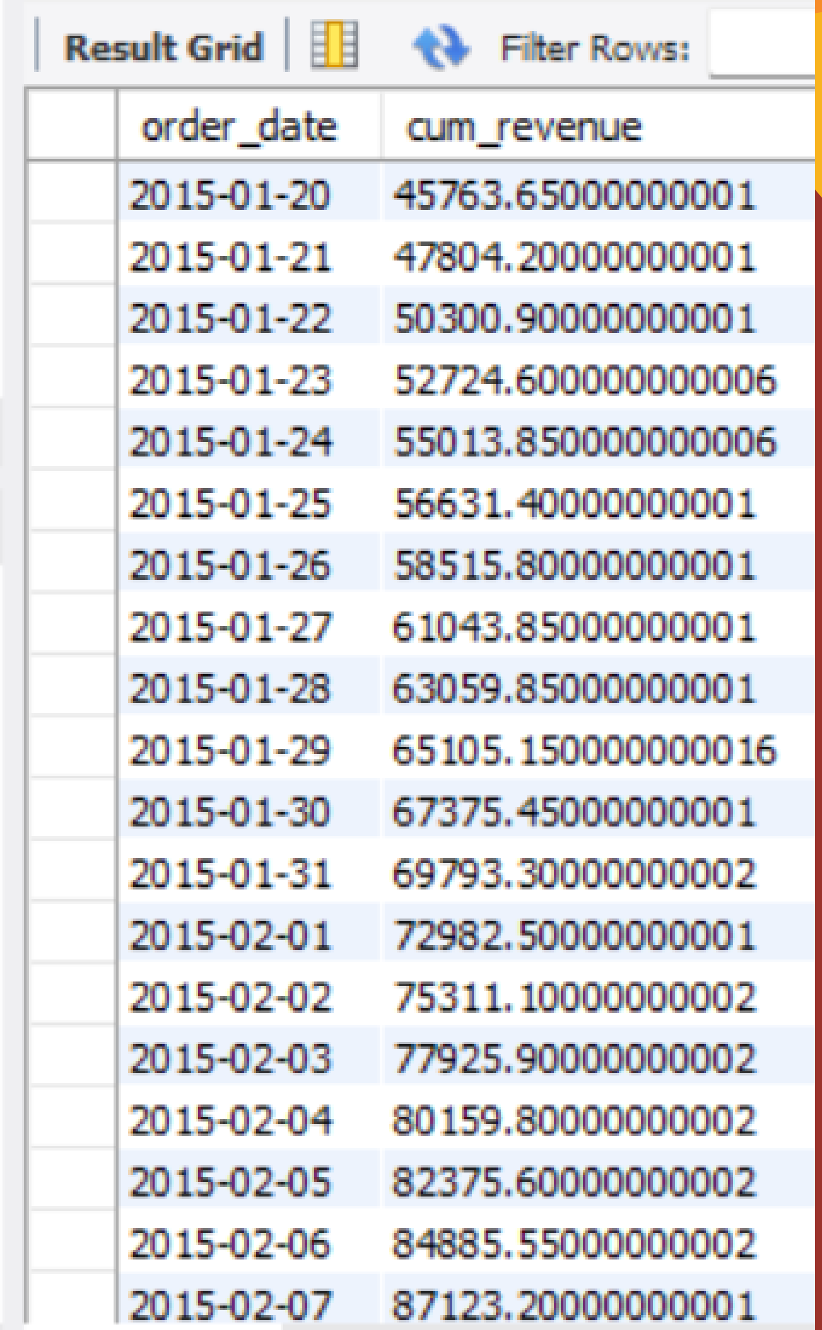


| name | revenue |
|---|---|
| The Thai Chicken Pizza | 43434.25 |
| The Barbecue Chicken Pizza | 42768 |
| The California Chicken Pizza | 41409.5 |
| The Classic Deluxe Pizza | 38180.5 |
| The Hawaiian Pizza | 32273.25 |
| The Pepperoni Pizza | 30161.75 |
| The Spicy Italian Pizza | 34831.25 |
| The Italian Supreme Pizza | 33476.75 |
| The Sicilian Pizza | 30940.5 |
| The Four Cheese Pizza | 32265.70000000065 |
| The Mexicana Pizza | 26780.75 |
| The Five Cheese Pizza | 26066.5 |

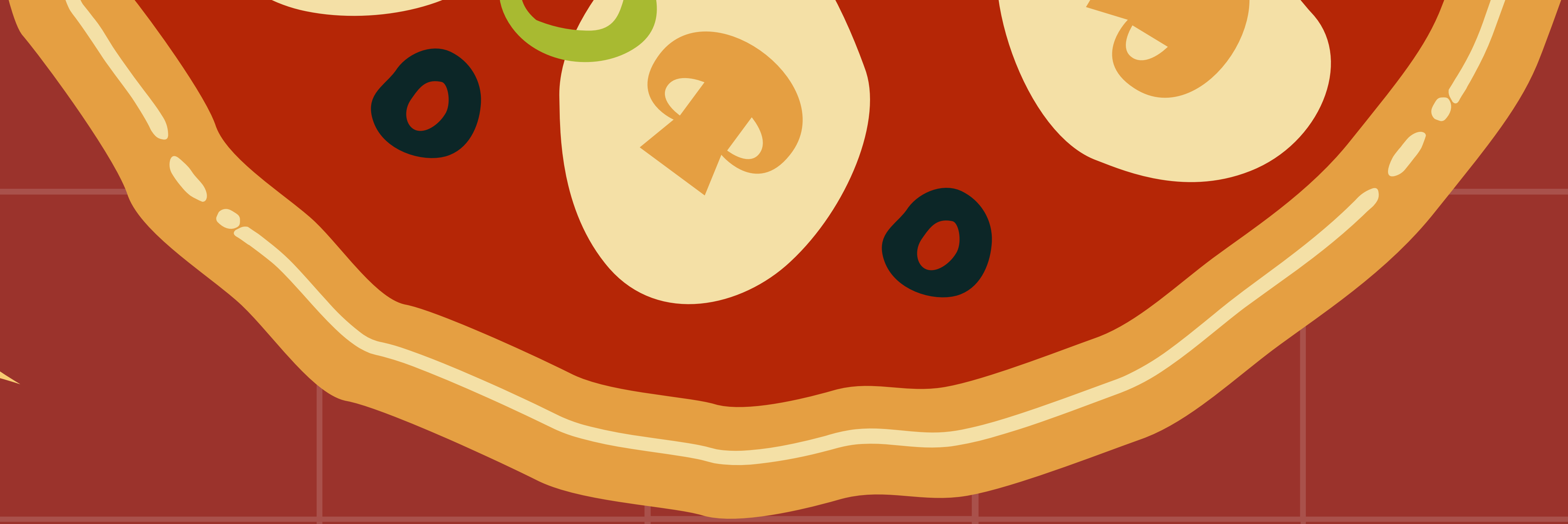
# Q13. ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.



```
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 |
|---|---|
| 2015-01-20 | 45763.65000000001 |
| 2015-01-21 | 47804.20000000001 |
| 2015-01-22 | 50300.90000000001 |
| 2015-01-23 | 52724.600000000006 |
| 2015-01-24 | 55013.850000000006 |
| 2015-01-25 | 56631.40000000001 |
| 2015-01-26 | 58515.80000000001 |
| 2015-01-27 | 61043.85000000001 |
| 2015-01-28 | 63059.85000000001 |
| 2015-01-29 | 65105.150000000016 |
| 2015-01-30 | 67375.45000000001 |
| 2015-01-31 | 69793.30000000002 |
| 2015-02-01 | 72982.50000000001 |
| 2015-02-02 | 75311.10000000002 |
| 2015-02-03 | 77925.90000000002 |
| 2015-02-04 | 80159.80000000002 |
| 2015-02-05 | 82375.60000000002 |
| 2015-02-06 | 84885.55000000002 |
| 2015-02-07 | 87123.20000000001 |



# PIZZA SALES REPORT

# THANK YOU!

zishan9939@gmail.com