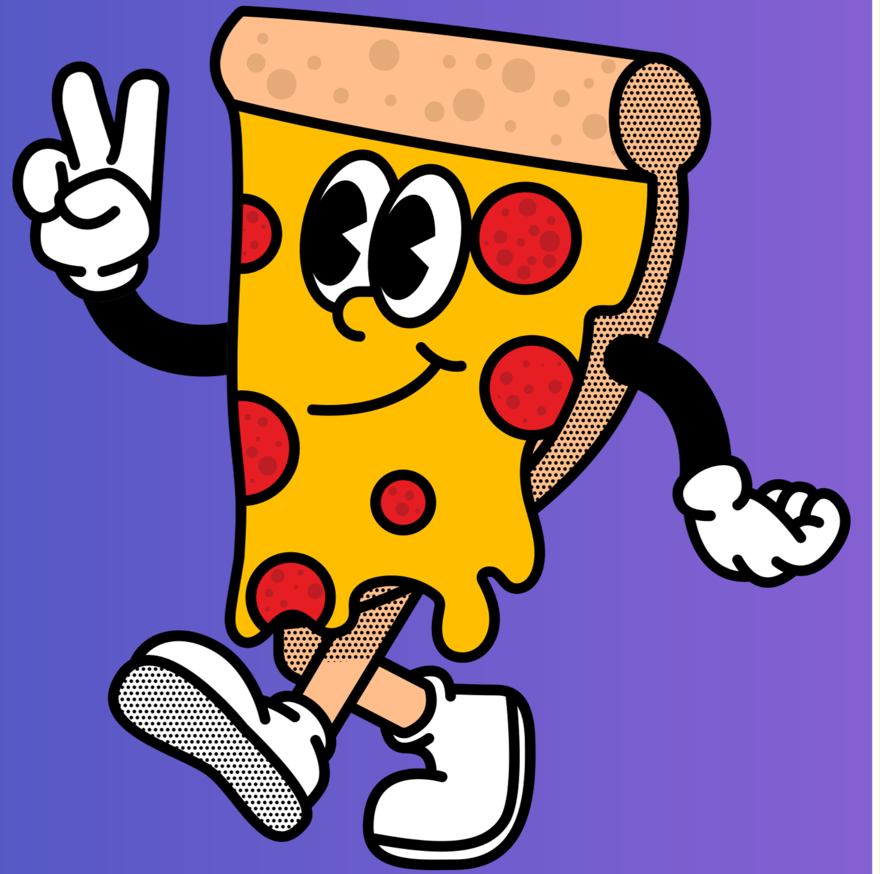
# PIZZA SALES DATA ANALYSIS



# USING SQL

presented by

Anant Kumar Tete

## INTRODUCTION

In this report, I have solved SQL queries from basic to advanced and solved different problems to extract out the relevant data

# QUESTIONS

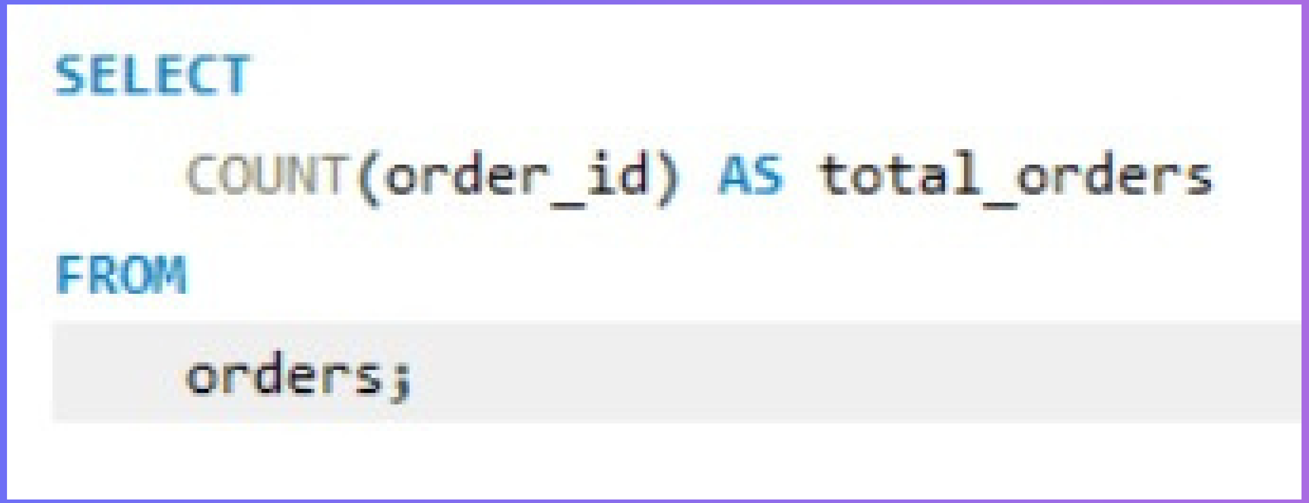
- Basic
- Retrieve the total number of orders placed.
- Calculate the total revenue generated from pizza sales.
- Identify the highest-priced pizza. Identify the most common pizza size ordered.
- List the top 5 most ordered pizza types along with their quantities.

- **Intermediate:**
- Join the necessary tables to find the total quantity of each pizza category ordered.
- Determine the distribution of orders by hour of the day.
- Group the orders by date and calculate the average number of pizzas ordered per day.
- Determine the top 3 most ordered pizza types based on revenue.

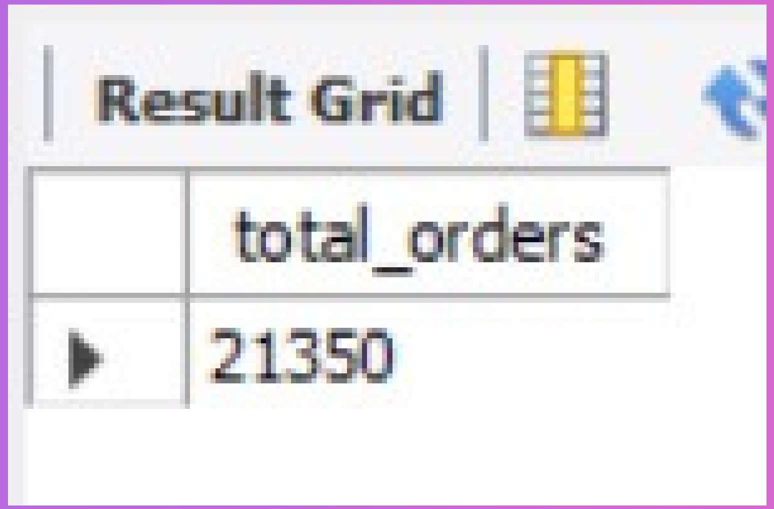
- **Advanced:**
- Calculate the percentage contribution of each pizza type to total revenue.
- Analyze the cumulative revenue generated over time.
- Determine the top 3 most ordered pizza types based on revenue for each pizza category.

## BASIC Q1

Retrieve the total number of orders placed.



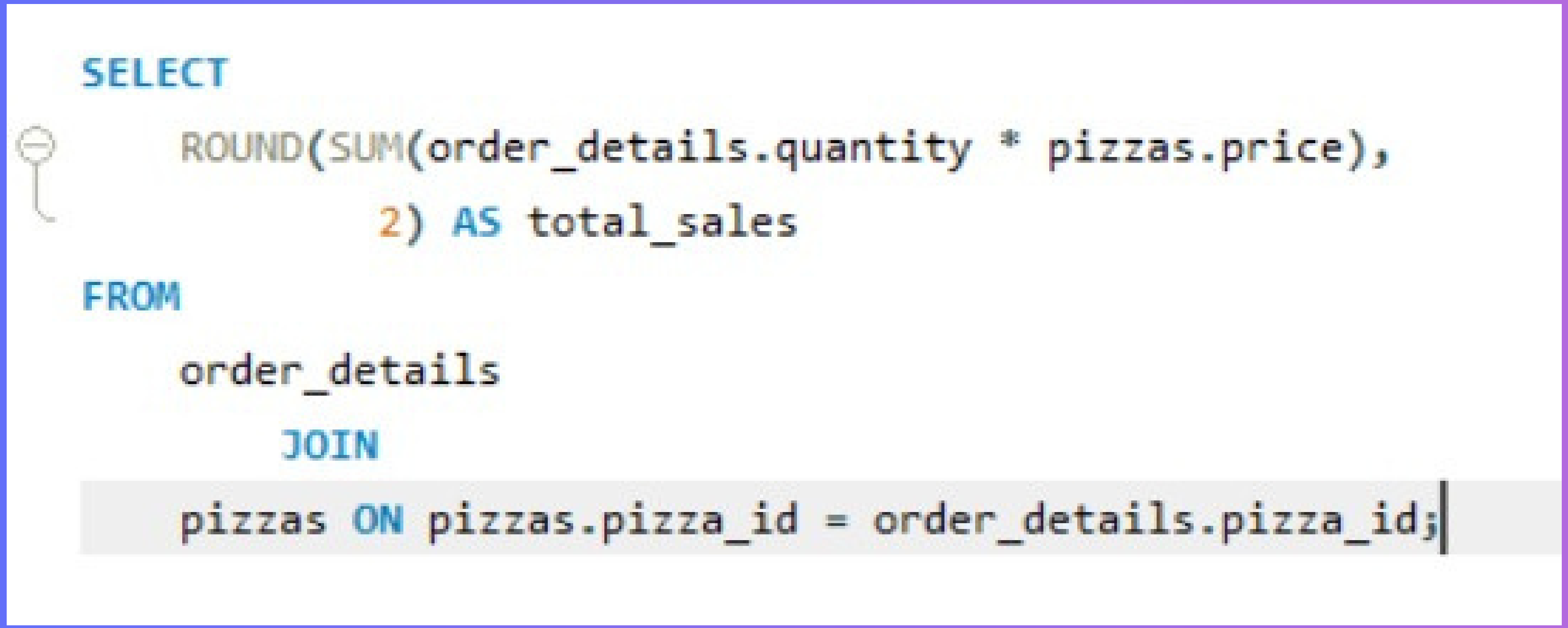
```
SELECT
    COUNT(order_id) AS total_orders
FROM
    orders;
```



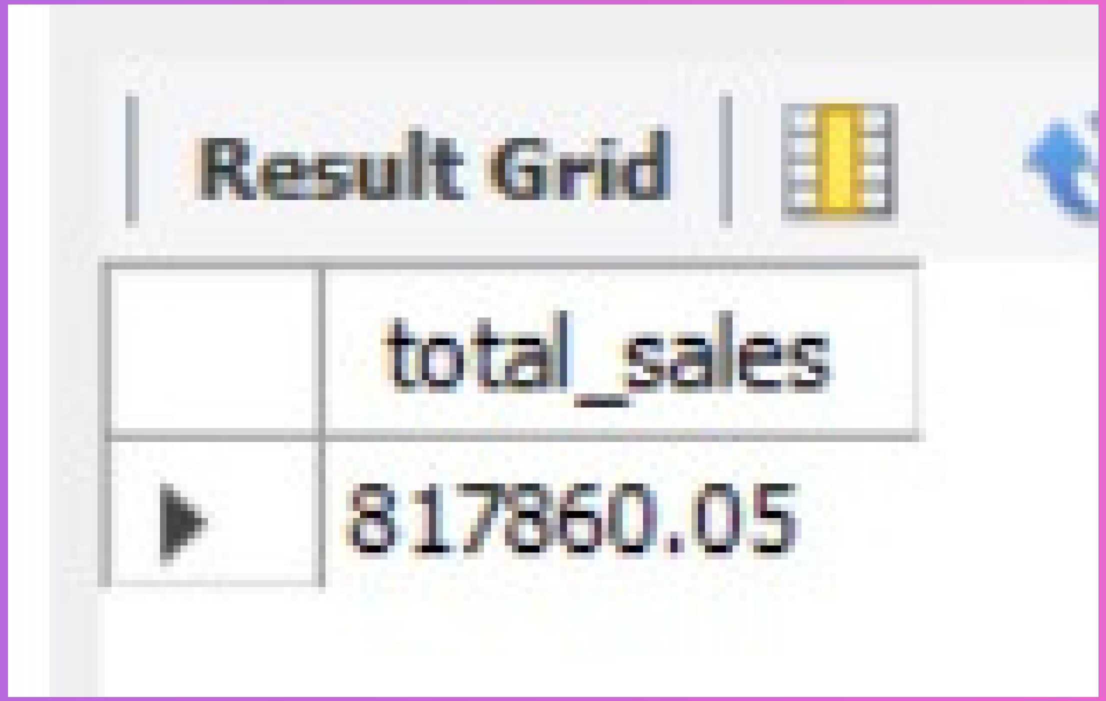
```
Result Grid
   total_orders
▶  21350
```

## BASIC Q2

Calculate the total revenue generated from pizza sales.



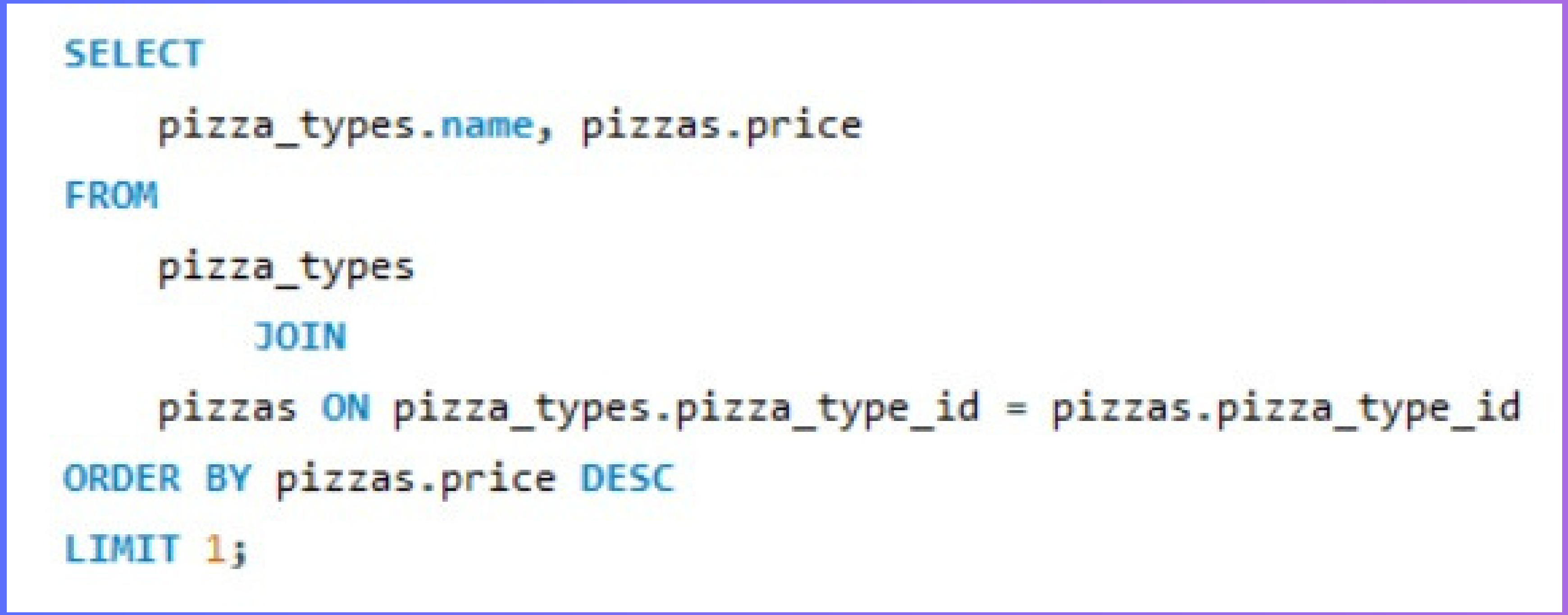
```
SELECT
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS total_sales
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id;
```



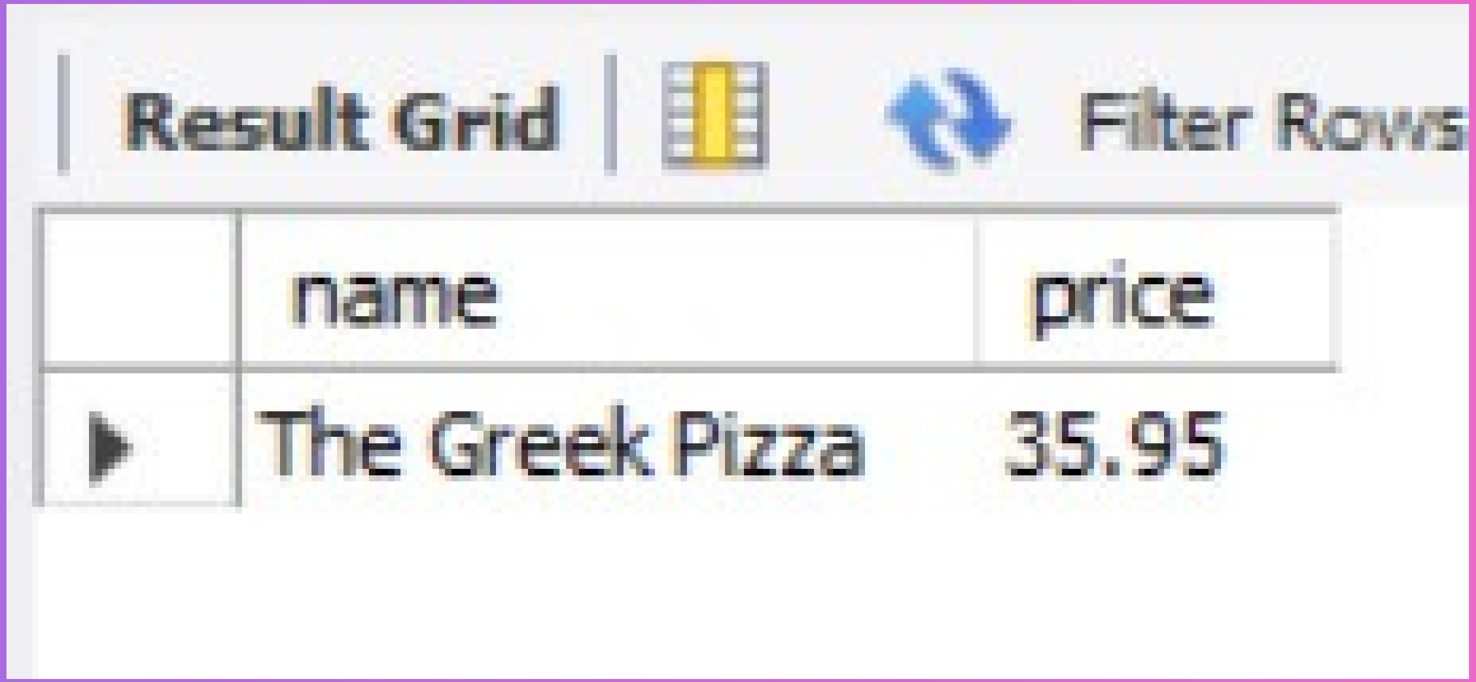
```
Result Grid
   total_sales
▶  817860.05
```

## BASIC Q3

Identify the highest-priced pizza.



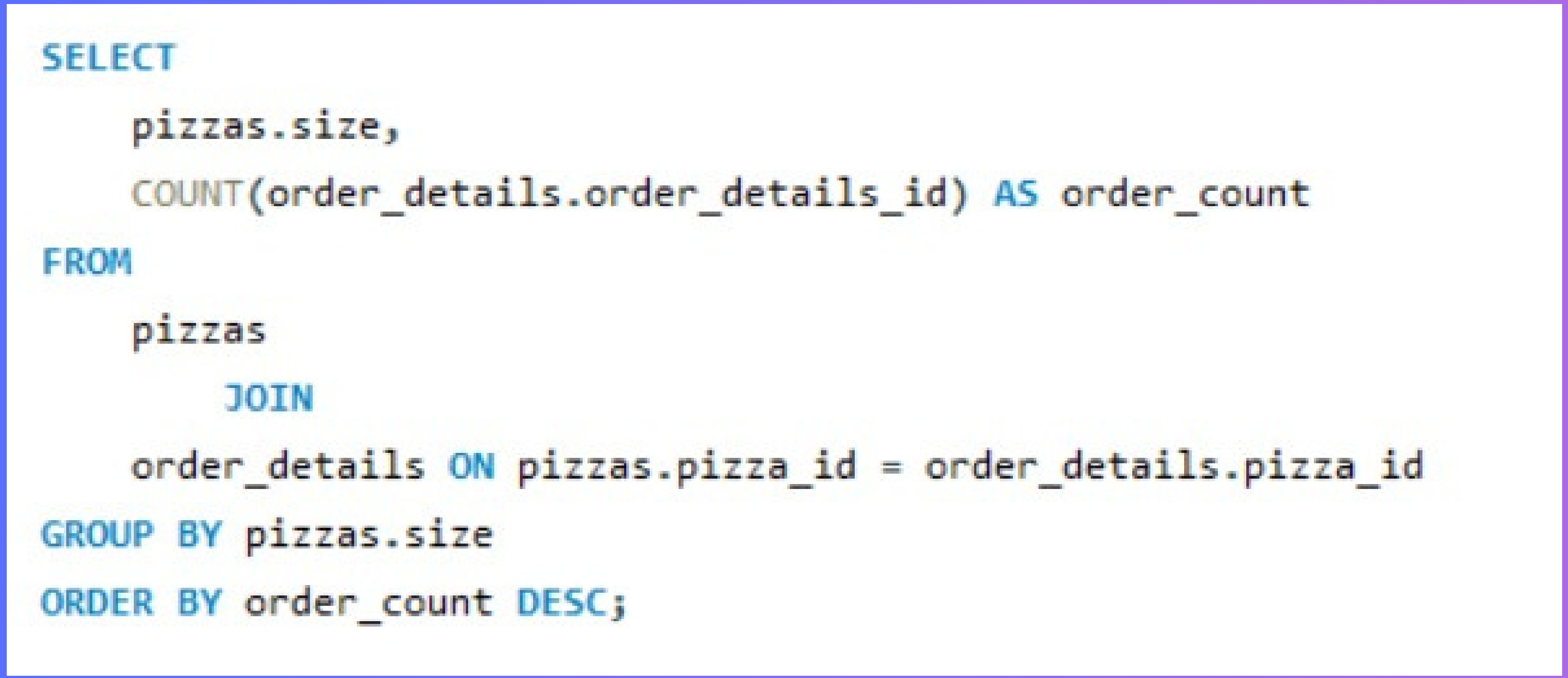
```
SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
```



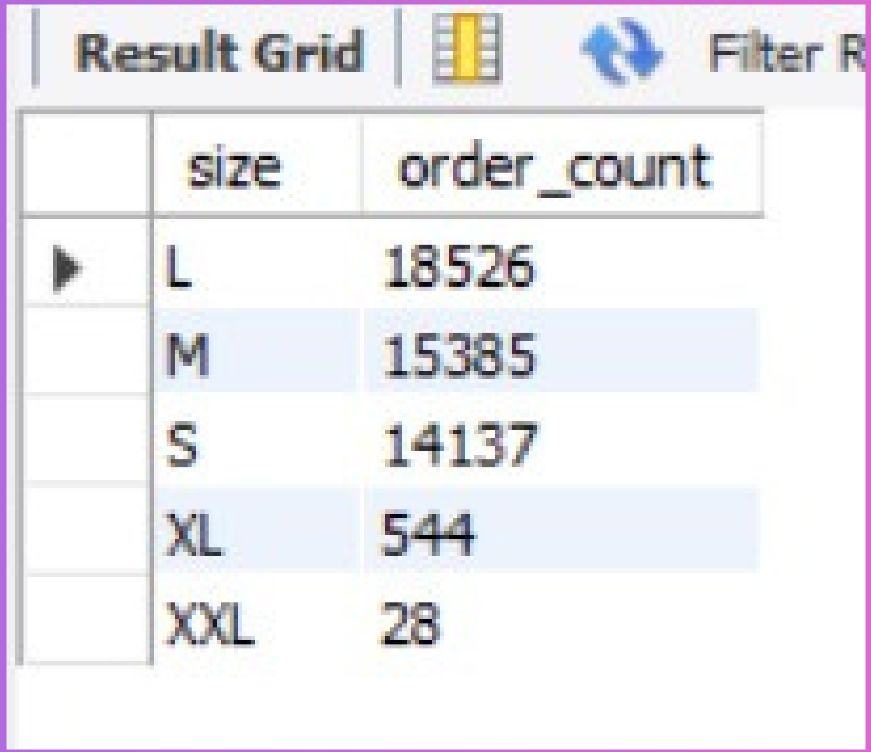
| name | price |
|---|---|
| The Greek Pizza | 35.95 |

## BASIC Q4

Identify the most common pizza size ordered.



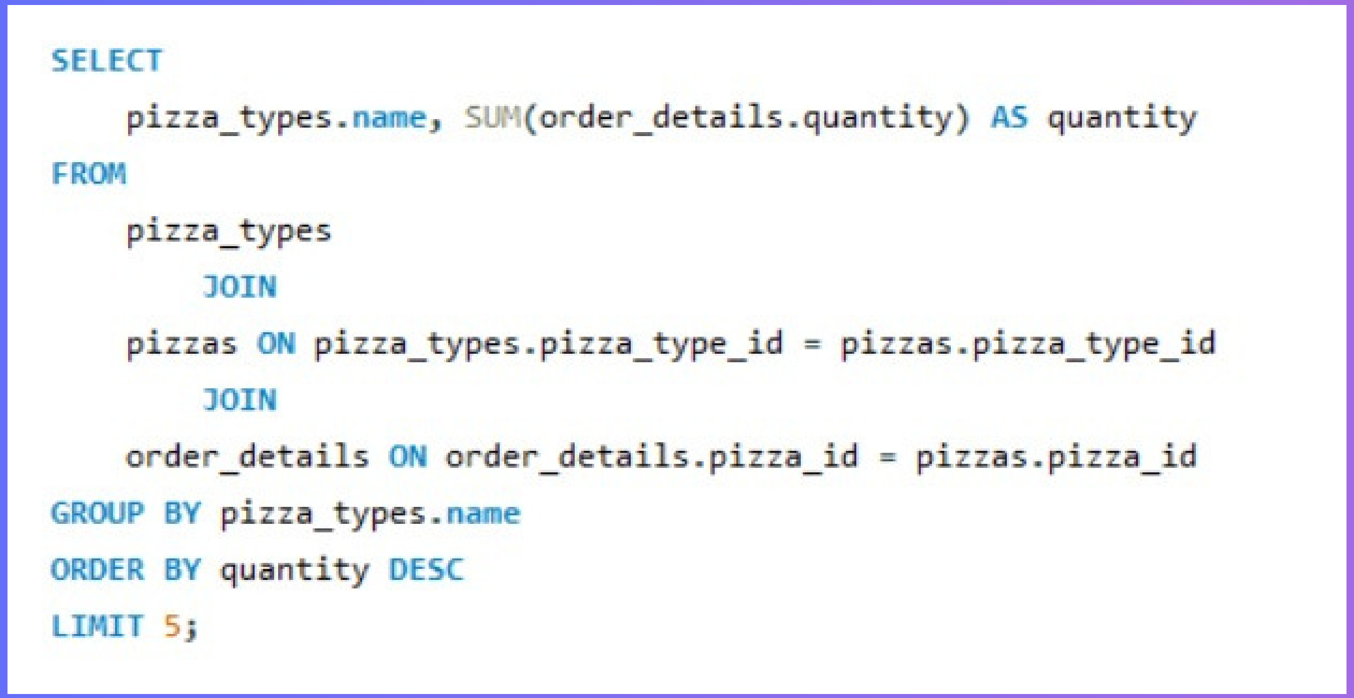
```
SELECT
    pizzas.size,
    COUNT(order_details.order_details_id) AS order_count
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC;
```



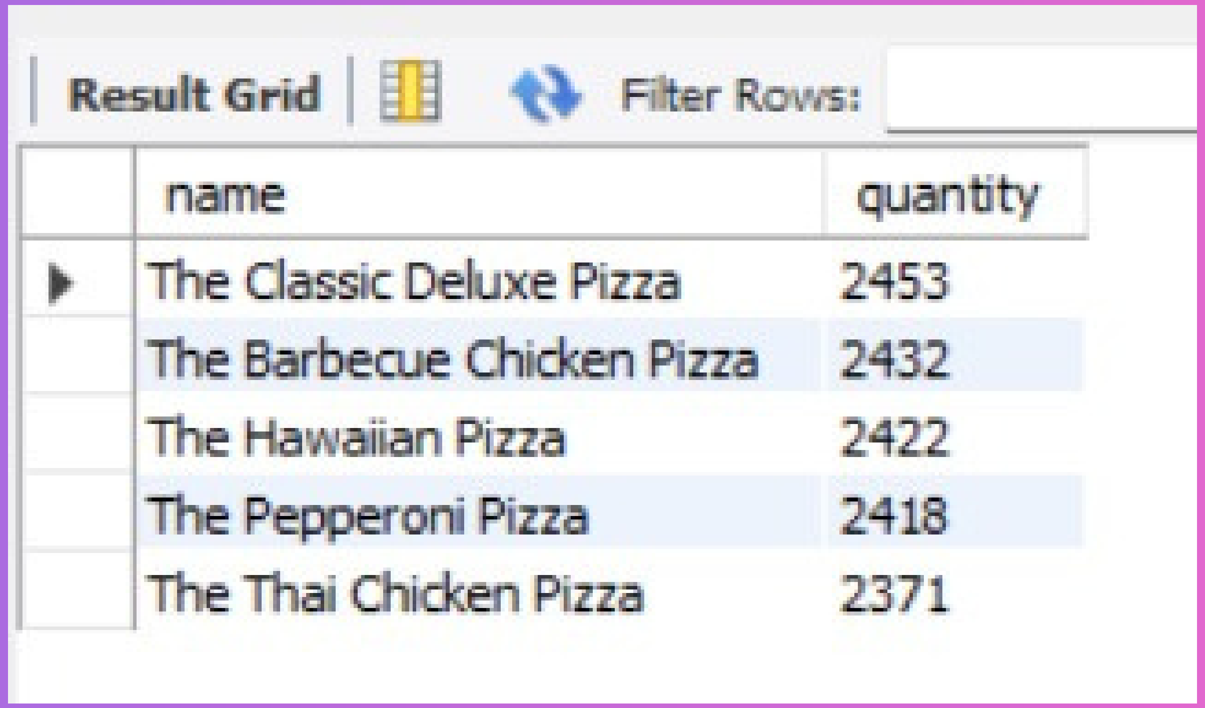
| size | order_count |
|---|---|
| L | 18526 |
| M | 15385 |
| S | 14137 |
| XL | 544 |
| XXL | 28 |

## BASIC Q5

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



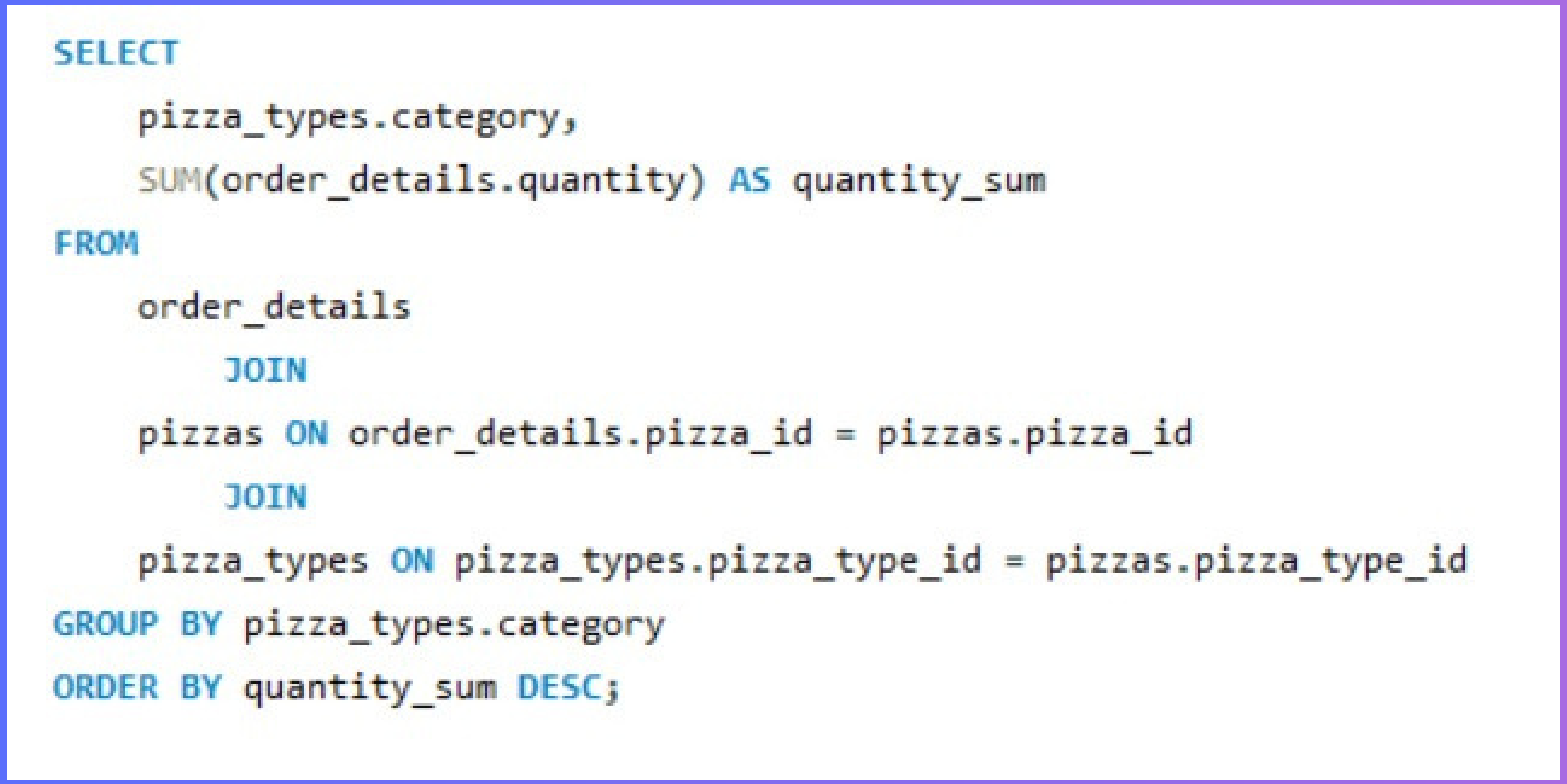
```
SELECT
    pizza_types.name, SUM(order_details.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY quantity DESC
LIMIT 5;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



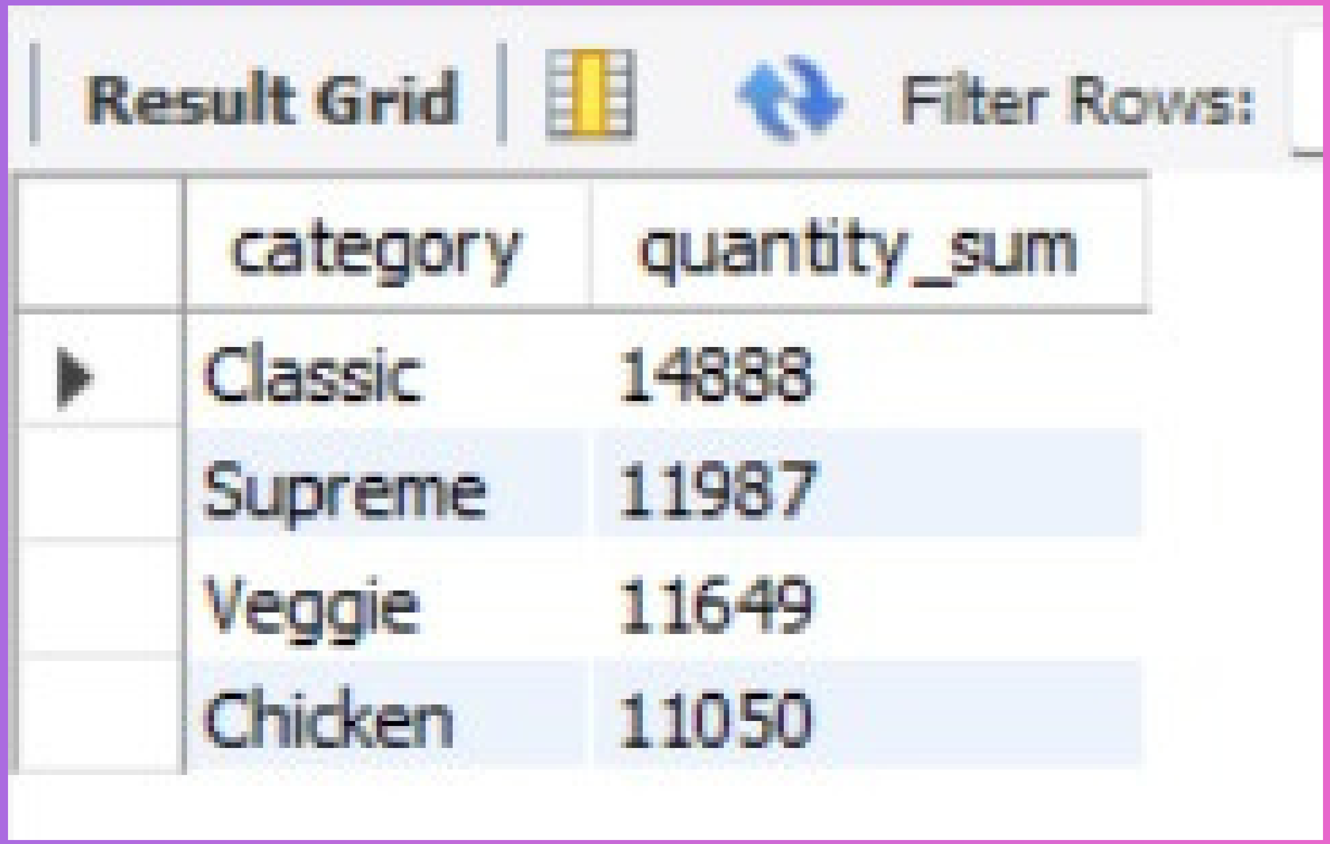
| name | quantity |
|---|---|
| The Classic Deluxe Pizza | 2453 |
| The Barbecue Chicken Pizza | 2432 |
| The Hawaiian Pizza | 2422 |
| The Pepperoni Pizza | 2418 |
| The Thai Chicken Pizza | 2371 |

## INTERMEDIATE Q1

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



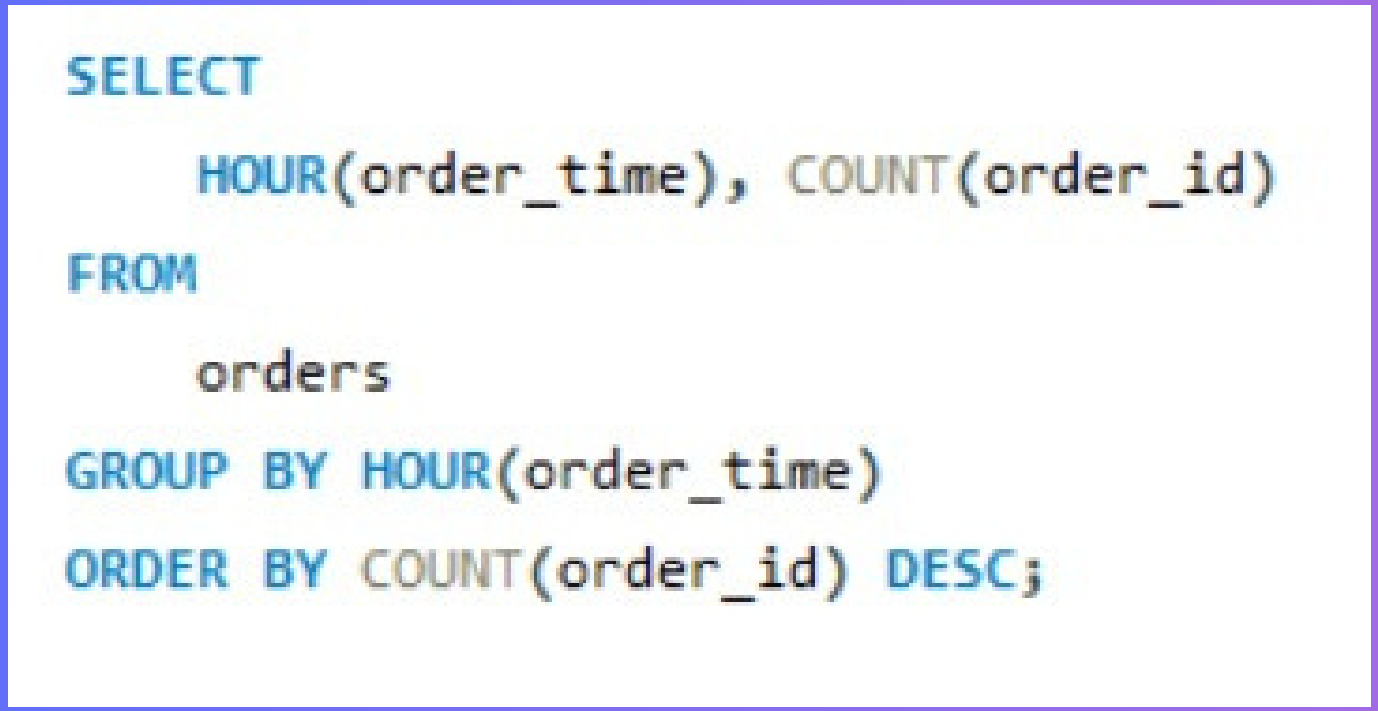
```
SELECT
    pizza_types.category,
    SUM(order_details.quantity) AS quantity_sum
FROM
    order_details
        JOIN
    pizzas ON order_details.pizza_id = pizzas.pizza_id
        JOIN
    pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY pizza_types.category
ORDER BY quantity_sum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



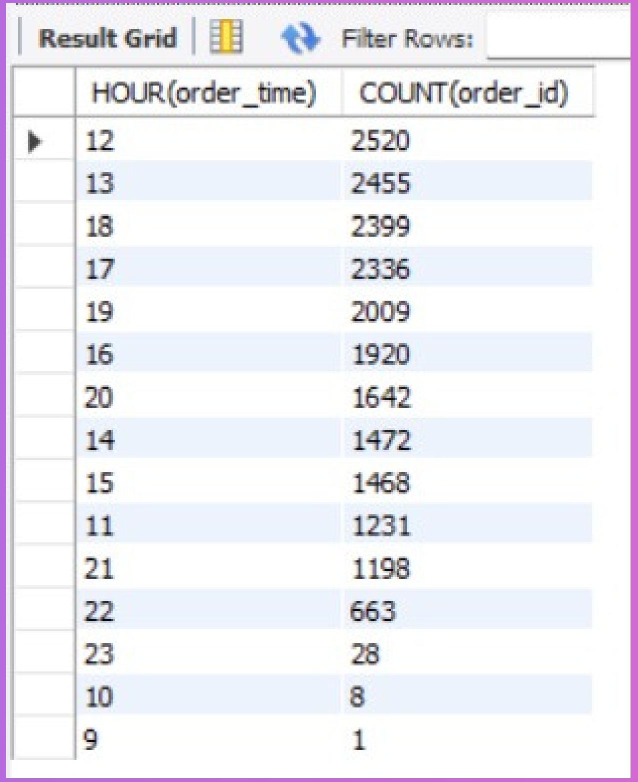
| category | quantity_sum |
|---|---|
| Classic | 14888 |
| Supreme | 11987 |
| Veggie | 11649 |
| Chicken | 11050 |

## INTERMEDIATE Q2

Determine the distribution of orders by hour of the day.



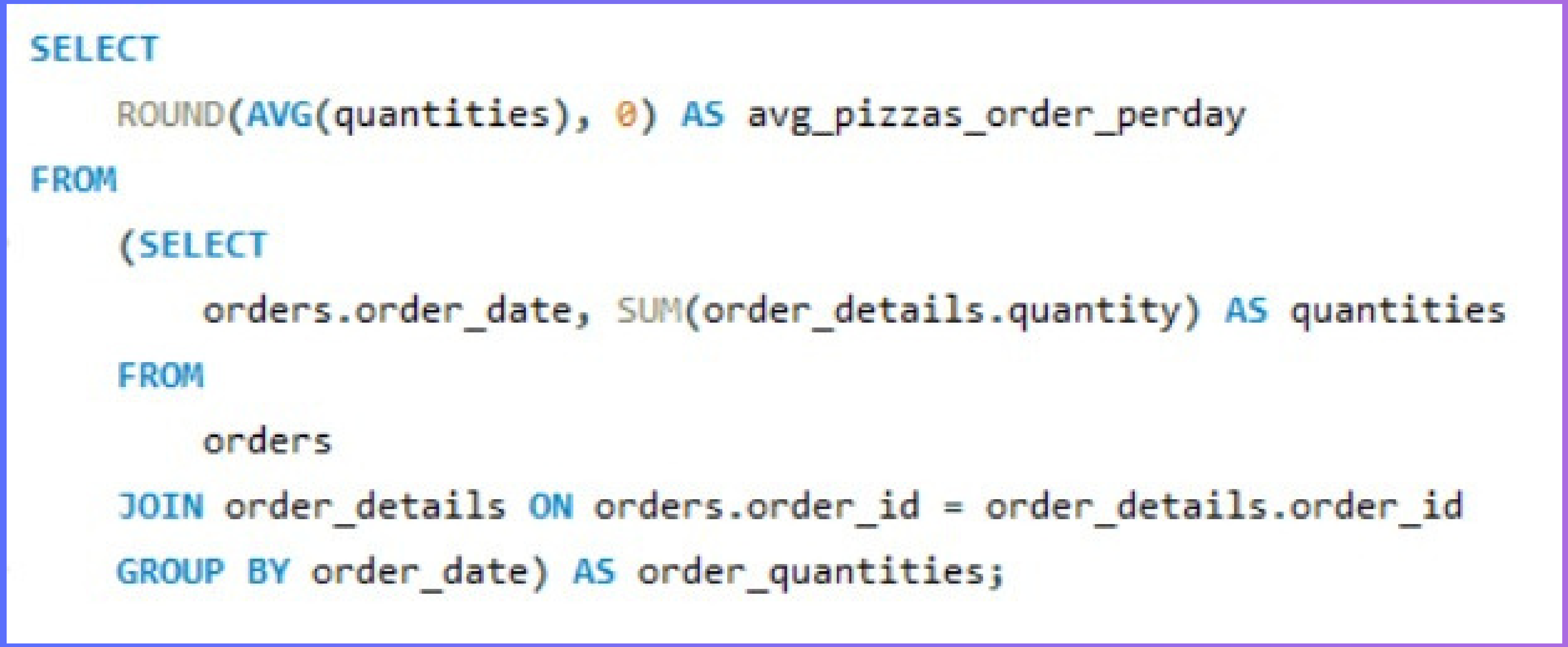
```
SELECT
    HOUR(order_time), COUNT(order_id)
FROM
    orders
GROUP BY HOUR(order_time)
ORDER BY COUNT(order_id) DESC;
```



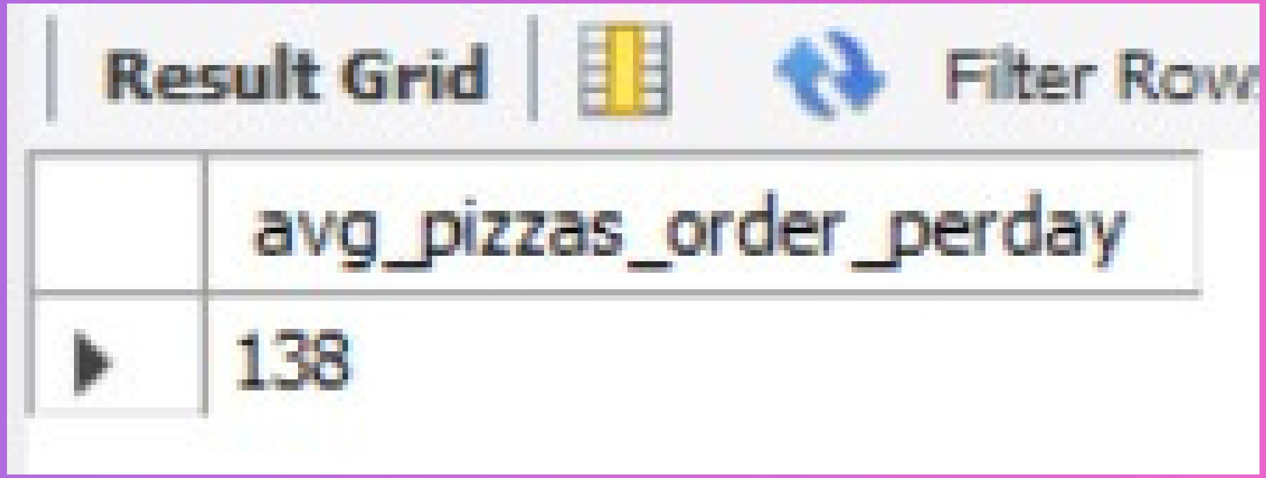
| HOUR(order_time) | COUNT(order_id) |
|---|---|
| 12 | 2520 |
| 13 | 2455 |
| 18 | 2399 |
| 17 | 2336 |
| 19 | 2009 |
| 16 | 1920 |
| 20 | 1642 |
| 14 | 1472 |
| 15 | 1468 |
| 11 | 1231 |
| 21 | 1198 |
| 22 | 663 |
| 23 | 28 |
| 10 | 8 |
| 9 | 1 |

## INTERMEDIATE Q3

Group the orders by date and calculate the average number of pizzas ordered per day.



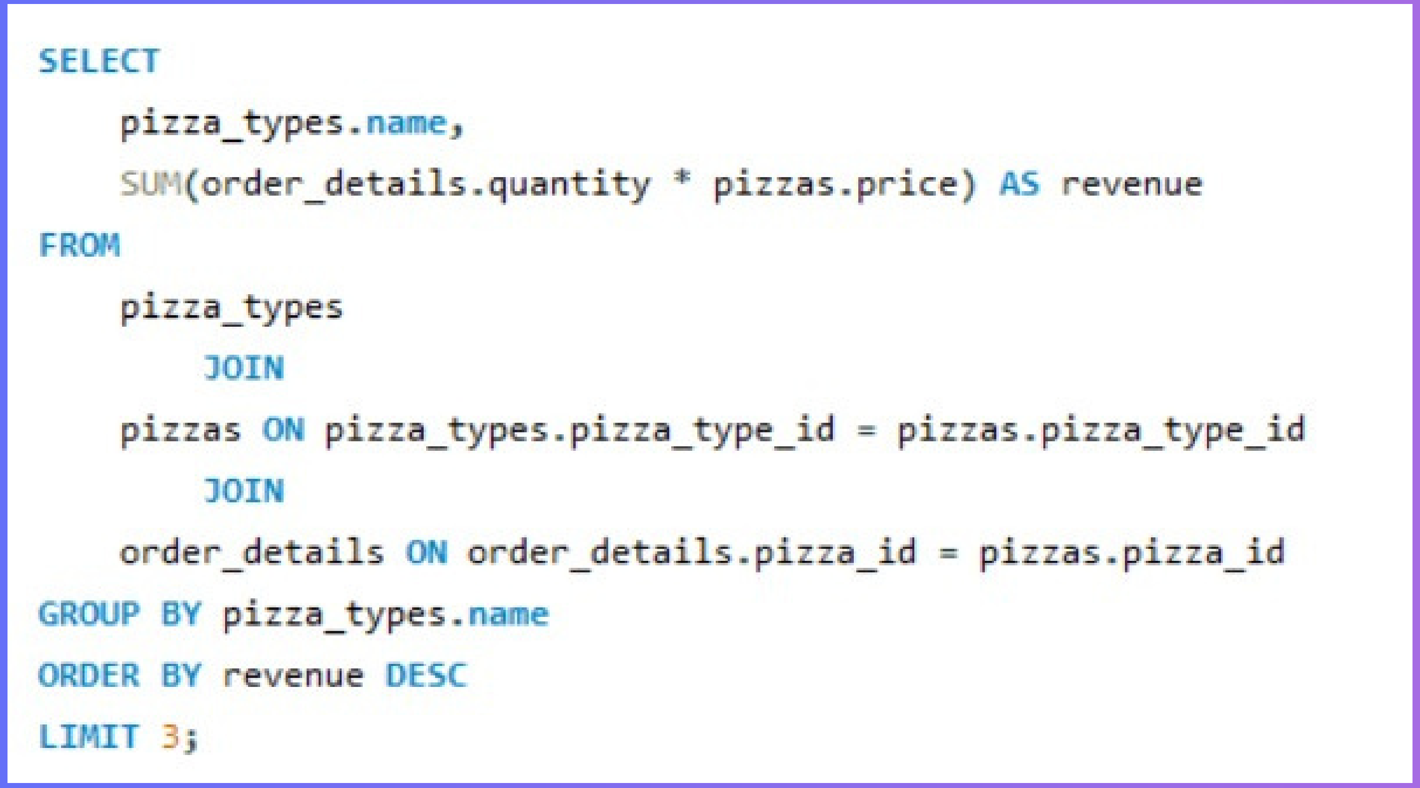
```
SELECT
    ROUND(AVG(quantities), 0) AS avg_pizzas_order_perday
FROM
    (SELECT
        orders.order_date, SUM(order_details.quantity) AS quantities
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY order_date) AS order_quantities;
```



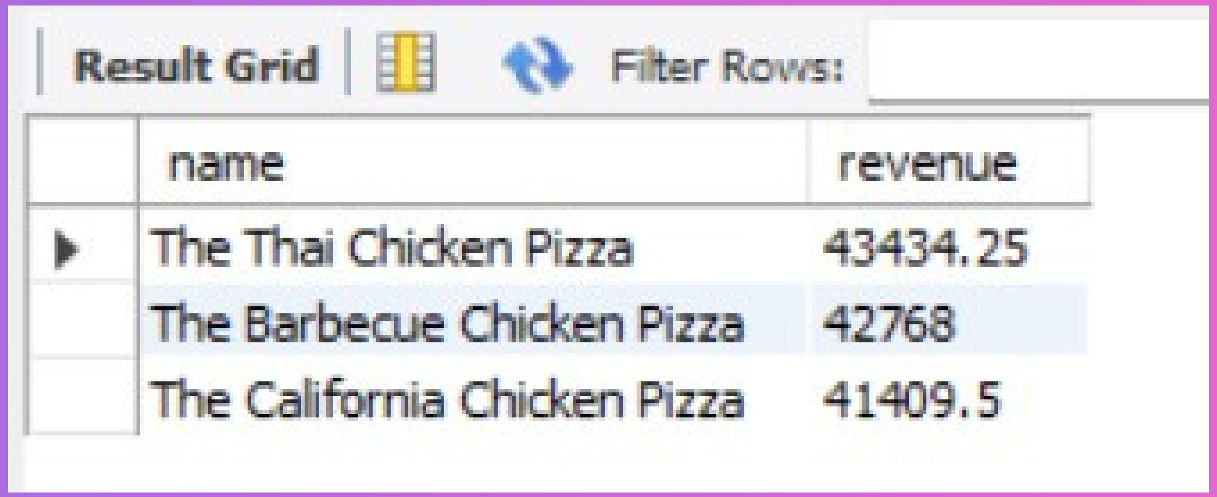
| avg_pizzas_order_perday |
|---|
| 138 |

## INTERMEDIATE Q4

Determine the top 3 most ordered pizza types based on revenue.



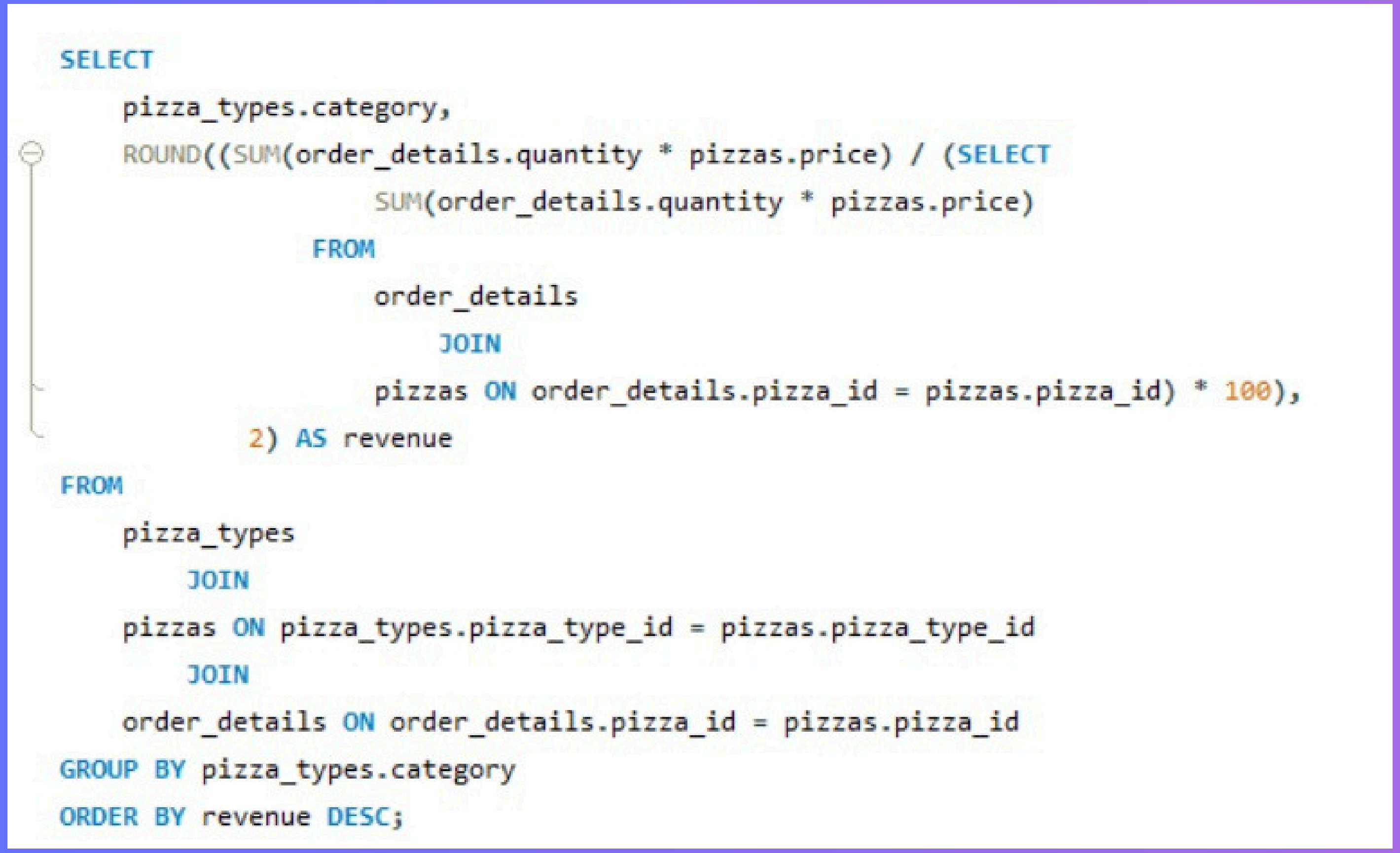
```
SELECT
    pizza_types.name,
    SUM(order_details.quantity * pizzas.price) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue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



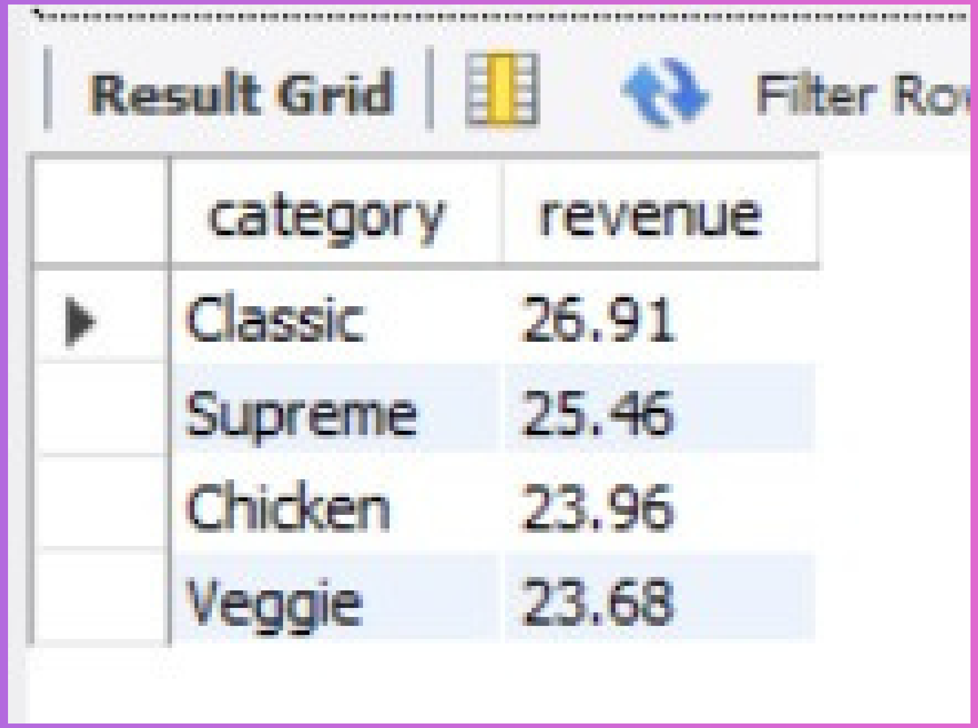
| name | revenue |
|---|---|
| The Thai Chicken Pizza | 43434.25 |
| The Barbecue Chicken Pizza | 42768 |
| The California Chicken Pizza | 41409.5 |

## ADVANCE Q1

Calculate the percentage contribution of each pizza type to total revenue.



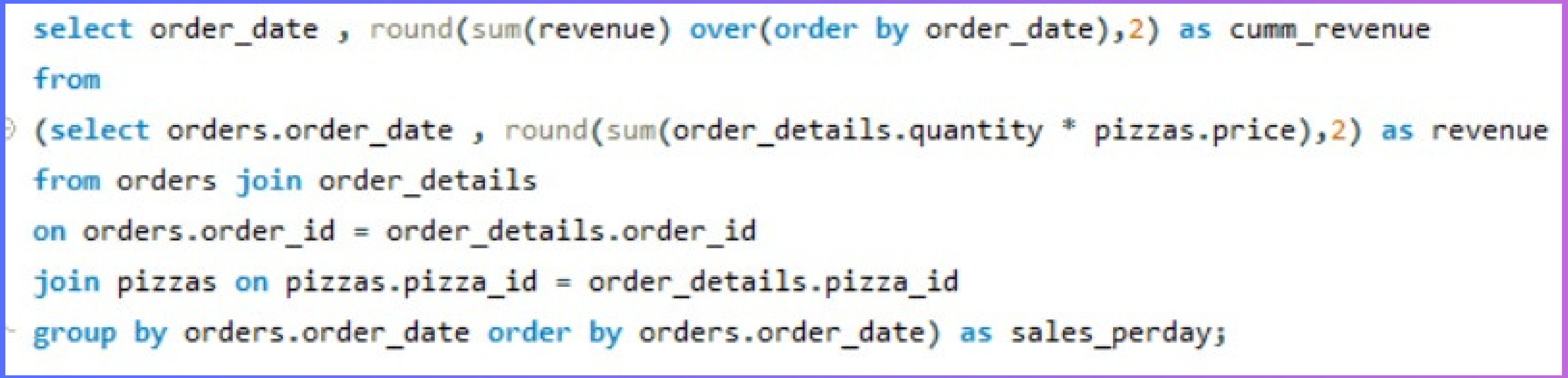
```
SELECT
    pizza_types.category,
    ROUND((SUM(order_details.quantity * pizzas.price) / (SELECT
                SUM(order_details.quantity * pizzas.price)
            FROM
                order_details
                    JOIN
                pizzas ON order_details.pizza_id = pizzas.pizza_id) *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



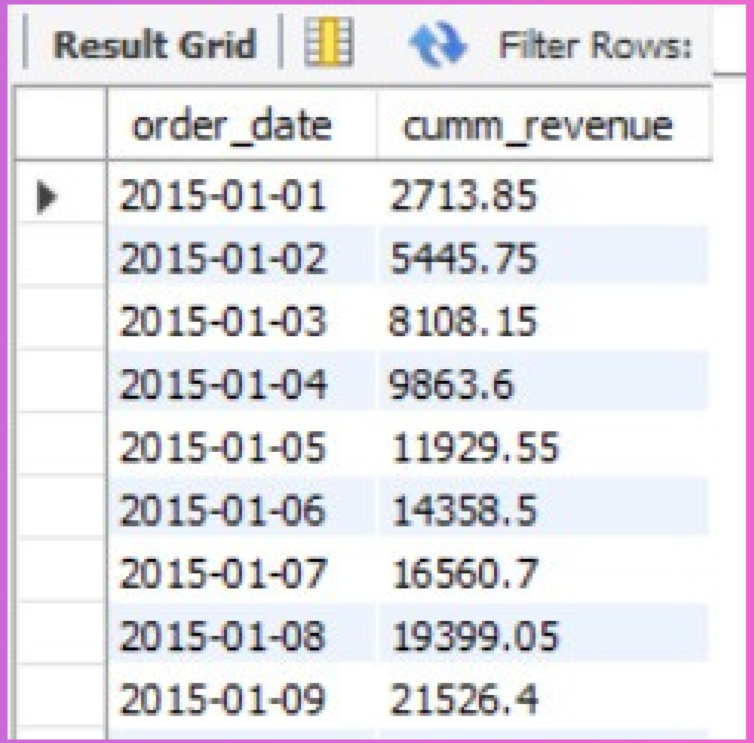
| category | revenue |
|---|---|
| Classic | 26.91 |
| Supreme | 25.46 |
| Chicken | 23.96 |
| Veggie | 23.68 |

## ADVANCE Q2

Analyze the cumulative revenue generated over time.



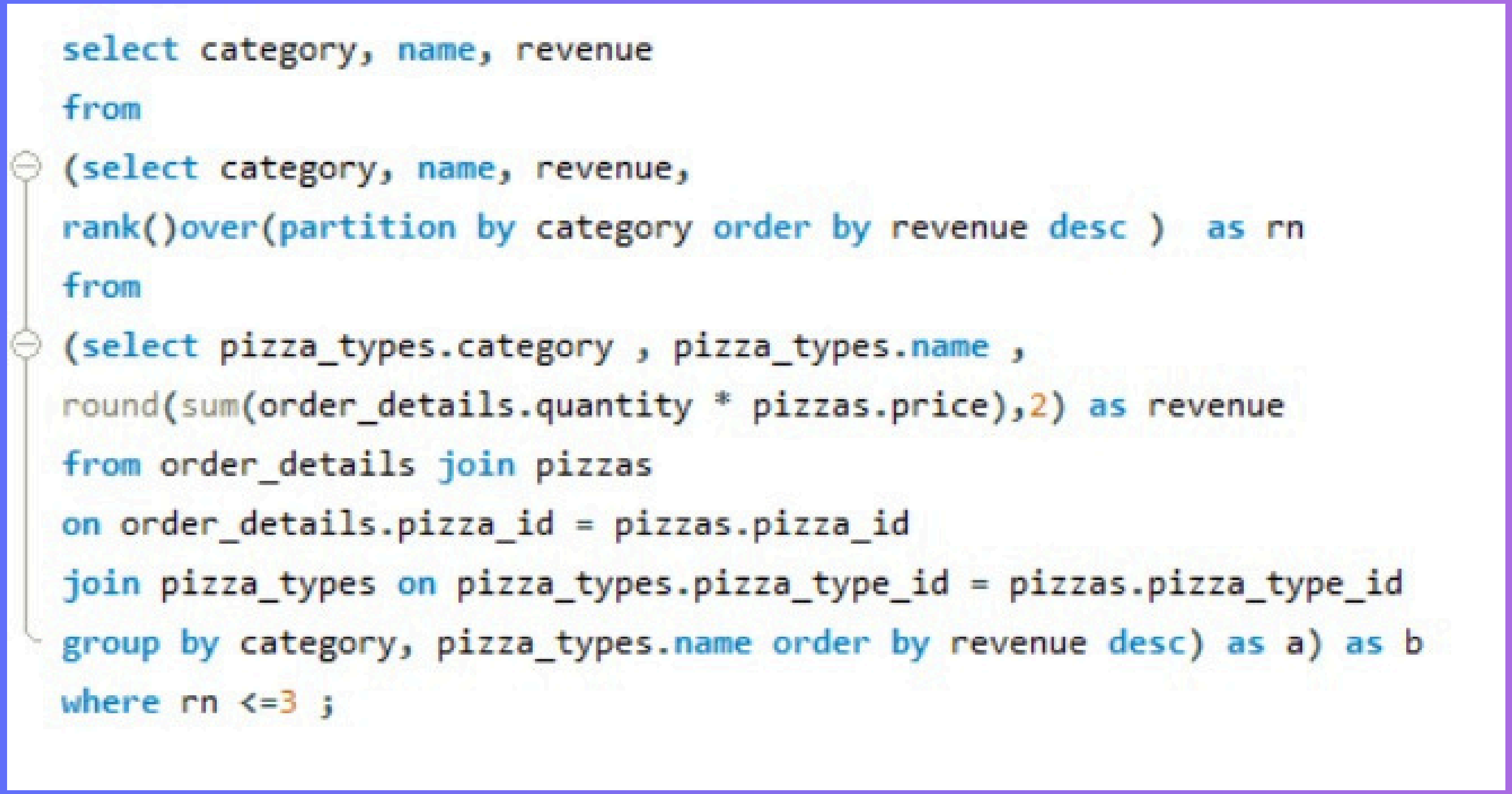
```sql
select order_date , round(sum(revenue) over(order by order_date),2) as cumm_revenue
from
(select orders.order_date , round(sum(order_details.quantity * pizzas.price),2) as revenue
from orders join order_details
on orders.order_id = order_details.order_id
join pizzas on pizzas.pizza_id = order_details.pizza_id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



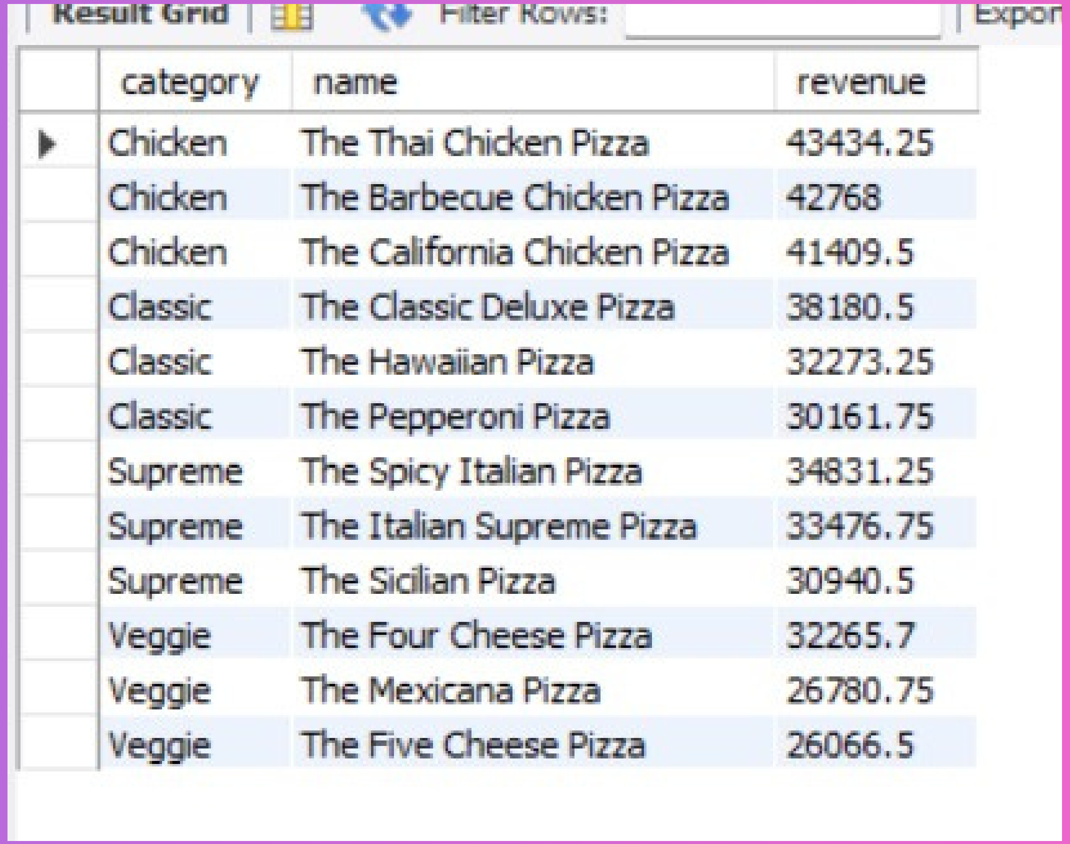
```
order_date   cumm_revenue
2015-01-01   2713.85
2015-01-02   5445.75
2015-01-03   8108.15
2015-01-04   9863.6
2015-01-05   11929.55
2015-01-06   14358.5
2015-01-07   16560.7
2015-01-08   19399.05
2015-01-09   21526.4
```

## ADVANCE Q3

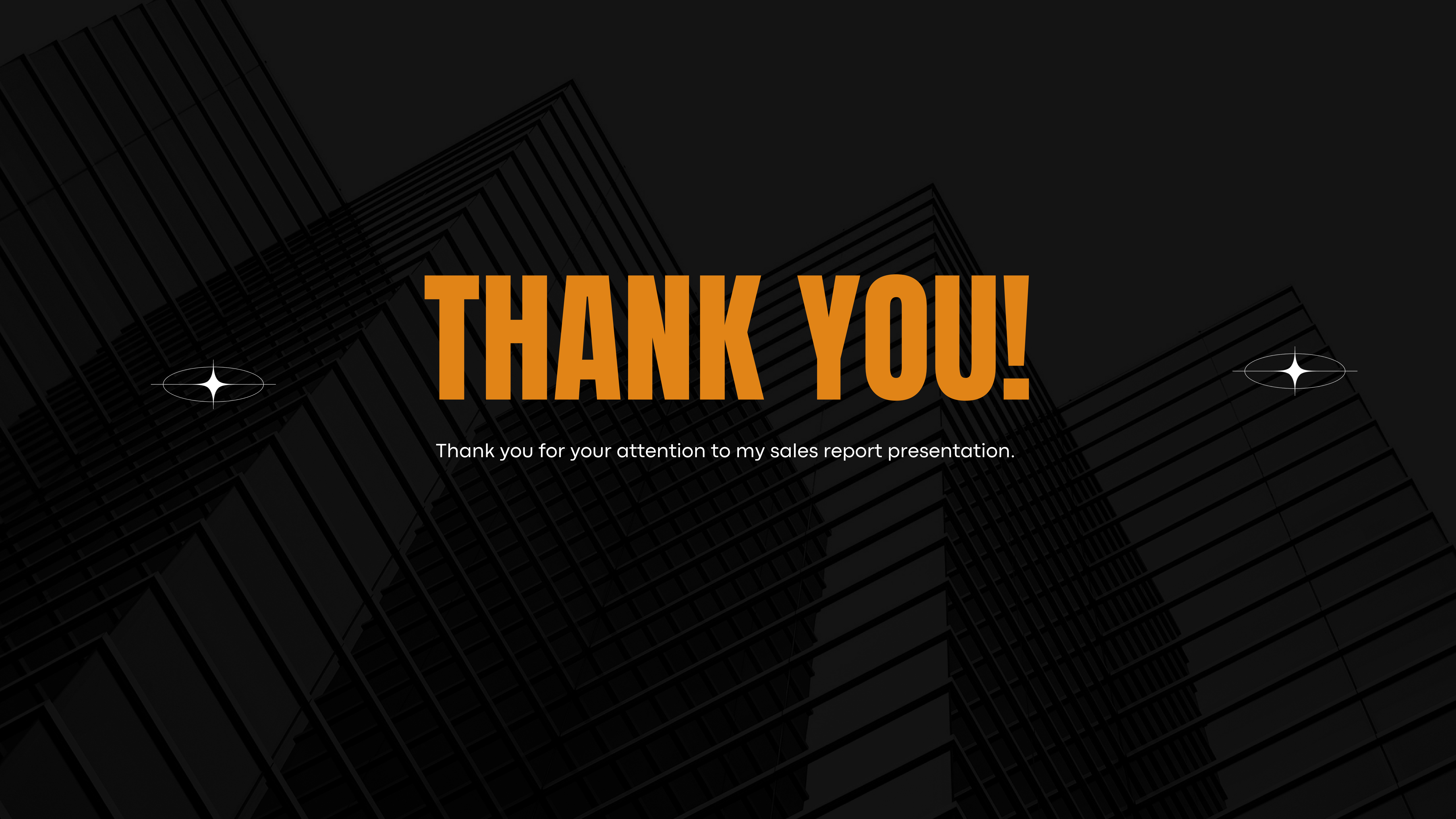
Determine the top 3 most ordered pizza types based on revenue for each pizza category.



```sql
select category, 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 ,
round(sum(order_details.quantity * pizzas.price),2) as revenue
from order_details join pizzas
on order_details.pizza_id = pizzas.pizza_id
join pizza_types on pizza_types.pizza_type_id = pizzas.pizza_type_id
group by category, pizza_types.name order by revenue desc) as a) as b
where rn <=3 ;
```



```
category   name                           revenue
Chicken    The Thai Chicken Pizza         43434.25
Chicken    The Barbecue Chicken Pizza     42768
Chicken    The California Chicken Pizza   41409.5
Classic    The Classic Deluxe Pizza       38180.5
Classic    The Hawaiian Pizza             32273.25
Classic    The Pepperoni Pizza            30161.75
Supreme    The Spicy Italian Pizza        34831.25
Supreme    The Italian Supreme Pizza      33476.75
Supreme    The Sicilian Pizza             30940.5
Veggie     The Four Cheese Pizza          32265.7
Veggie     The Mexicana Pizza             26780.75
Veggie     The Five Cheese Pizza          26066.5
```



# THANK YOU!

Thank you for your attention to my sales report presentation.