WELCOME TO

Pizza Sales Analysis using SQL

LARANA PIZZA

INTRODUCTION TO PROJECT

- This project analyzes the sales data of a pizza restaurant using only SQL.
- Tools used: MySQL Workbench
- Database: pizzahut (4 main tables – orders, order_details, pizzas, pizza_types)
- No Excel, no Power BI — pure SQL-based insights!

LARANA PIZZA

PROJECT OBJECTIVE

- Understand sales performance and customer behavior.
- Identify top-performing pizza types and categories.
- Analyze revenue trends and pizza preferences.
- Help business decisions using SQL data insights.

DATABASE SCHEMA

ORDERS --------< ORDER_DETAILS >-------- PIZZAS --------< PIZZA_TYPES

orders (order_id, order_date, order_time)

order_details (order_id, pizza_id, quantity)

pizzas (pizza_id, pizza_type_id, size, price)

pizza_types (pizza_type_id, name, category, ingredients)

ANALYSIS SECTIONS

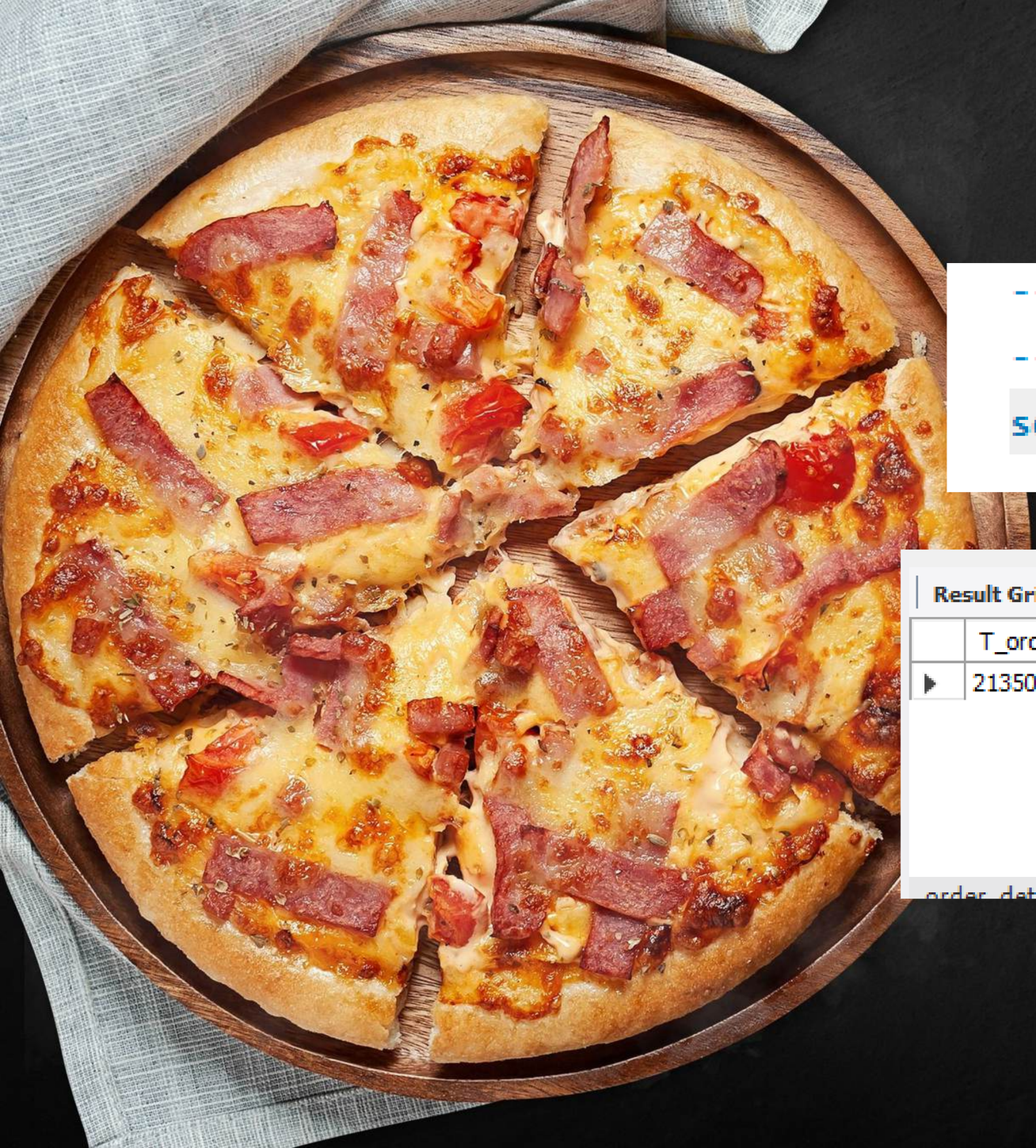
📌BASIC INSIGHTS

- ✅ Total Orders Placed
- ✅ Total Revenue
- ✅ Highest-Priced Pizza
- ✅ Most Common Pizza Size
- ✅ Top 5 Most Ordered Pizzas

ANALYSIS SECTIONS

📌INTERMEDIATE INSIGHTS

- ✅ TOTAL QUANTITY ORDERED PER PIZZA CATEGORY
- ✅ HOUR-WISE ORDER DISTRIBUTION
- ✅ ORDERS GROUPED BY DAY AND AVG. QUANTITY
- ✅ CATEGORY-WISE PIZZA DISTRIBUTION



```sql
-- Basic:

-- Retrieve the total number of orders placed.
select count(order_id) as T_order from orders;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

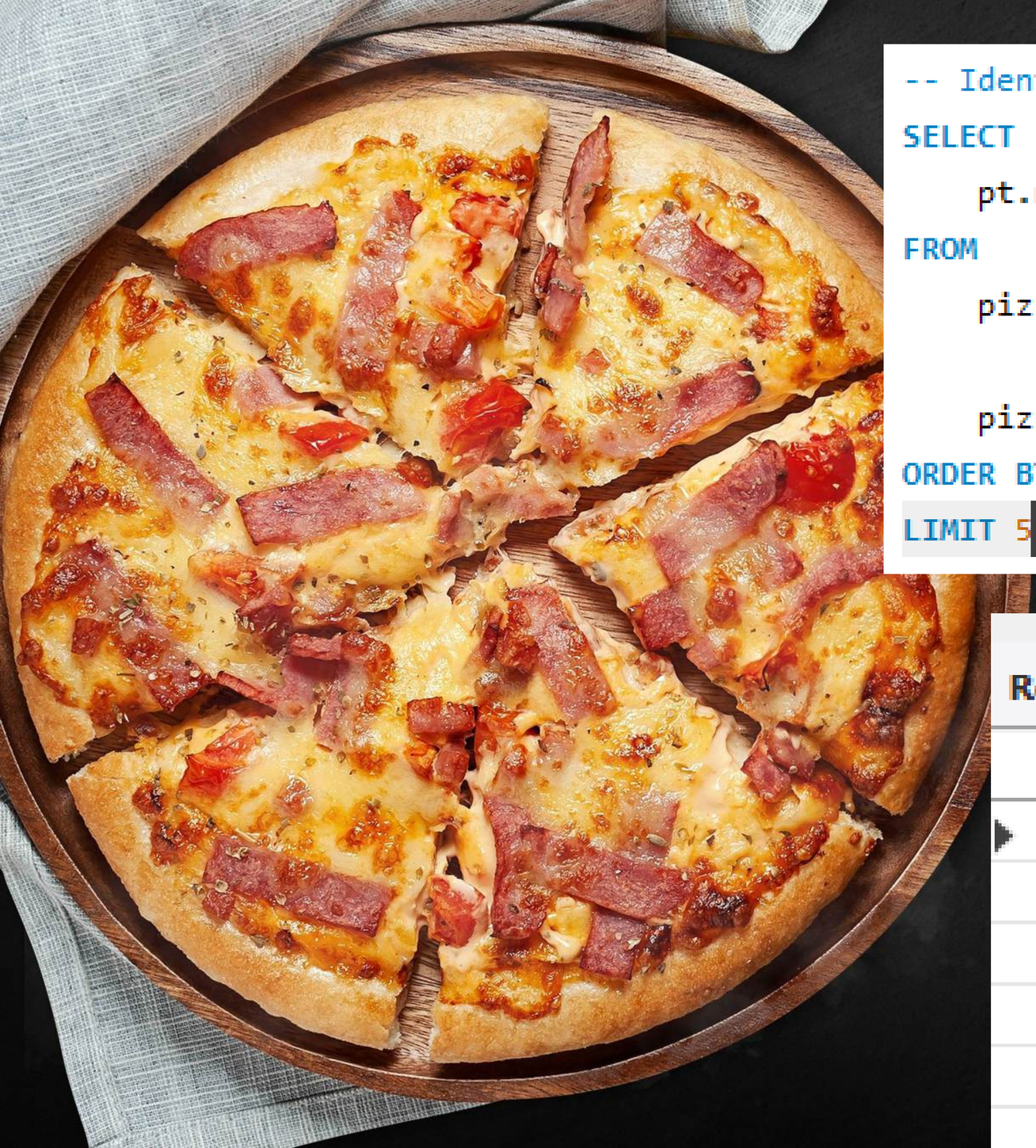
	T_order
▶	21350

order_details 153 | orders 154 | pizza_types 155 | pizzas 156 | Result 157

```sql
-- Calculate the total revenue generated from pizza sales.
SELECT
    ROUND(SUM(od.quantity * p.price), 0) AS t_revenue
FROM
    order_details od
        JOIN
    pizzas p ON od.pizza_id = p.pizza_id;
```

Result Grid

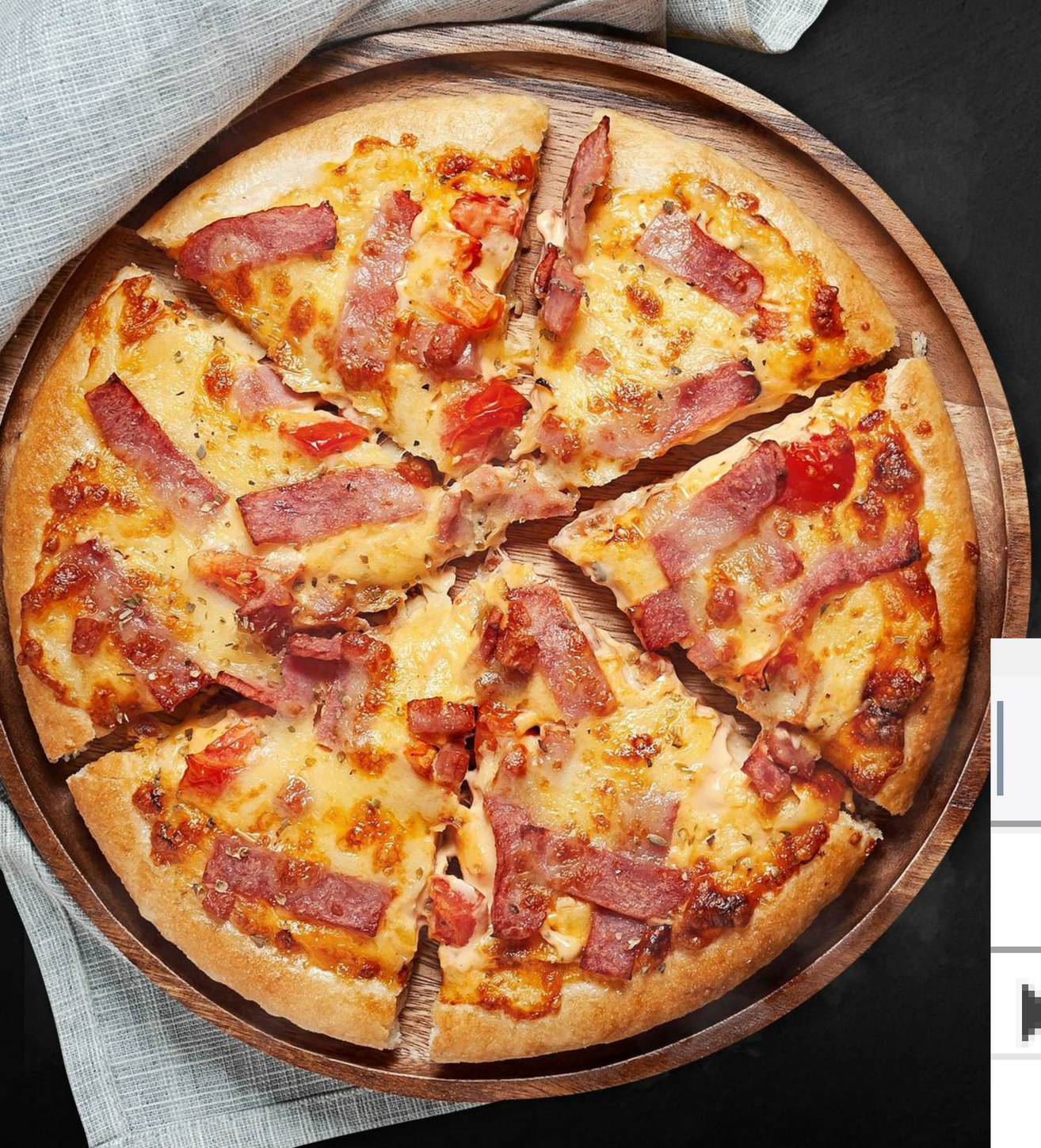
	t_revenue
▶	817860



```sql
-- Identify the highest-priced pizza.
SELECT
    pt.name, (p.price)
FROM
    pizza_types pt
        JOIN
    pizzas p ON pt.pizza_type_id = p.pizza_type_id
ORDER BY p.price DESC
LIMIT 5
```

Result Grid | Filter Rows:

name	price
The Greek Pizza	35.95
The Greek Pizza	25.5
The Brie Carre Pizza	The Brie Carre
The Italian Vegetables Pizza	21
The Barbecue Chicken Pizza	20.75

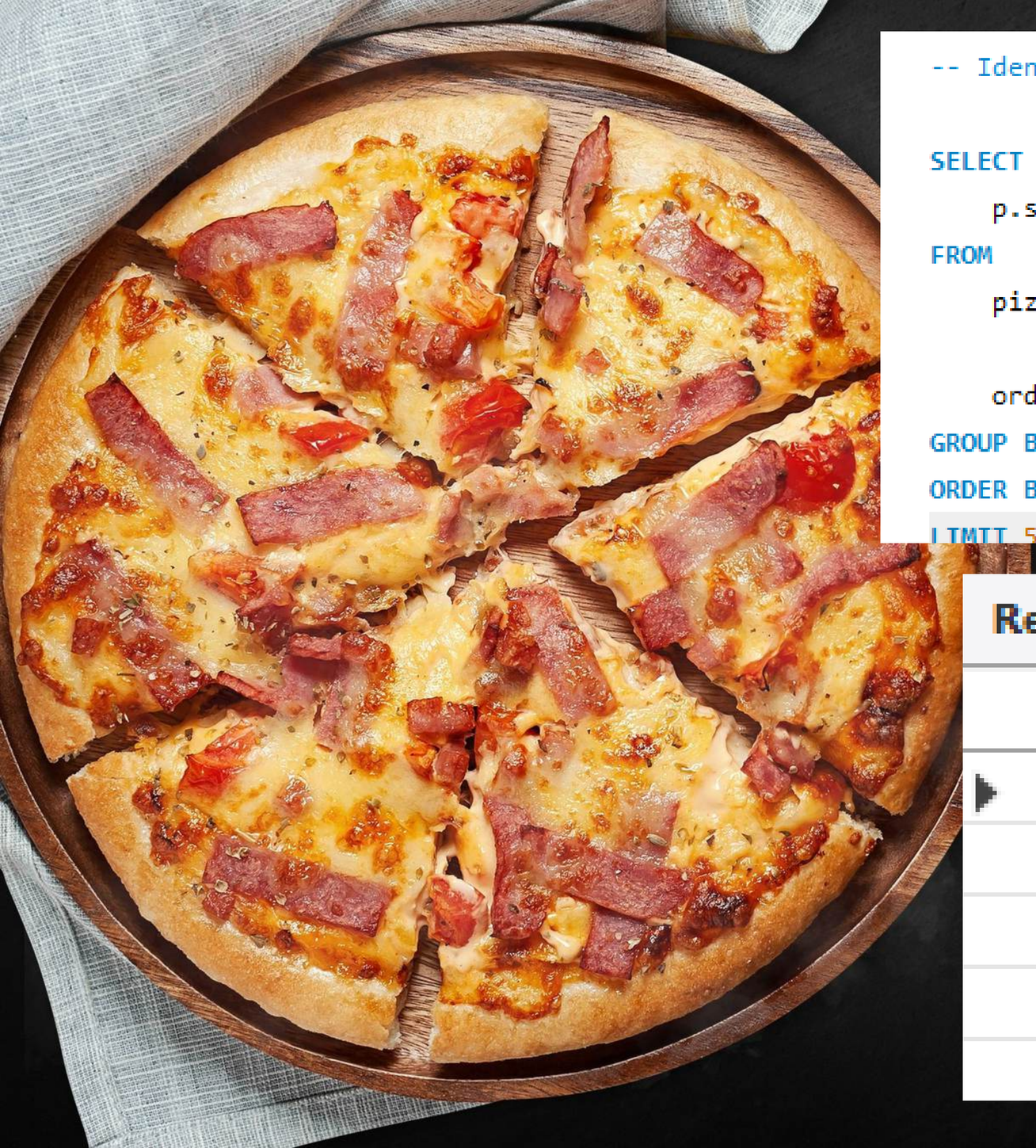


```sql
-- Identify the highest-priced pizza.

SELECT
    MAX(price)
FROM
    pizzas;
SELECT
    MAX(Price) AS max_pr_pizzas
FROM
    pizzas;
```

Result Grid

	max_pr_pizzas
▶	35.95



```sql
-- Identify the most common pizza size ordered.

SELECT
    p.size AS p_size, SUM(od.quantity) AS total_quantity
FROM
    pizzas p
        JOIN
    order_details od ON p.pizza_id = od.pizza_id
GROUP BY p.size
ORDER BY total_quantity DESC
LIMIT 5
```

Result Grid | Filter Rows:

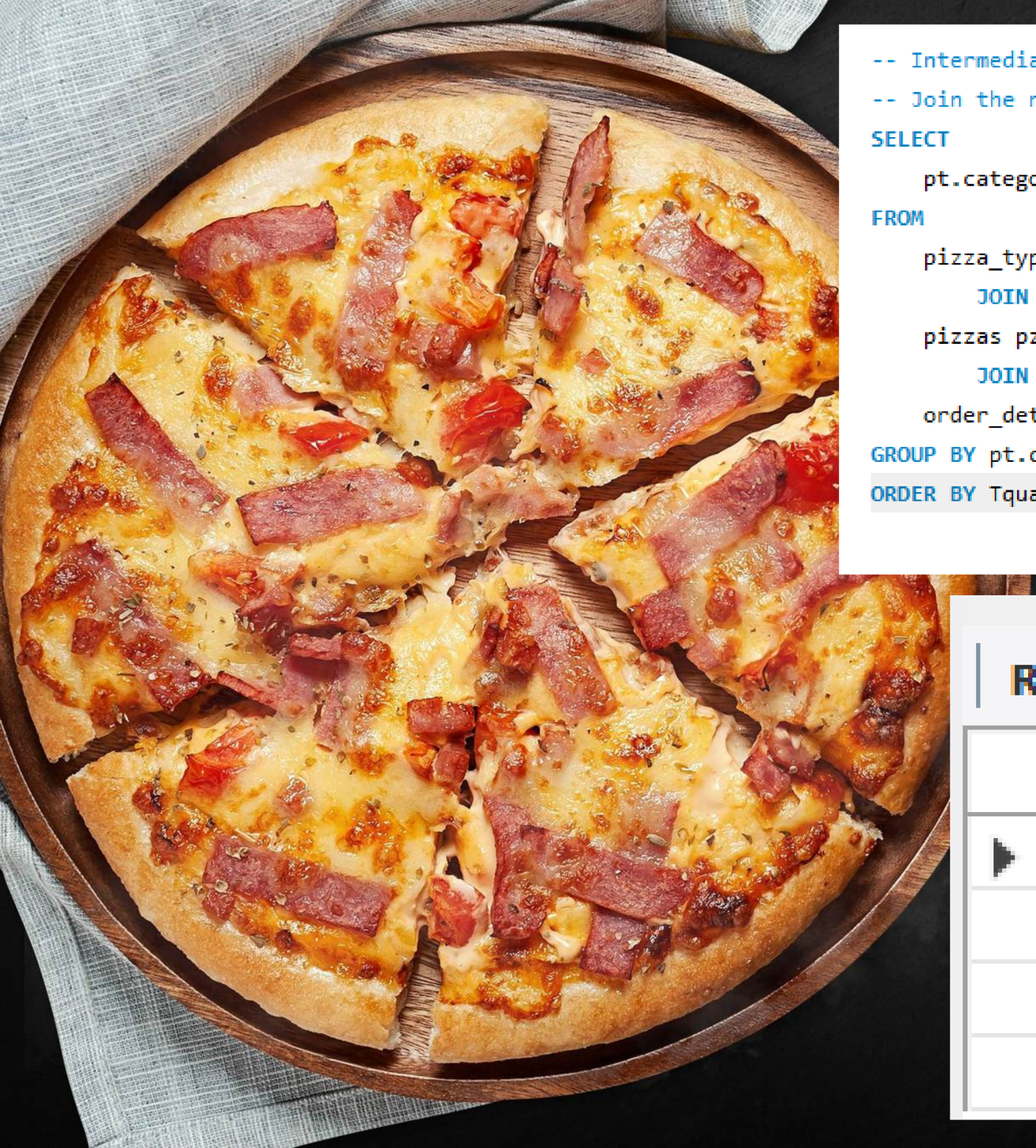
p_size	total_quantity
L	18956
M	15635
S	14403
XL	552
XXL	28

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

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

Result Grid | Filter Rows:

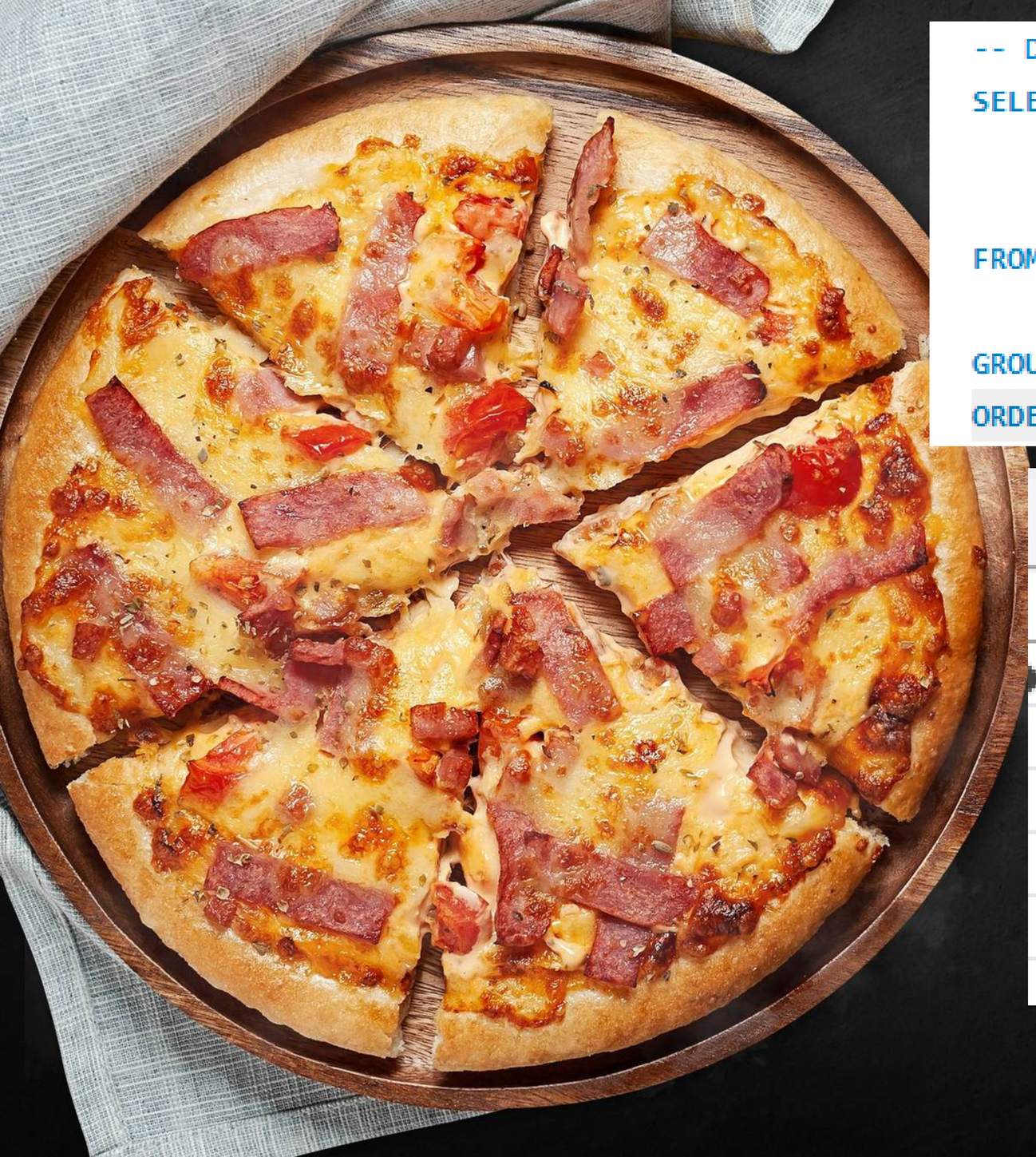
name	quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371



```sql
-- Intermediate:
-- Join the necessary tables to find the total quantity of each pizza category ordered.
SELECT
    pt.category, SUM(od.quantity) AS Tquantity
FROM
    pizza_types pt
        JOIN
    pizzas pz ON pt.pizza_type_id = pz.pizza_type_id
        JOIN
    order_details od ON od.pizza_id = pz.pizza_id
GROUP BY pt.category
ORDER BY Tquantity DESC;
```

Result Grid | Filter Rov

category	Tquantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050



```sql
-- Determine the distribution of orders by hour of the day.
SELECT
    HOUR(o.order_time) AS hourly_time,
    COUNT(order_id) AS t_orders
FROM
    orders o
GROUP BY HOUR(order_time)
ORDER BY HOUR(o.order_time) ASC
```

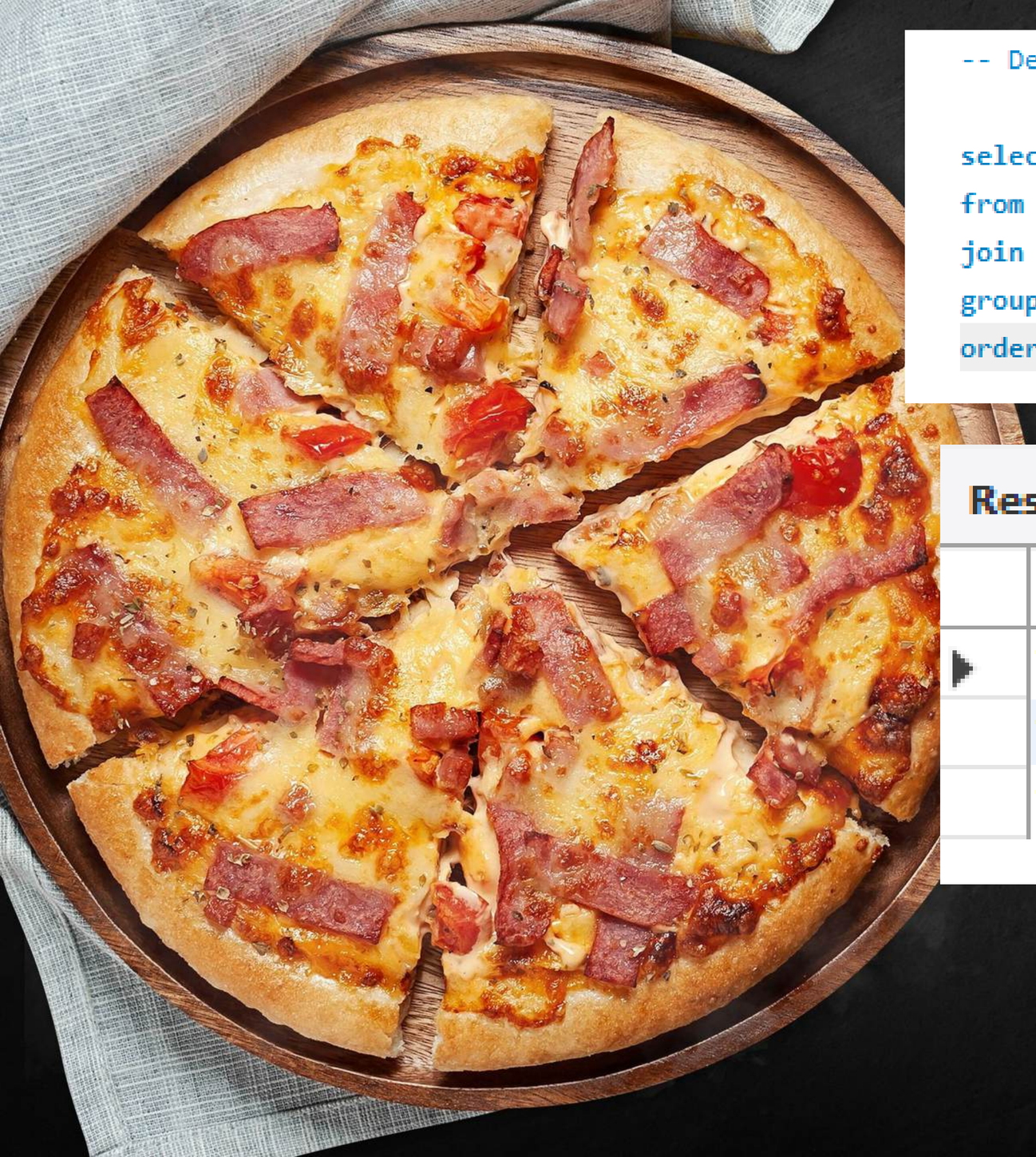
Result Grid | Filter Rows:

hourly_time	t_orders
9	1
10	8
11	1231
12	2520
13	2455
14	1472

```sql
-- Join relevant tables to find the category-wise distribution of pizzas.
SELECT
    category, COUNT(name) AS t_distribution
FROM
    pizza_types
GROUP BY category
```

Result Grid | Filter Rows:

category	t_distribution
Chicken	6
Classic	8
Supreme	9
Veggie	9

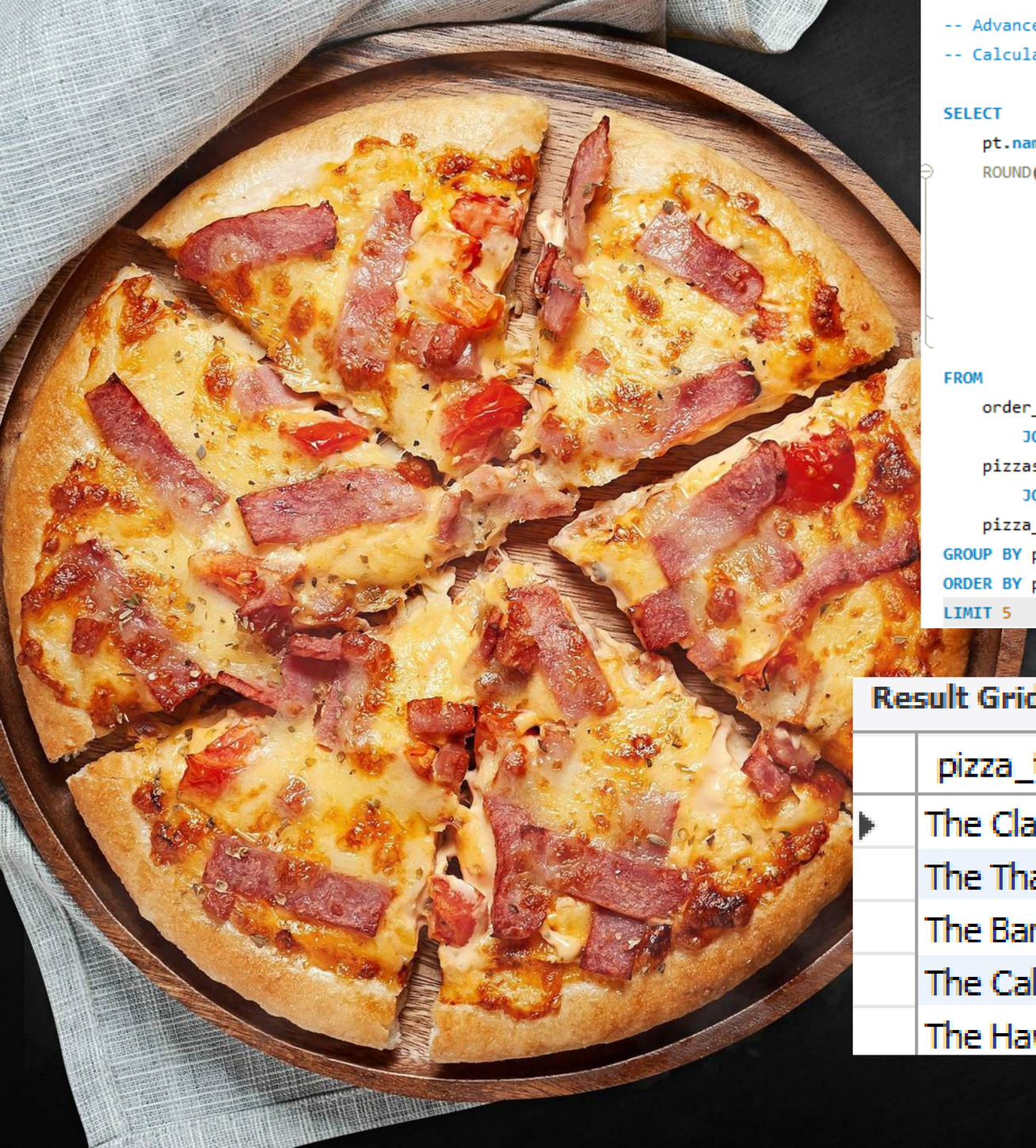


```sql
-- Determine the top 3 most ordered pizza types based on revenue.

select pt.name, sum(od.quantity* p.price) as top_3_rev
from order_details od join pizzas p on od.pizza_id = p.pizza_id
join pizza_types pt on pt.pizza_type_id = p.pizza_type_id
group by pt.name
order by top_3_rev desc limit 3
```

Result Grid | Filter Rows:

name	top_3_rev
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

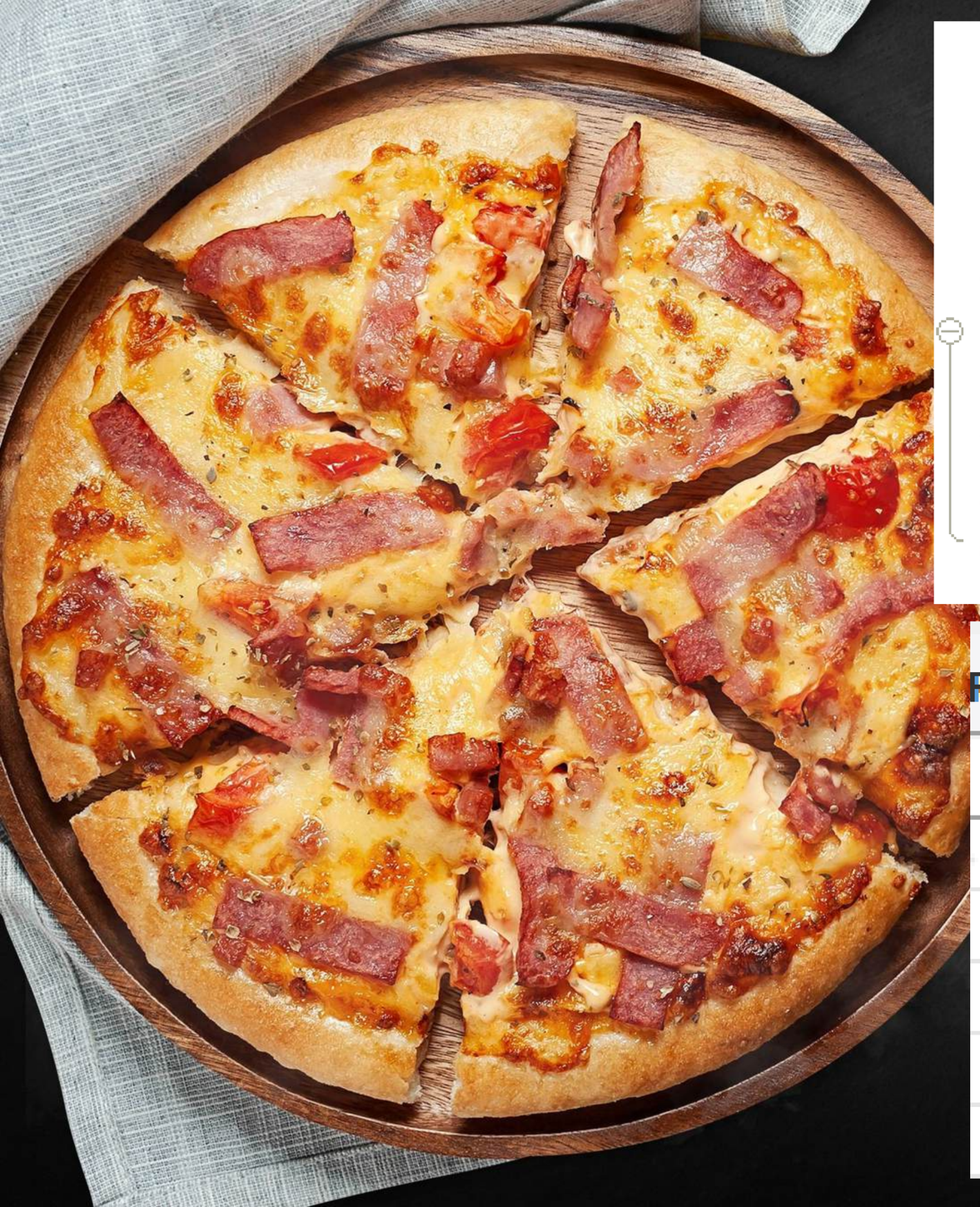


```sql
-- Advanced:
-- Calculate the percentage contribution of each pizza type to total revenue.

SELECT
    pt.name pizza_types,
    ROUND(SUM(od.quantity * p.price) / (SELECT
                    SUM(od2.quantity * p2.price)
                FROM
                    order_details od2
                        JOIN
                    pizzas p2 ON od2.pizza_id = p2.pizza_id) * 100,
            0) AS percentage_contribution
FROM
    order_details od
        JOIN
    pizzas p ON od.pizza_id = p.pizza_id
        JOIN
    pizza_types pt ON pt.pizza_type_id = p.pizza_type_id
GROUP BY pt.name
ORDER BY percentage_contribution DESC
LIMIT 5
```

Result Grid | Filter Rows: | Export:

pizza_types	percentage_contribution
The Classic Deluxe Pizza	5
The Thai Chicken Pizza	5
The Barbecue Chicken Pizza	5
The California Chicken Pizza	5
The Hawaiian Pizza	4



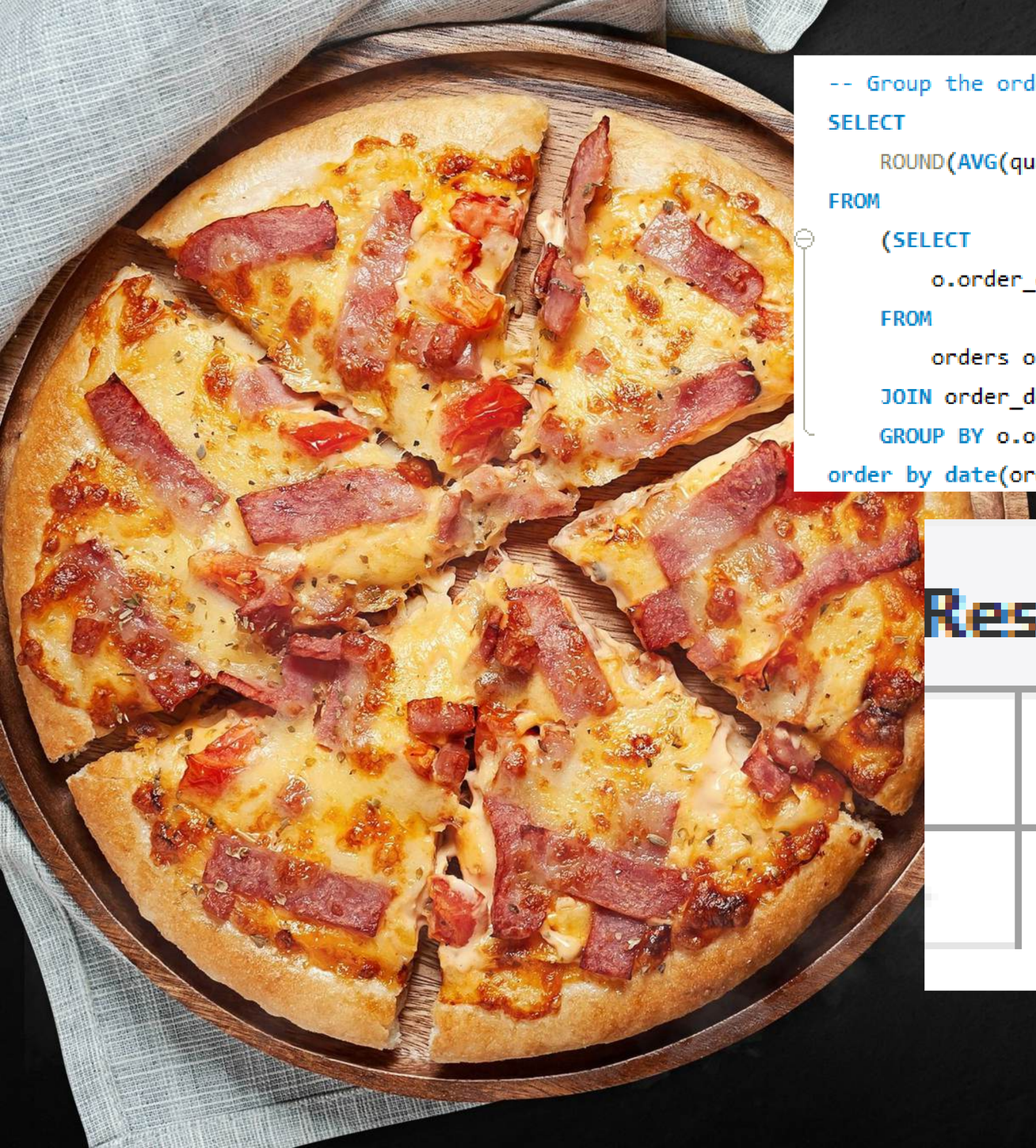
```sql
-- Analyze the cumulative revenue generated over time.

select sales.order_date,
sum(daily_revenue) over (order by sales.order_date) as cum_sum
from
(select o.order_date,
sum(od.quantity*p.price) as daily_revenue
from order_details od join pizzas p on od.pizza_id = p.pizza_id
join orders o on od.order_id = o.order_id
group by o.order_date) as sales;
```

Result Grid | Filter Rows:

order_date	cum_sum
2015-01-01	2713.850000000004
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55

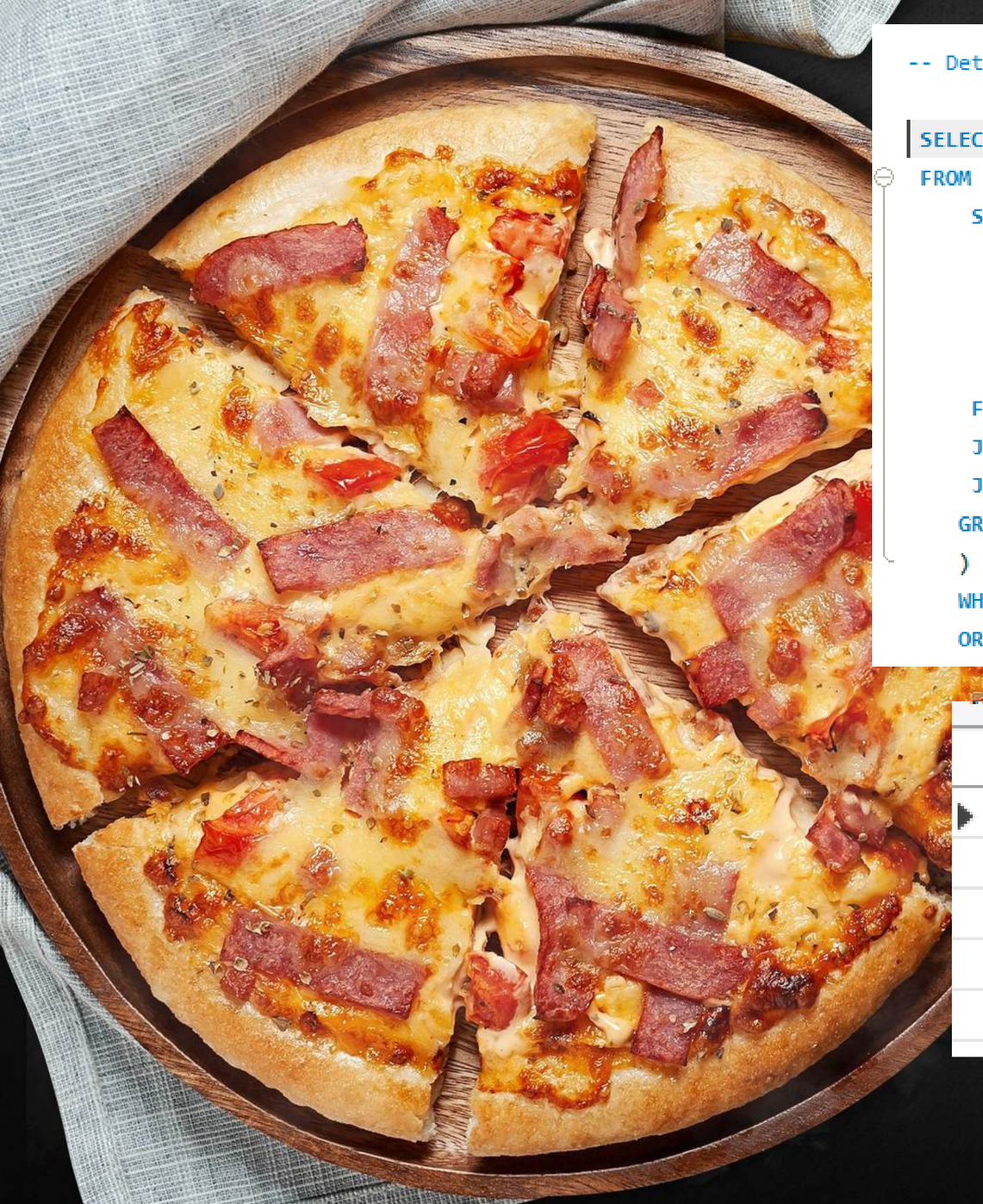
```sql
-- Analyze the cumulative revenue generated over time (day-wise)
SELECT
    sales.order_date,
    SUM(sales.daily_revenue) OVER (ORDER BY sales.order_date) AS cumulative_revenue
FROM (
    SELECT
        o.order_date,
        SUM(od.quantity * p.price) AS daily_revenue
    FROM
        order_details od
        JOIN pizzas p ON od.pizza_id = p.pizza_id
        JOIN orders o ON od.order_id = o.order_id
    GROUP BY
        o.order_date
) AS sales;
```



```sql
-- Group the orders by date and calculate the average number of pizzas ordered per day.
SELECT
    ROUND(AVG(quantity), 0) AS avg_sale_per_day
FROM
    (SELECT
        o.order_date, SUM(od.quantity) AS quantity
    FROM
        orders o
    JOIN order_details od ON o.order_id = od.order_id
    GROUP BY o.order_date) AS date_wise_pizza_ordered
order by date(order_date) desc
```

Result Grid | Filte

avg_sale_per_day
138



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

SELECT name, category, revenue
FROM (
    SELECT
        pt.name,
        pt.category,
        SUM(od.quantity * p.price) AS revenue,
        RANK() OVER (PARTITION BY pt.category ORDER BY SUM(od.quantity * p.price) DESC) AS rnk
    FROM pizza_types pt
    JOIN pizzas p ON pt.pizza_type_id = p.pizza_type_id
    JOIN order_details od ON od.pizza_id = p.pizza_id
    GROUP BY pt.name, pt.category
    ) AS ranked_pizzas
    WHERE rnk <= 3
    ORDER BY category, revenue DESC;
```

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

THANK YOU!

Hope you enjoyed this slice of data 🍕📈