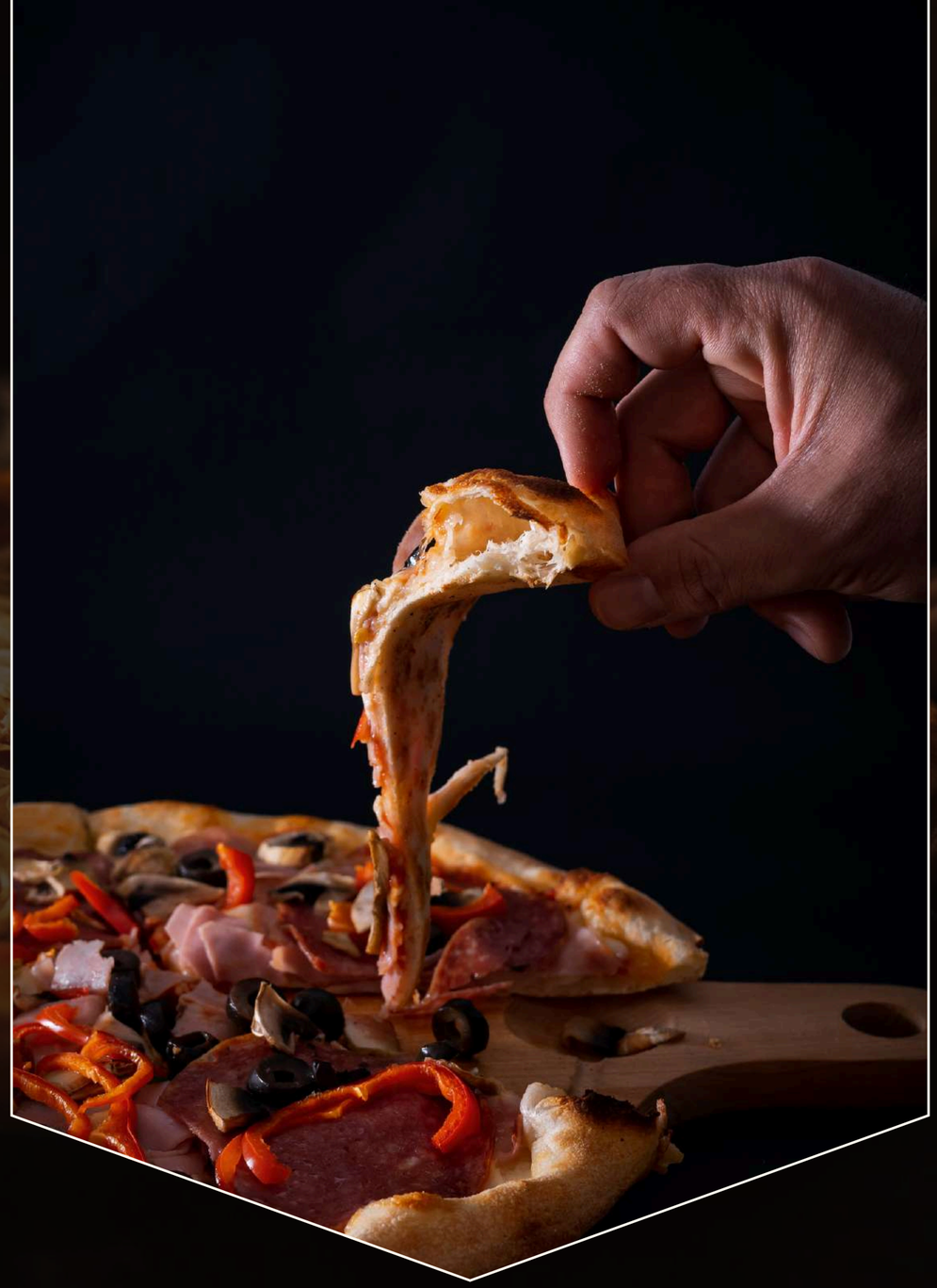
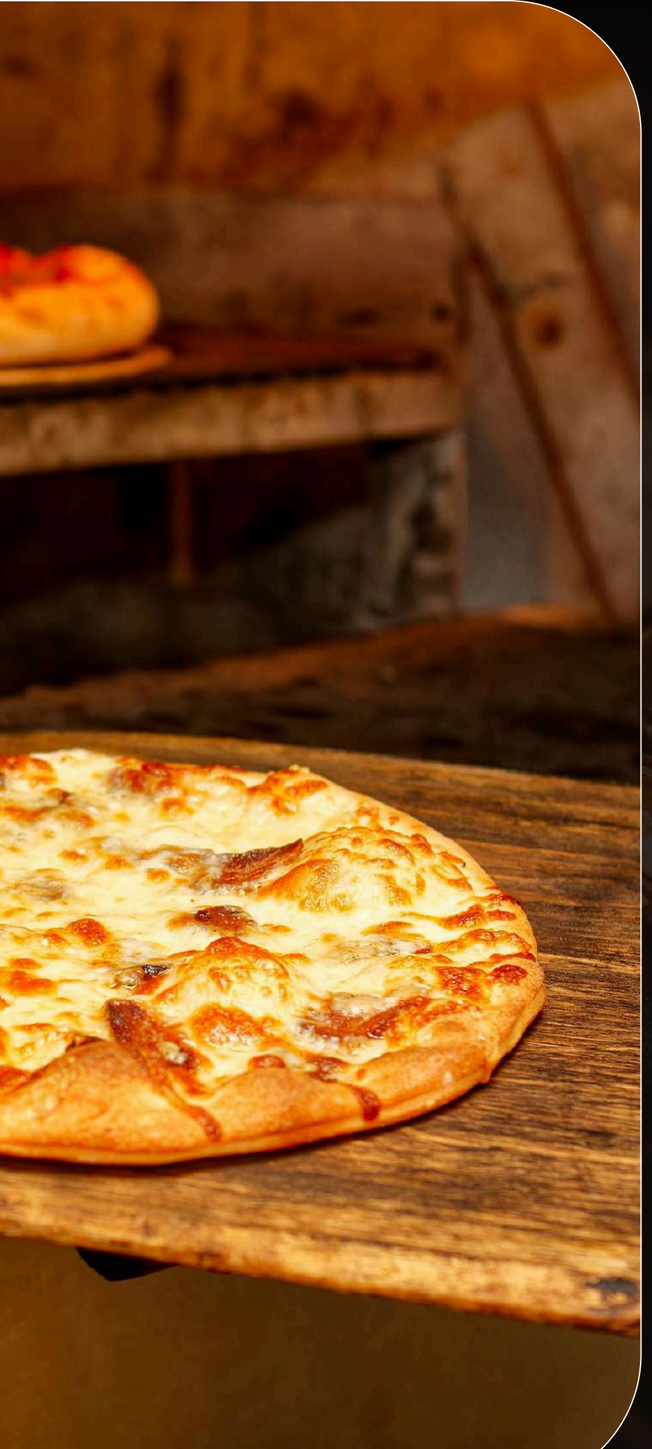
PIZZA SALES DATA ANALYSIS USING MYSQL





PROJECT OVERVIEW :

THIS PROJECT FOCUSES ON ANALYZING PIZZA SALES DATA USING MYSQL TO EXTRACT MEANINGFUL BUSINESS INSIGHTS.BY WRITING SQL QUERIES AT DIFFERENT DIFFICULTY LEVELS BASIC, INTERMEDIATE AND ADVANCED. THE PROJECT DEMONSTRATES DATA RETRIEVAL, AGGREGATION AND ANALYTICAL OPERATIONS IN A REAL-WORLD SALES SCENARIO.

OBJECTIVES :

- TO UNDERSTAND AND APPLY SQL CONCEPTS SUCH AS JOIN, GROUP BY, ORDER BY AND AGGREGATE FUNCTIONS.
- TO ANALYZE SALES PERFORMANCE, CUSTOMER PREFERENCES AND REVENUE PATTERNS FROM THE PIZZA DATASET.
- TO TRANSFORM RAW SALES DATA INTO VALUABLE INSIGHTS THAT CAN SUPPORT BUSINESS DECISION-MAKING.

KEY HIGHLIGHTS :

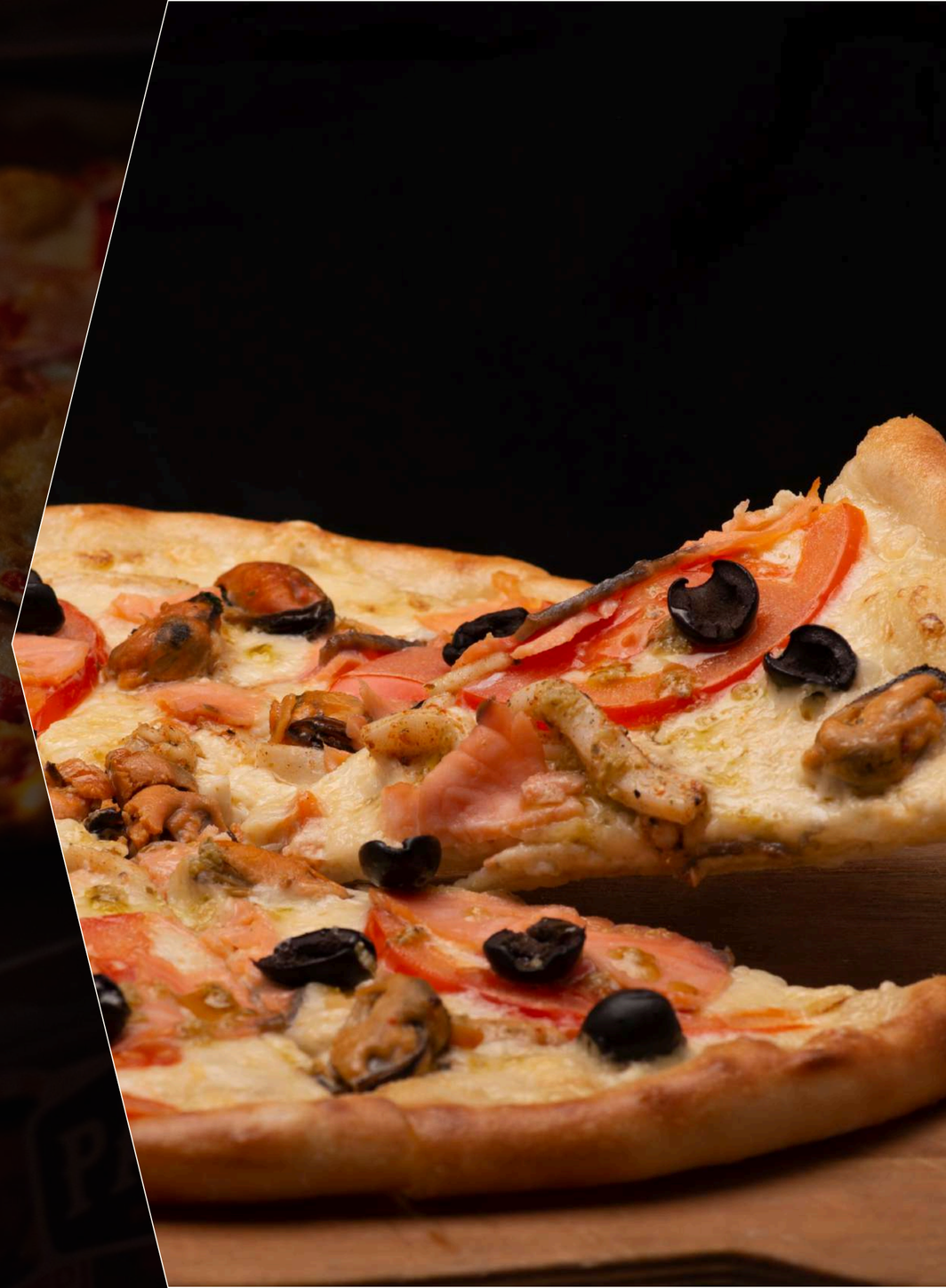
- BASIC ANALYSIS : RETRIEVED TOTAL ORDERS, TOTAL REVENUE, HIGHEST-PRICED PIZZA, MOST COMMON SIZE AND TOP 5 ORDERED PIZZA TYPES.
- INTERMEDIATE ANALYSIS : EXPLORED PIZZA CATEGORY-WISE ORDERS, HOURLY ORDER DISTRIBUTION AND DAILY AVERAGES OF PIZZAS SOLD.
- ADVANCED ANALYSIS : CALCULATED EACH PIZZA TYPE'S REVENUE CONTRIBUTION, TRACKED CUMULATIVE REVENUE GROWTH AND IDENTIFIED TOP-PERFORMING PIZZAS BY CATEGORY.

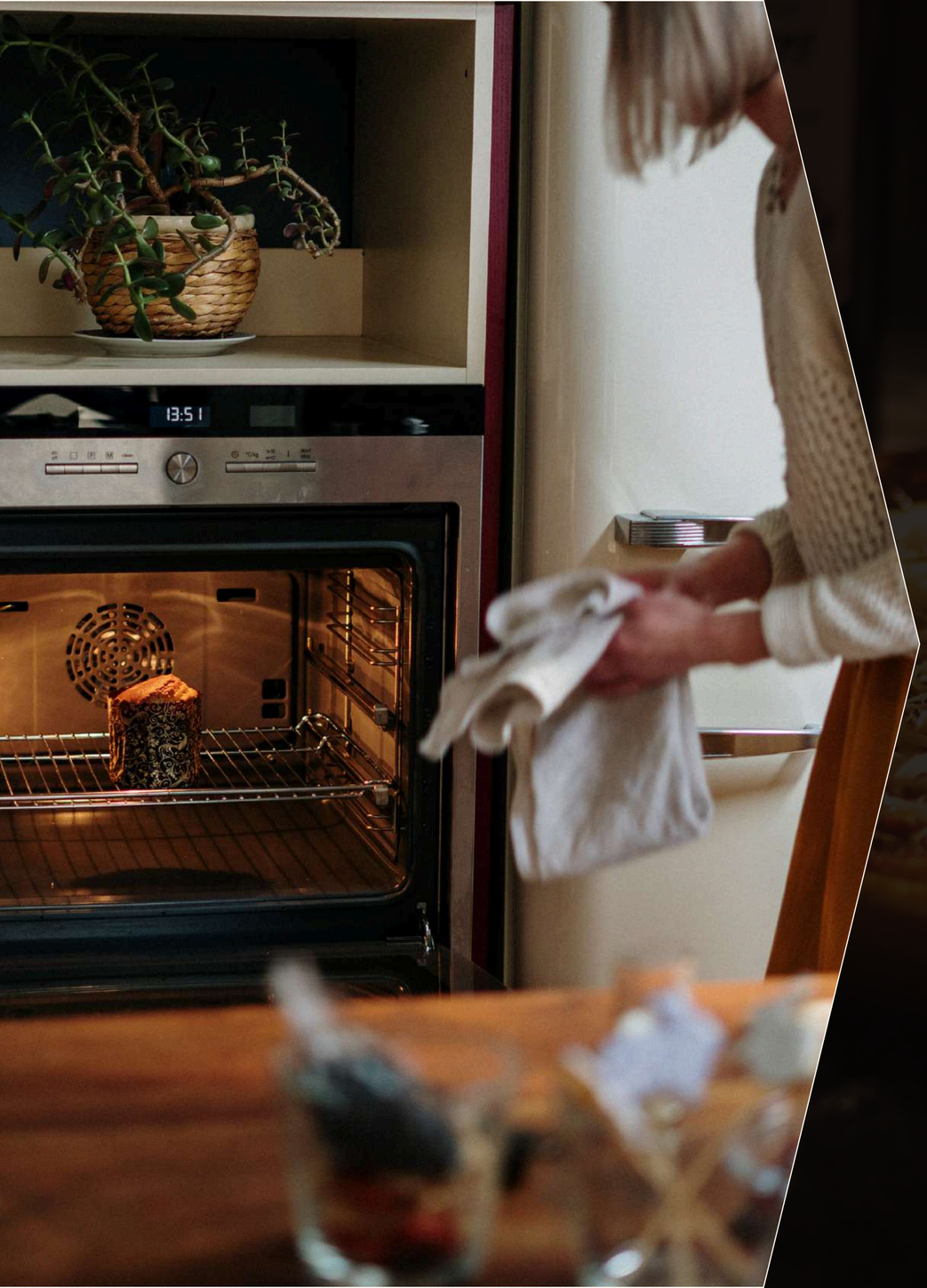
CREATING A DATABSE

```
create database Pizza_Hut;
```

IMPORTING THE FOLLOWING ".CSV" FILES AS TABLES

- order_details
- orders
- pizza_types
- pizzas





1. RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

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

OUTPUT:

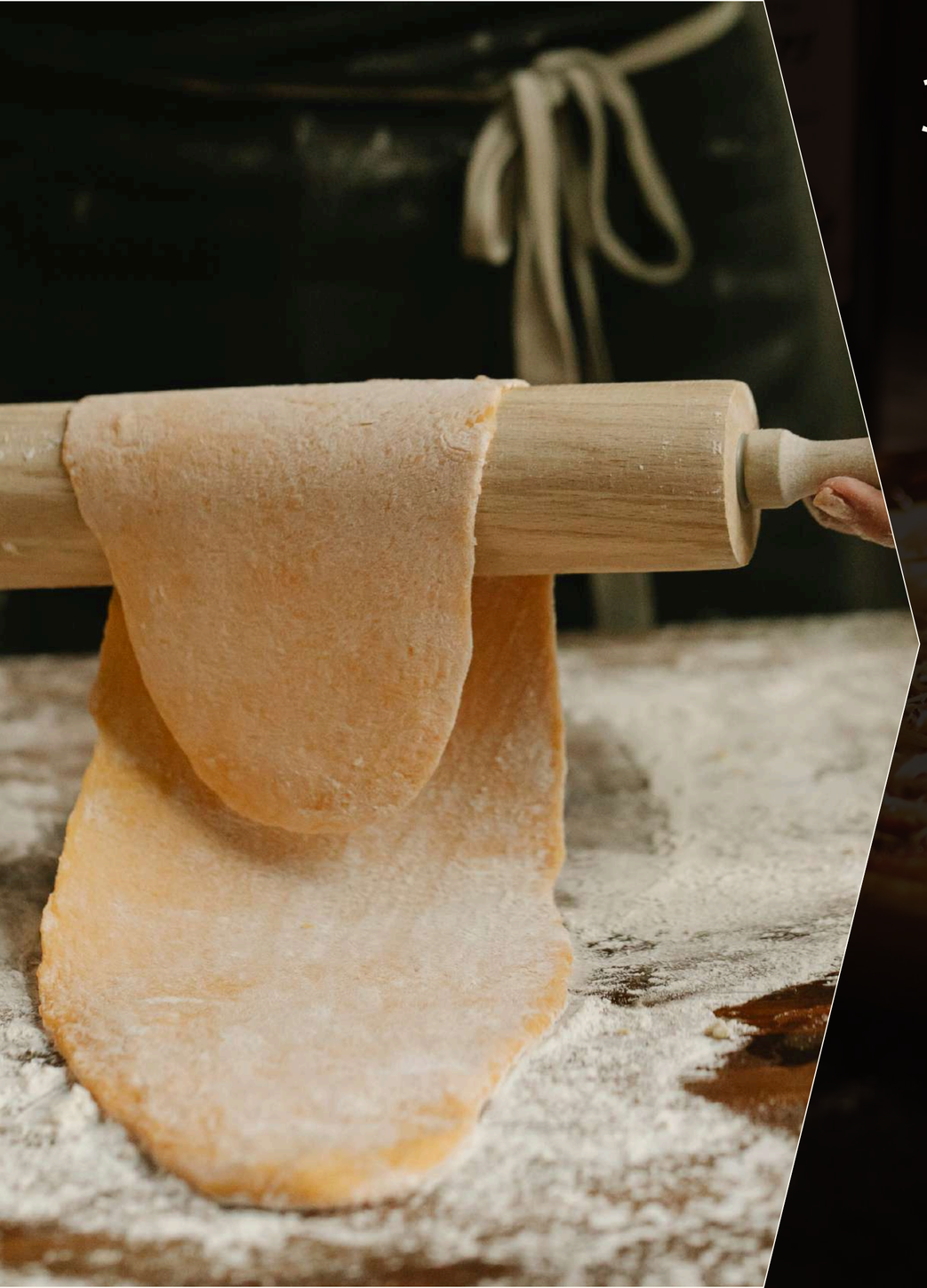
	COUNT(order_id)
▶	21350

2. 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;
```

OUTPUT:

	total_sales
▶	817860.05

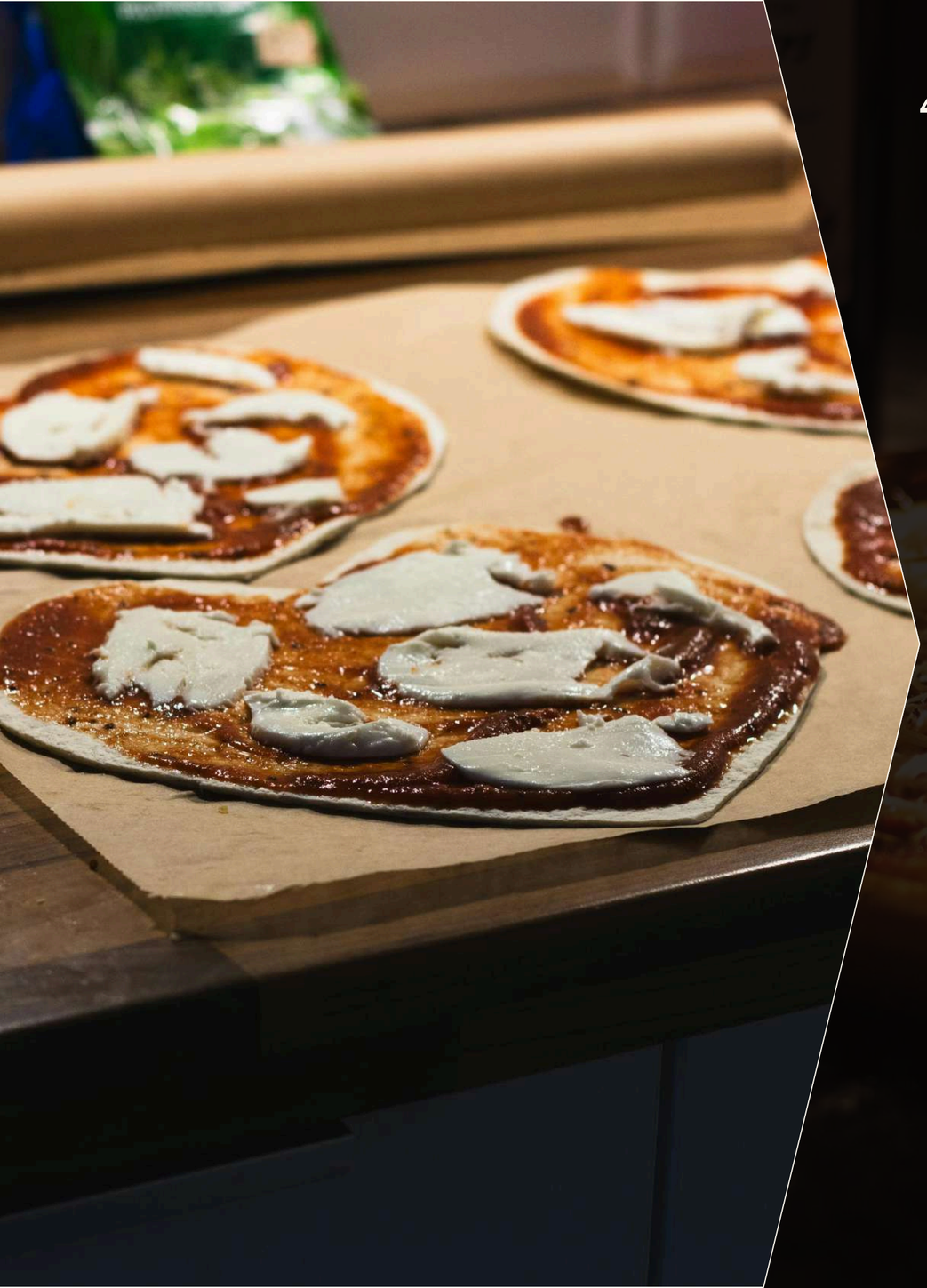


3. IDENTIFY THE HIGHEST-PRICED PIZZA.

```
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;
```

OUTPUT:

	name	price
▶	The Greek Pizza	35.95

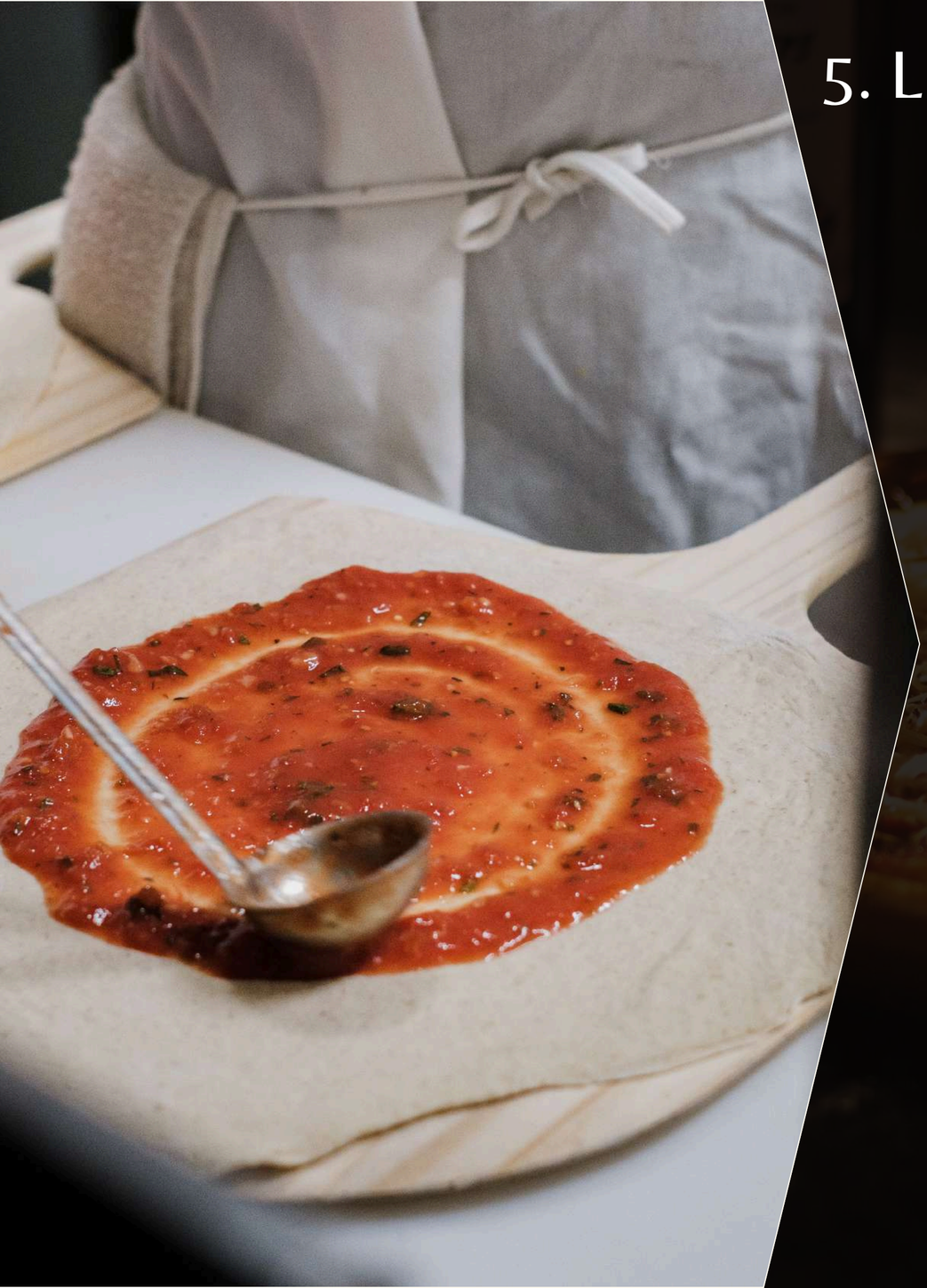


4. IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

```
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;
```

OUTPUT:

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

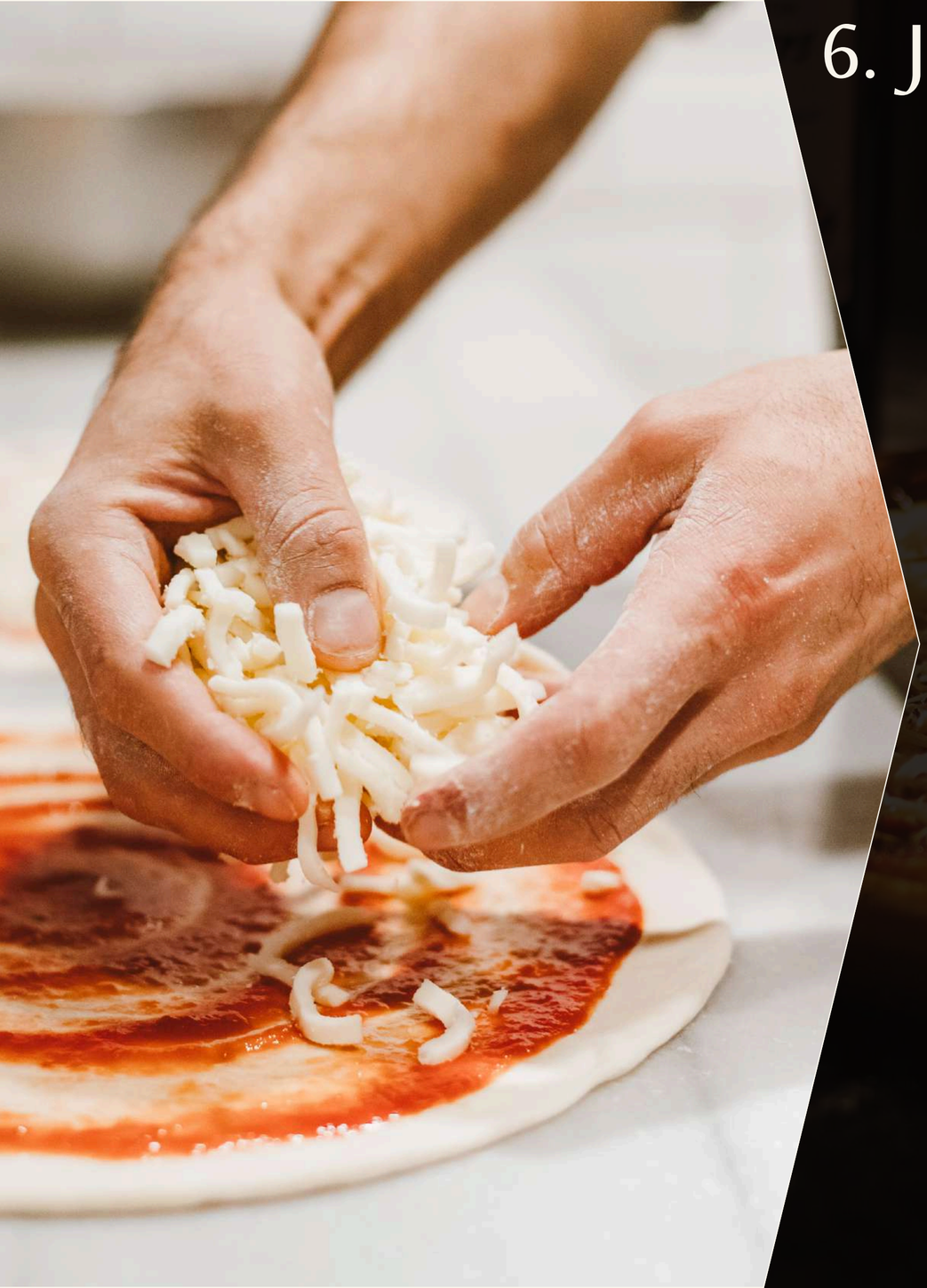


5. LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.

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

OUTPUT:

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



6. 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;
```

OUTPUT:

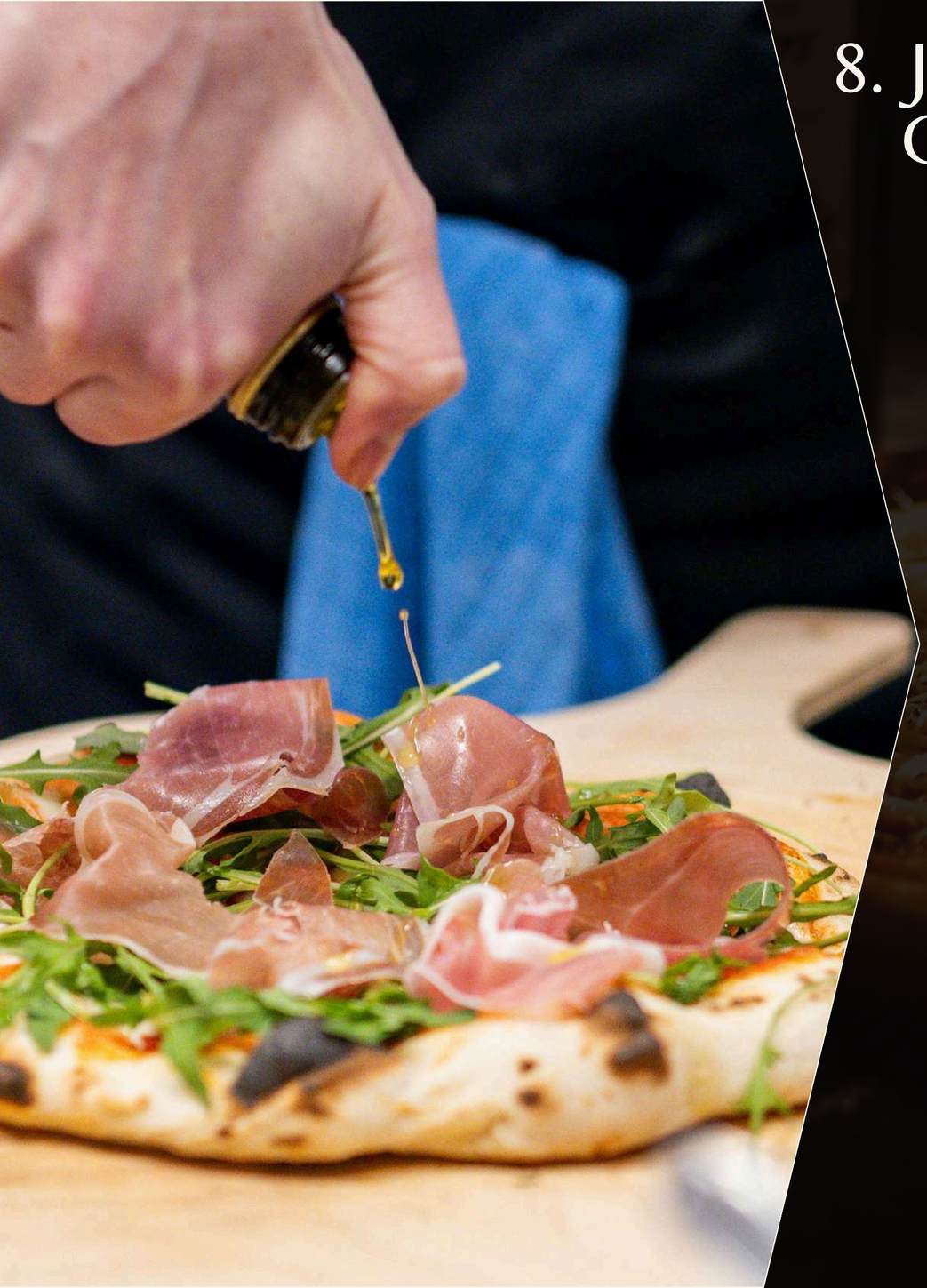
	category	Quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

7. DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

```
SELECT
    HOUR(order_time) AS Hours, COUNT(order_id) AS Order_Count
FROM
    orders
GROUP BY Hours;
```

OUTPUT:

	Hours	Order_Count
▶	11	1231
	12	2520
	13	2455
	14	1472
	15	1468
	16	1920
	17	2336

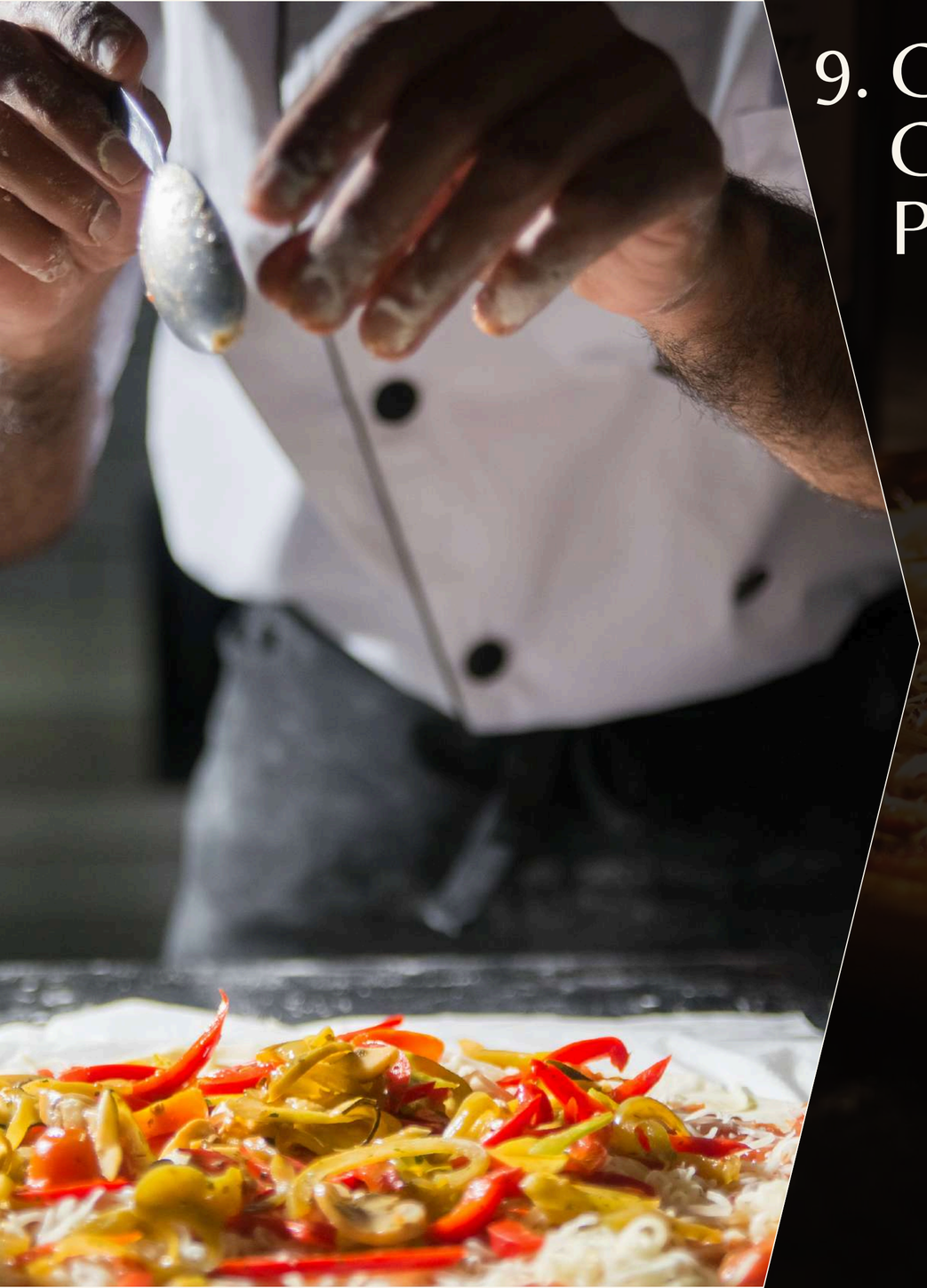


8. JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

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

OUTPUT:

	category	COUNT(name)
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

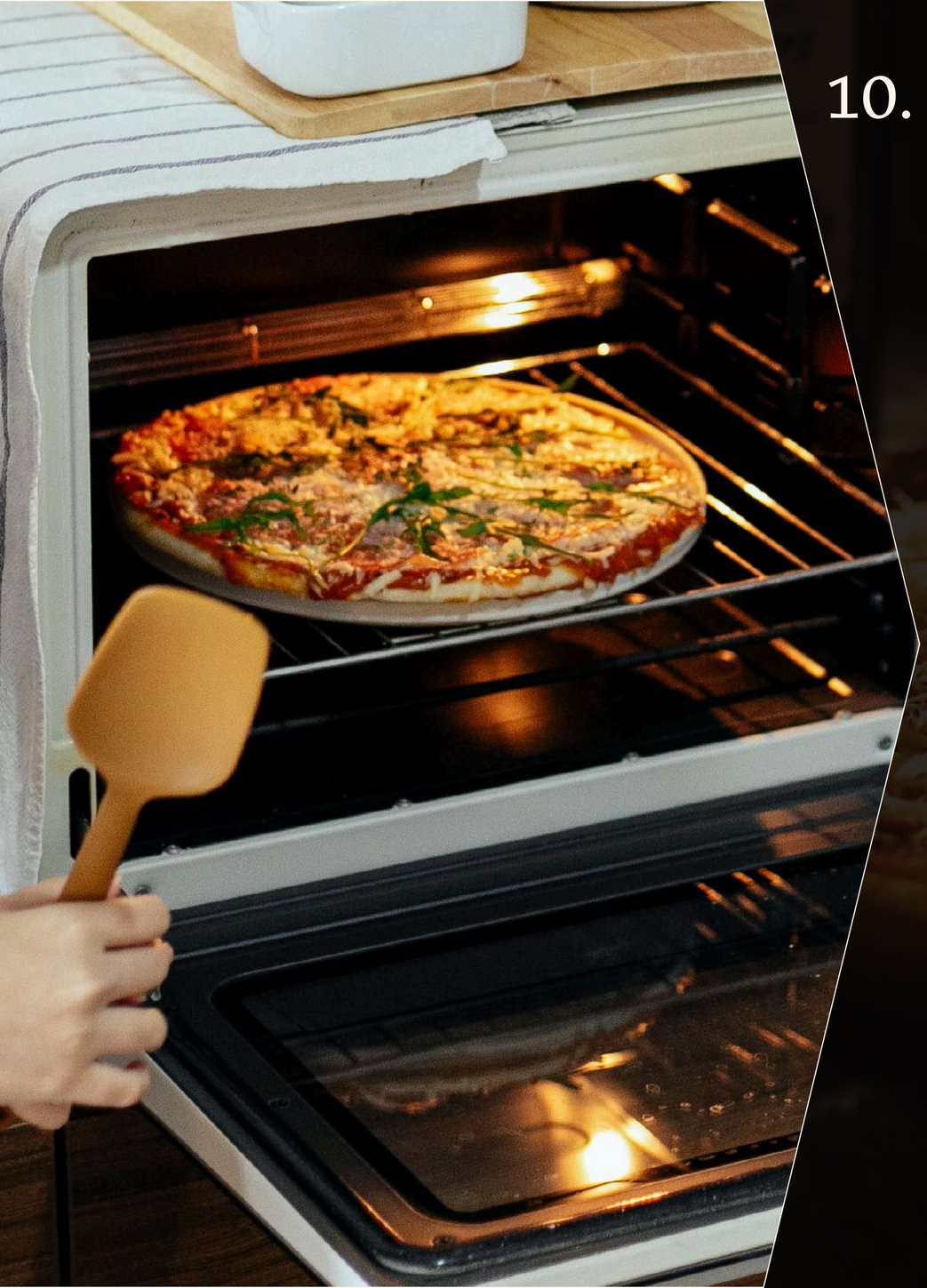


9. GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

```
SELECT
    ROUND(AVG(quantity), 0) as Avg_Pizzas_Ordered_Per_Day
FROM
    (SELECT
        orders.order_date, SUM(order_details.quantity) AS quantity
    FROM
        orders
    JOIN order_details ON order_details.order_id = orders.order_id
    GROUP BY orders.order_date) AS Orders_Quantity;
```

OUTPUT:

	Avg_Pizzas_Ordered_Per_Day
▶	138

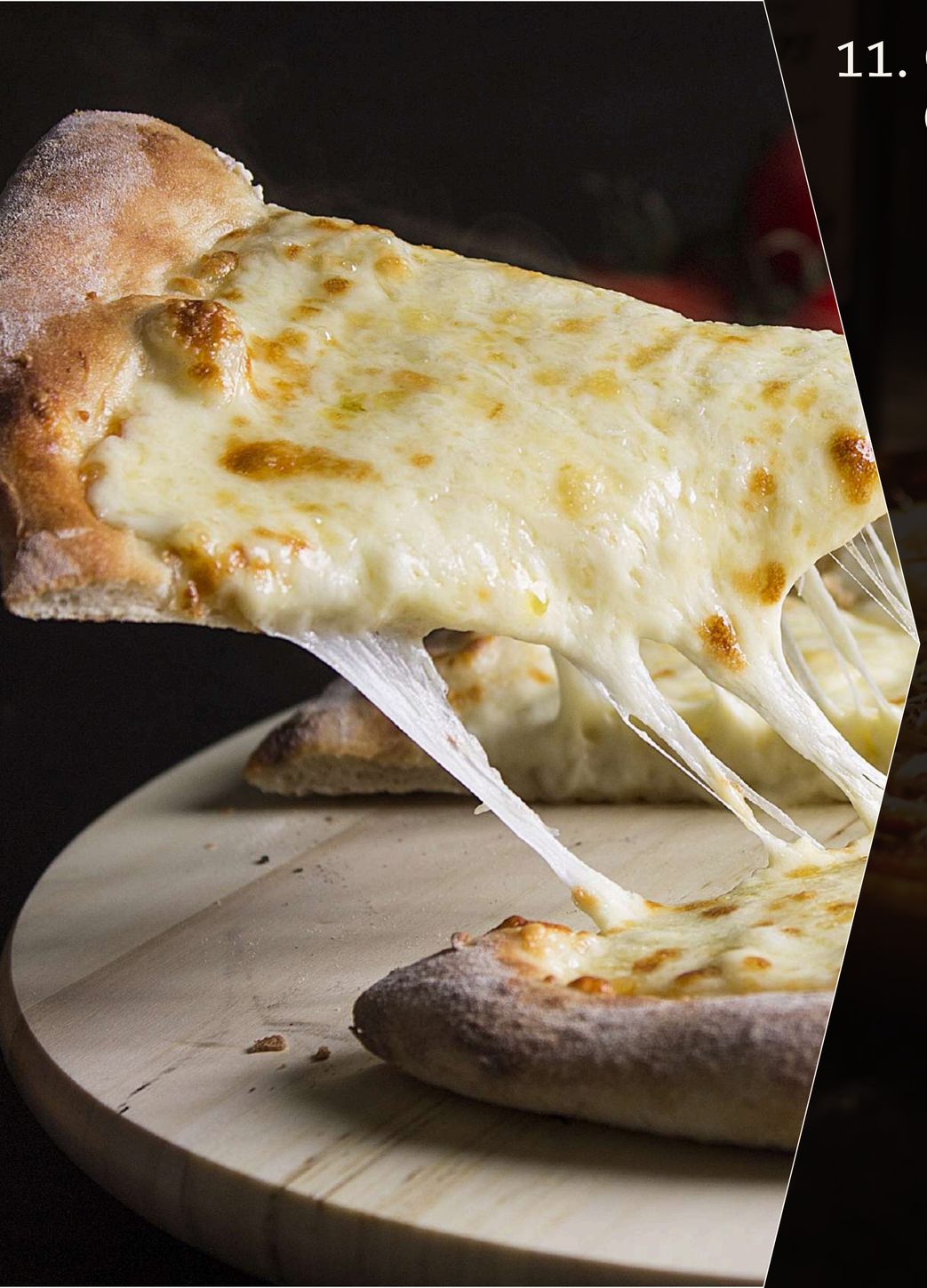


10. DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```
select (pizza_types.name) as Pizzas_Names,
round(sum(order_details.quantity * pizzas.price),0) 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 Pizzas_Names order by Revenue desc limit 3;
```

OUTPUT:

	Pizzas_Names	Revenue
▶	The Thai Chicken Pizza	43434
	The Barbecue Chicken Pizza	42768
	The California Chicken Pizza	41410

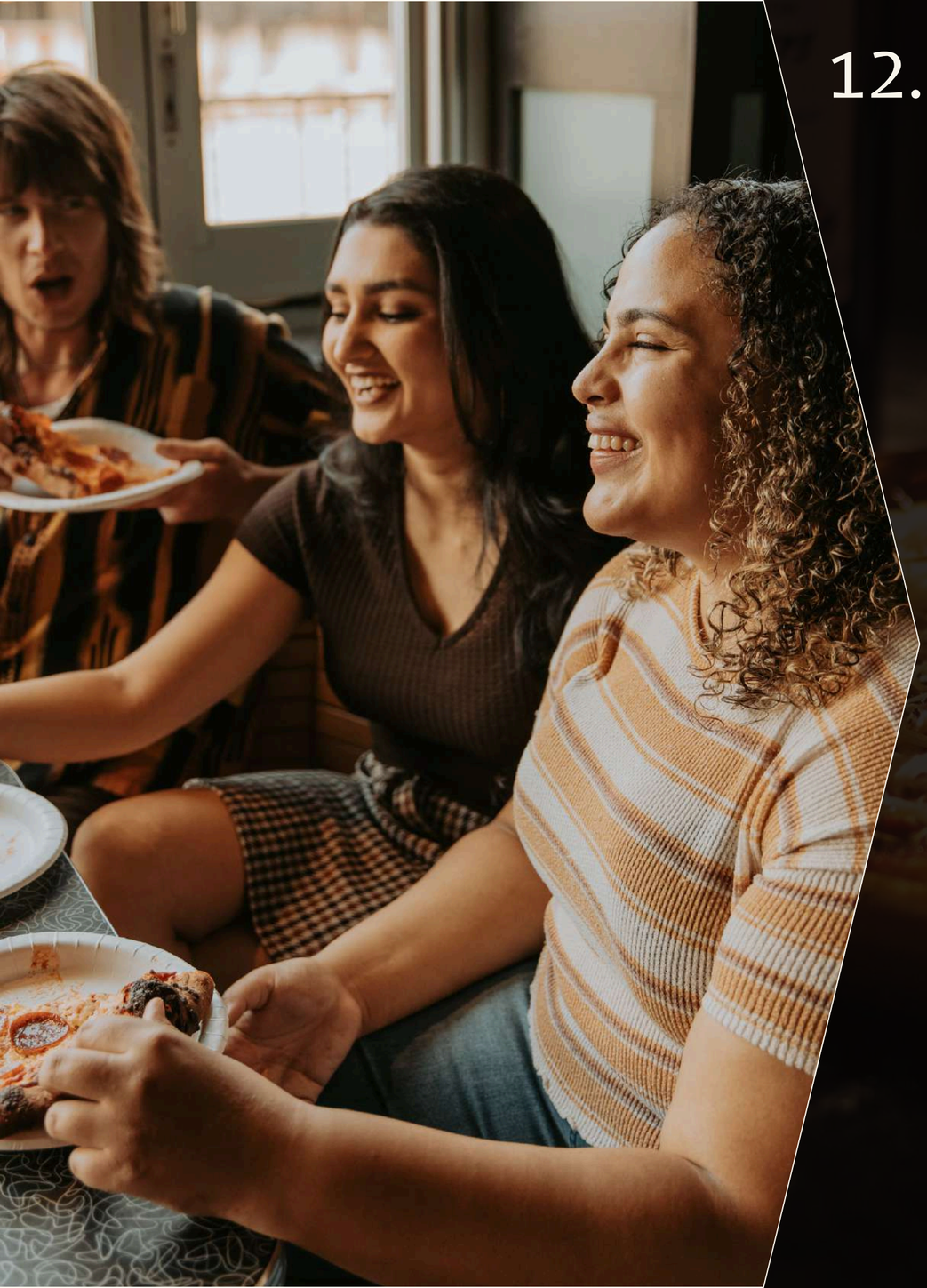


11. CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

```
SELECT
    pizza_types.category,
    ROUND(SUM(order_details.quantity * pizzas.price) / (SELECT
                    ROUND(SUM(order_details.quantity * pizzas.price),
                                2) AS Total_Sales
                FROM
                    order_details
                        JOIN
                    pizzas ON order_details.pizza_id = pizzas.pizza_id) * 100,
            2) AS Percentage
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizza_types.category;
```

OUTPUT:

	category	Percentage
▶	Classic	26.91
	Veggie	23.68
	Supreme	25.46
	Chicken	23.96



12. ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```
select order_date, revenue,
sum(Revenue) over(order by order_date) as cum_revenue
from
  (select orders.order_date,
round(sum(order_details.quantity * pizzas.price),2) 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;
```

OUTPUT:

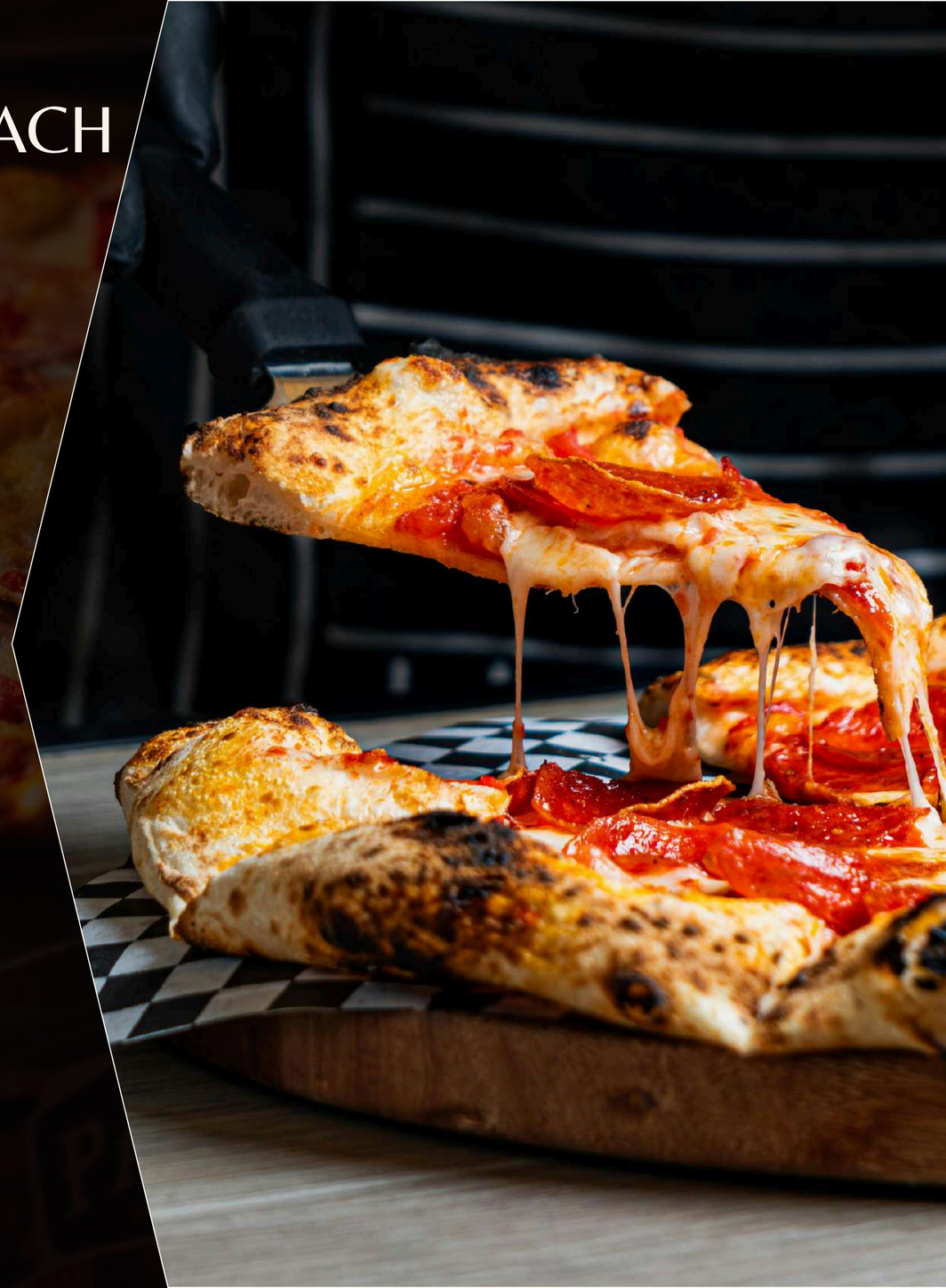
	Order_Date	Revenue	Cum_Revenue
▶	2015-01-01	2713.85	2713.85
	2015-01-02	2731.9	5445.75
	2015-01-03	2662.4	8108.15
	2015-01-04	1755.45	9863.6
	2015-01-05	2065.95	11929.55
	2015-01-06	2428.95	14358.5
	2015-01-07	2202.2	16560.7

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

```sql
select category, name, Revenue,
rank() over(partition by category order by revenue desc) as rn
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 a;
```

OUTPUT:

	category	name	Revenue	rn
▶	Chicken	The Thai Chicken Pizza	43434.25	1
	Chicken	The Barbecue Chicken Pizza	42768	2
	Chicken	The California Chicken Pizza	41409.5	3
	Chicken	The Southwest Chicken Pizza	34705.75	4
	Chicken	The Chicken Alfredo Pizza	16900.25	5
	Chicken	The Chicken Pesto Pizza	16701.75	6
	Classic	The Classic Deluxe Pizza	38180.5	1



Thank you