Where Every Slice is a Taste of Perfection

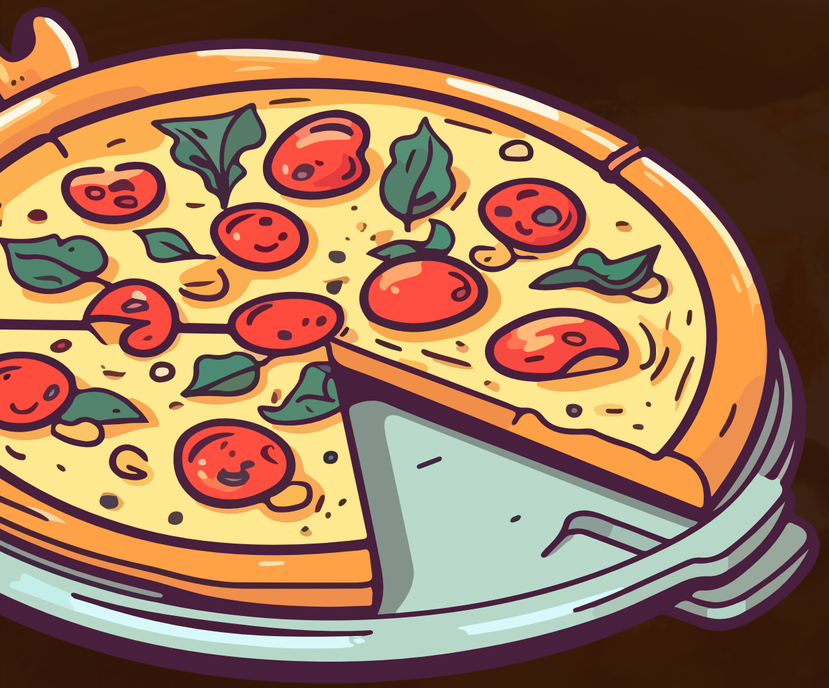
PIZZA SALES ANALYSIS USING MYSQL

ORDER NOW

Start Your Slide

INTRODUCTION

PIZZA SALES -PROJECT



Hey,

I'm Udit Rawat. I have created a Pizza Sales Analysis Project using MySQL. In this project, I used a pizza sales dataset to write SQL queries and analyze business performance. The analysis focused on understanding sales trends, identifying top-performing pizzas, peak order times, and customer buying behavior.

This project helped me strengthen my MySQL skills and gain insights into real-world data analysis.

RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

```
3 •    SELECT
4          COUNT(order_id) AS total_orders
5      FROM
6          orders;
```

Result Grid

	total_orders
▶	21350

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

```
3    SELECT
4        ROUND(SUM(order_details.quantity * pizzas.price),
5                2) AS total_sales
6    FROM
7        order_details
8            JOIN
9        pizzas ON pizzas.pizza_id = order_details.pizza_id
```

Result Grid

	total_sales
▶	817860.05

IDENTIFY THE HIGHEST PRICED PIZZA.

```sql
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;
```

Result Grid | Filter

name	price
The Greek Pizza	35.95

IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

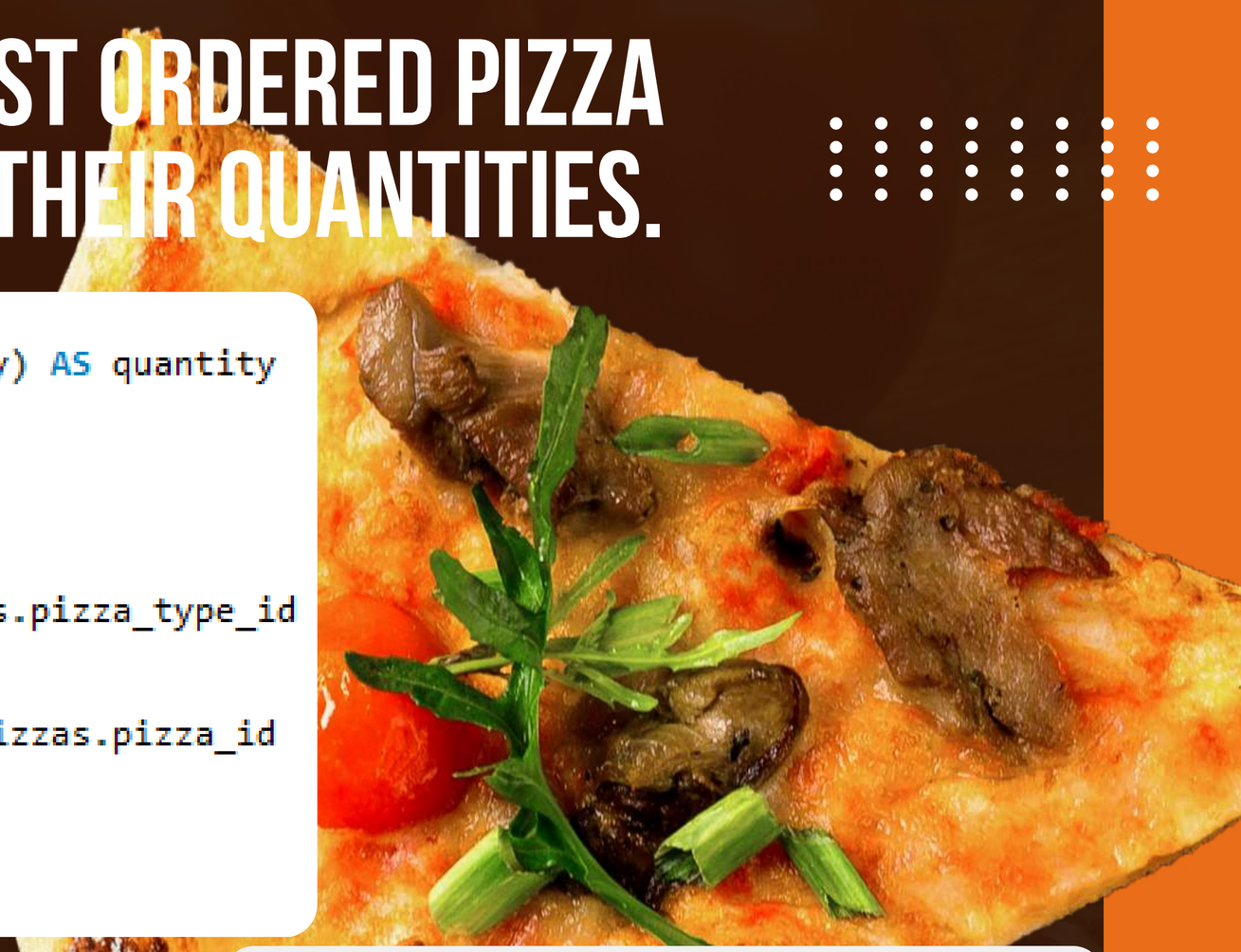
```sql
SELECT
    pizzas.size,
    COUNT(order_details.order_details_id) AS order_count
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC;
```

Result Grid | Filter

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

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

```sql
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;
```



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

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

```sql
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;
```

Result Grid | Filter

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

```sql
SELECT
    HOUR(order_time) AS hour, COUNT(order_id)
FROM
    orders
GROUP BY HOUR(order_time);
```

Result Grid | Filter R

hour	COUNT(order_id)
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399

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

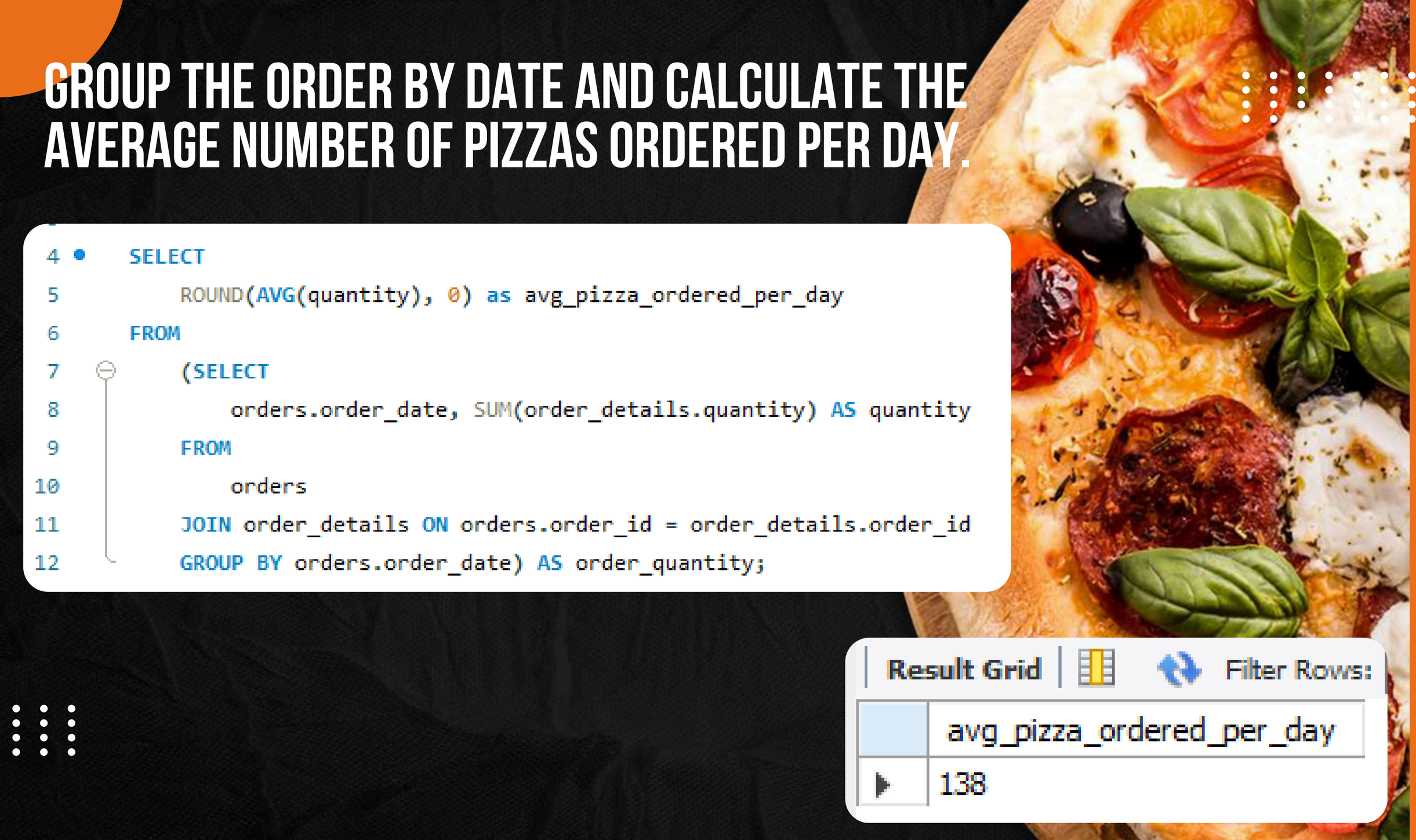
```
3 •    Select category, count(name) from pizza_types
4      group by category;
```

Result Grid | Filter Rows:

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

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

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



Result Grid | Filter Rows:

avg_pizza_ordered_per_day
138

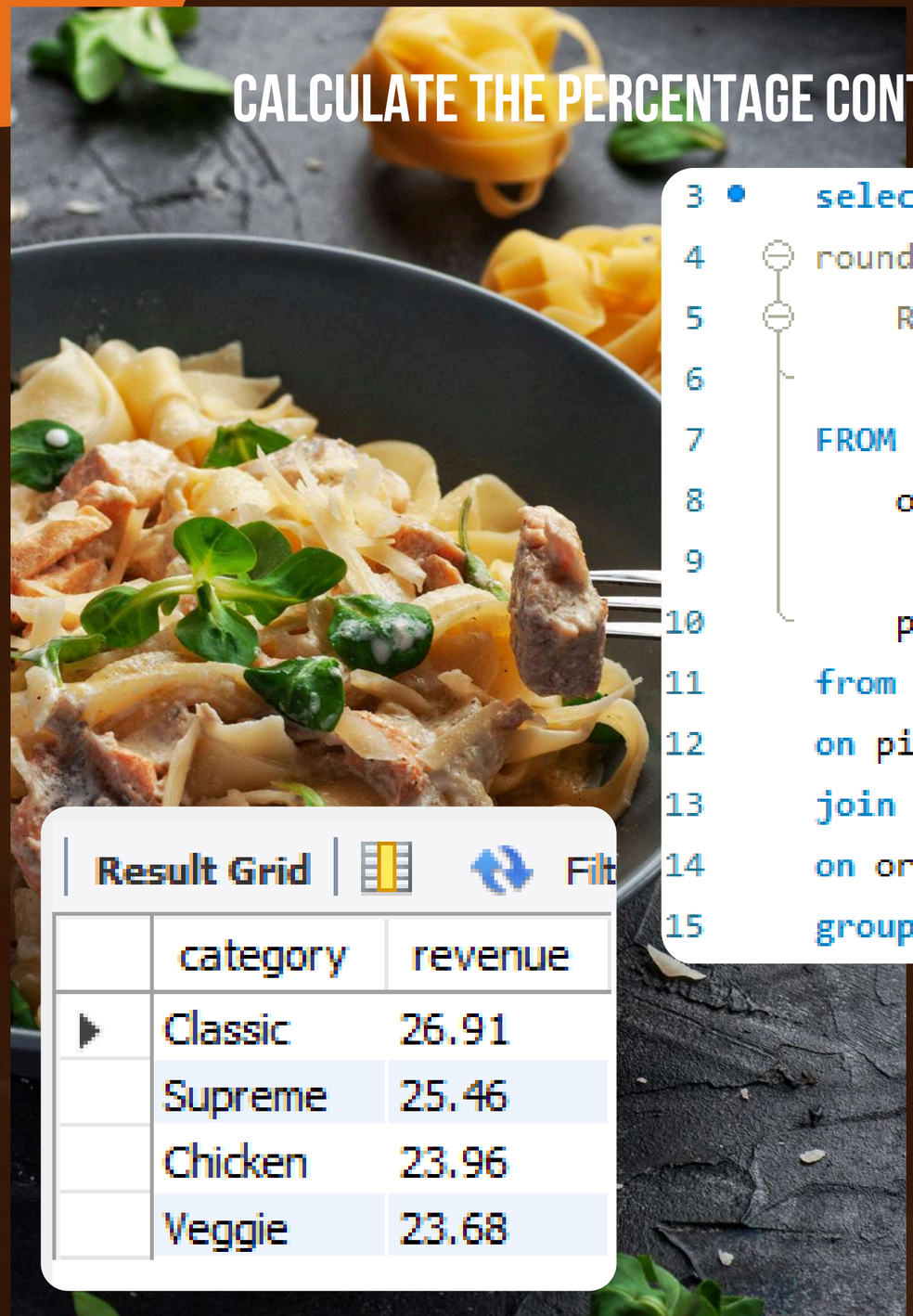
DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```sql
SELECT
    pizza_types.name,
    SUM(order_details.quantity * pizzas.price) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.pizza_type_id = pizza_types.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;
```

Result Grid | Filter Rows:

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

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



```sql
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 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 order_details
on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.category order by revenue desc;
```

Result Grid | Filt

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```
3  •  select order_date,
4     sum(revenue) over(order by order_date) as cum_revenue
5     from
6     (select orders.order_date,
7     sum(order_details.quantity * pizzas.price) as revenue
8     from order_details join pizzas
9     on order_details.pizza_id = pizzas.pizza_id
10    join orders
11    on orders.order_id = order_details.order_id
12    group by orders.order_date) as sales;
```

Result Grid | Filter Rows:

order_date	cum_revenue
2015-01-01	2713.8500000000004
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

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

Result Grid | Filter Rows:

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

```
4  select name, revenue from
5  (select category, name, revenue,
6  rank() over(partition by category order by revenue desc) as rn
7  from
8  (select pizza_types.category, pizza_types.name,
9  sum((order_details.quantity) * pizzas.price) as revenue
10 from pizza_types join pizzas
11 on pizza_types.pizza_type_id = pizzas.pizza_type_id
12 join order_details
13 on order_details.pizza_id = pizzas.pizza_id
14 group by pizza_types.category, pizza_types.name) as a) as b
15 where rn<=3;
```


Pizza sales analysis using MySQL
THANK YOU
FOR ATTENTION
udit.rawat1256@gmail.com