

Hello ! my name is Akram Qureshi. In this project i have used SQL to analyze and optimize pizza sales data.



Retrieve the total number of orders placed

SELECT

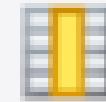
COUNT(order_id)

FROM

orders;



Result Grid



COUNT(order_id)
21350

Calculate the total revenue generated from pizza sales

SELECT

```
ROUND(SUM(order_details.quantity * pizzas.price),  
2) AS totalrevenue
```

FROM

```
order_details
```

JOIN

```
pizzas ON order_details.pizza_id = pizzas.pizza_id;
```

Result Grid



	totalrevenue
▶	817860.05



Identify the highest-priced pizza.

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

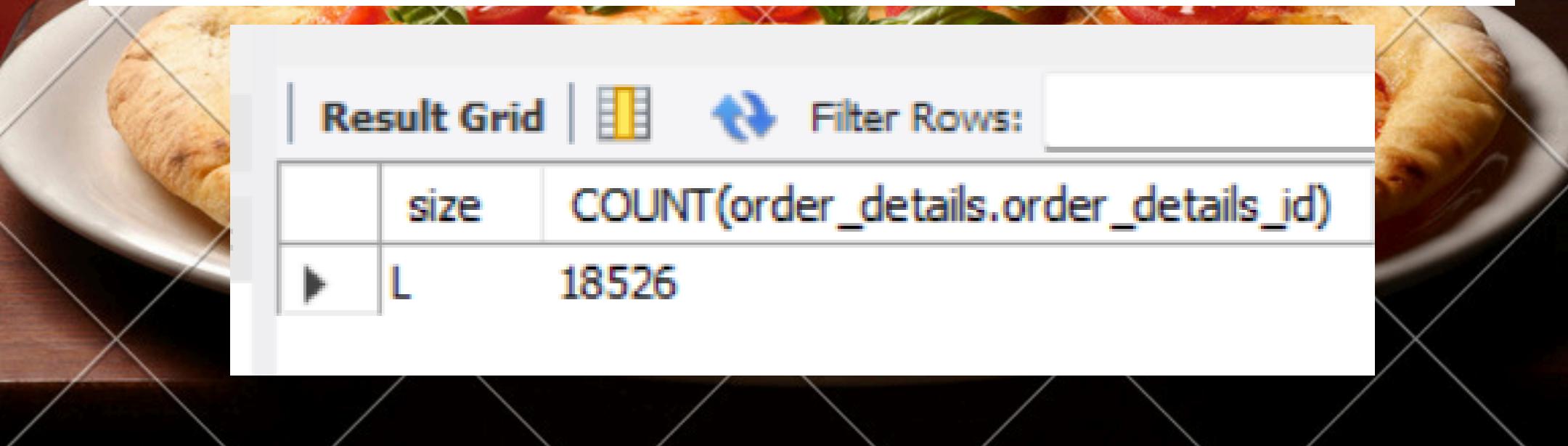


A large image of a pizza on a plate is visible in the background, partially obscured by a data grid. The data grid shows the results of the SQL query, with a single row for the highest-priced pizza.

	pizza_type_id	price
▶	the_greek	35.95

Identify the most common pizza size ordered.

```
SELECT
    pizzas.size, COUNT(order_details.order_details_id)
FROM
    pizzas
    JOIN
        order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY size
LIMIT 1;
```

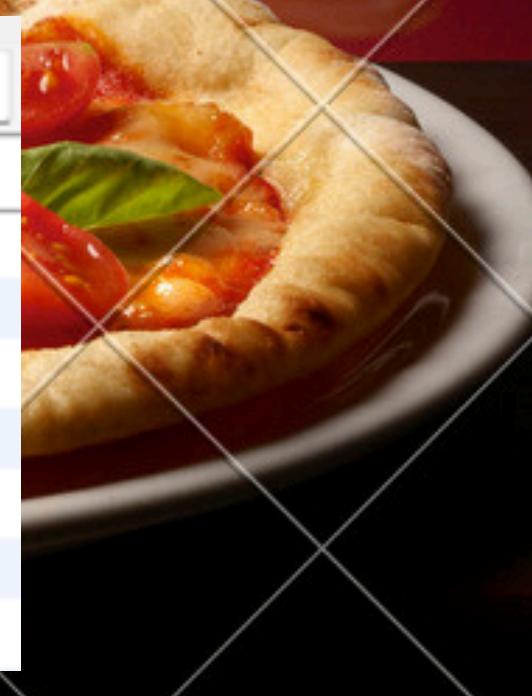


A screenshot of a MySQL Workbench interface showing the results of a SQL query. The results are displayed in a 'Result Grid' table. The table has two columns: 'size' and 'COUNT(order_details.order_details_id)'. The data row shows 'L' in the 'size' column and '18526' in the 'COUNT...' column. The table includes standard database navigation buttons for 'Result Grid', 'Filter Rows', and 'Rows'.

	size	COUNT(order_details.order_details_id)
▶	L	18526

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(pizzas.price * order_details.quantity) 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
JOIN orders
ON orders.order_id = order_details.order_id
GROUP BY pizza_types.category,pizza_types.name) AS a) AS b
WHERE rn<=3;
```



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

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

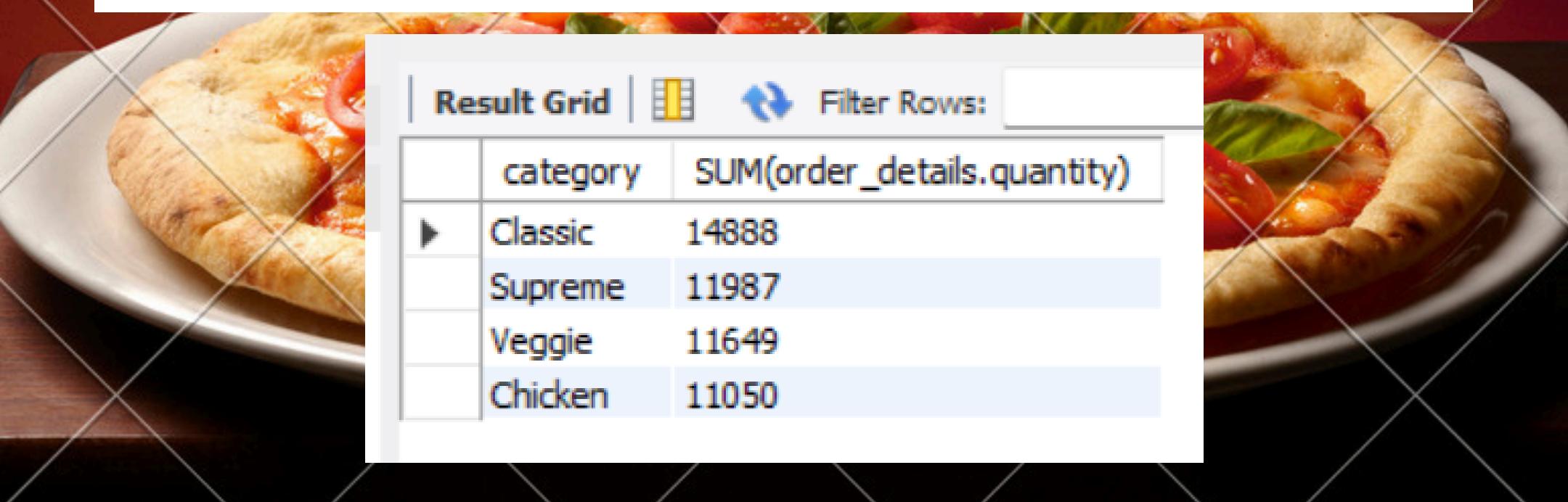
```
SELECT
    pizza_types.pizza_type_id, SUM(order_details.quantity)
FROM
    order_details
    JOIN
    pizzas ON order_details.pizza_id = pizzas.pizza_id
    JOIN
    pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY pizza_types.pizza_type_id
ORDER BY SUM(order_details.quantity)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



	pizza_type_id	SUM(order_details.quantity)
▶	classic_dlx	2453
	bbq_dkn	2432
	hawaiian	2422
	pepperoni	2418
	thai_dkn	2371

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

```
SELECT
    pizza_types.category, SUM(order_details.quantity)
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 SUM(order_details.quantity) DESC;
```



A plate containing four pizzas, each with a different topping combination. From left to right, the pizzas are: Classic (cheese), Supreme (cheese, pepperoni, and mushrooms), Veggie (cheese, bell peppers, and onions), and Chicken (cheese, chicken, and bell peppers).

	category	SUM(order_details.quantity)
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

Determine the distribution of orders by hour of the day.

SELECT

HOUR(time), COUNT(order_id)

FROM

orders

GROUP BY HOUR(time);



Result Grid



Filter Rows:

	HOUR(time)	COUNT(order_id)
▶	11	1231
	12	2520
	13	2455
	14	1472
	15	1468
	16	1920
	17	7336

Result 1 ×



Join relevant tables to find the category-wise distribution of pizzas.

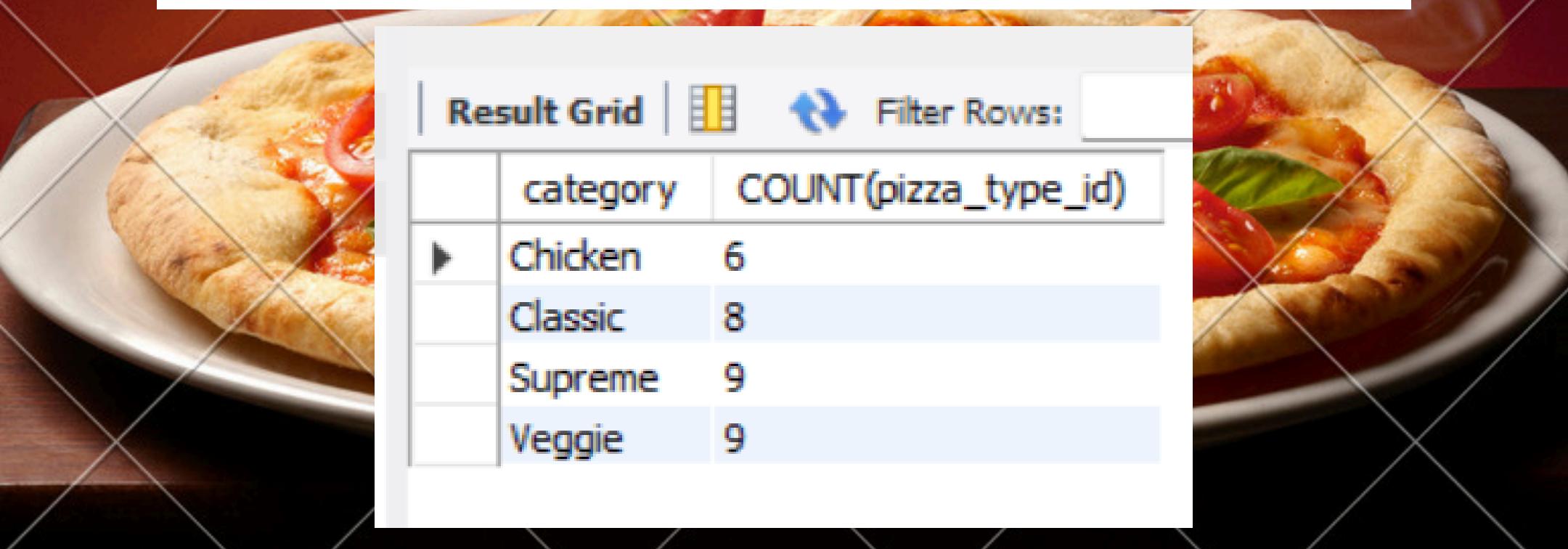
SELECT

category, COUNT(pizza_type_id)

FROM

pizza_types

GROUP BY category;



A screenshot of a database query results window. The window title is 'Result Grid'. It shows a table with two columns: 'category' and 'COUNT(pizza_type_id)'. The data is as follows:

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

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

```
SELECT
```

```
    AVG(quantity_w)
```

```
FROM
```

```
(SELECT
```

```
    orders.date, SUM(order_details.quantity) AS quantity_w
```

```
FROM
```

```
    order_details
```

```
JOIN orders ON order_details.order_id = orders.order_id
```

```
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 orders.date) AS order_quqntity;
```

Result Grid



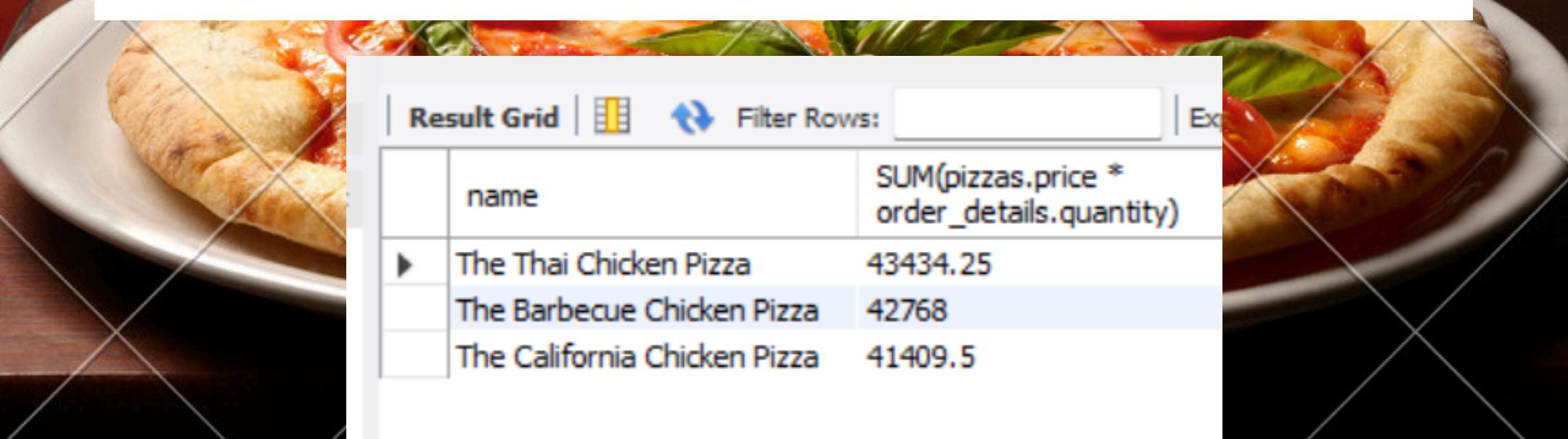
Filter

AVG(quantity_w)
138.4749



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

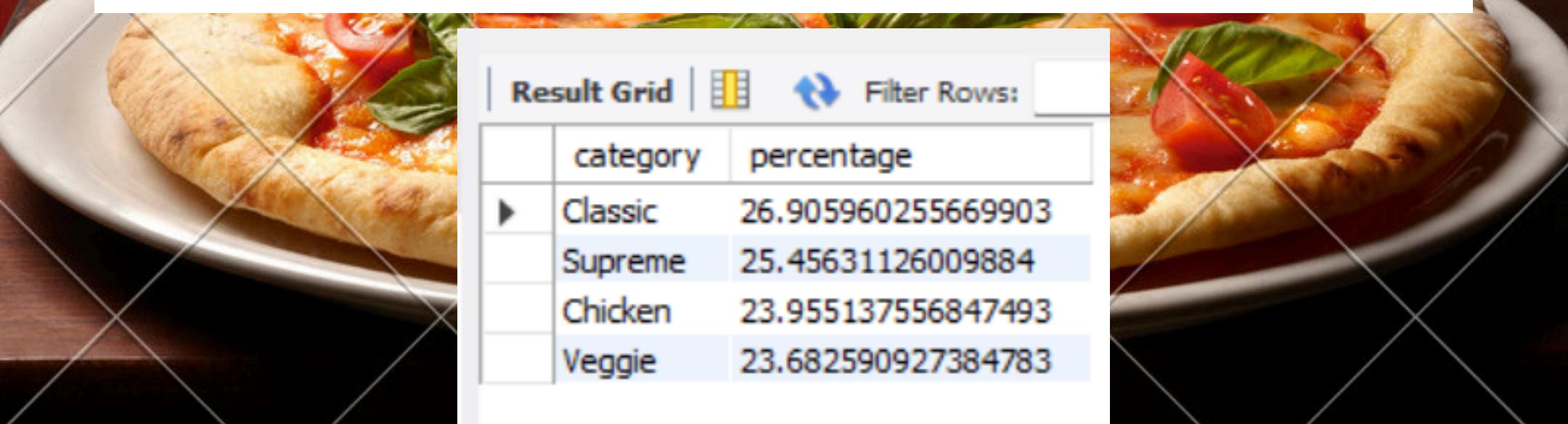
```
SELECT
    pizza_types.name, SUM(pizzas.price * order_details.quantity)
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
        JOIN
    pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY pizza_types.name
ORDER BY SUM(pizzas.price *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



name	SUM(pizzas.price * order_details.quantity)
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.

```
SELECT
    pizza_types.category,
    (SUM(pizzas.price * order_details.quantity) / (SELECT
        SUM(pizzas.price * order_details.quantity)
    FROM
        pizzas
    JOIN
        order_details ON pizzas.pizza_id = order_details.pizza_id)) * 100 AS percentage
FROM
    pizzas
    JOIN
        order_details ON pizzas.pizza_id = order_details.pizza_id
    JOIN
        pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY pizza_types.category
ORDER BY percentage DESC;
```



A photograph of a pizza slice with toppings like tomato and basil, and a whole pizza in the background.

	category	percentage
▶	Classic	26.905960255669903
	Supreme	25.45631126009884
	Chicken	23.955137556847493
	Veggie	23.682590927384783

Analyze the cumulative revenue generated over time..

```
SELECT date , SUM(revenue) OVER(ORDER BY date) AS cum_revenue
FROM
  (SELECT orders.date, SUM(pizzas.price * order_details.quantity) AS revenue
  FROM pizzas
  JOIN order_details
  ON pizzas.pizza_id = order_details.pizza_id
  JOIN orders
  ON orders.order_id = order_details.order_id
  GROUP BY orders.date) AS sales;
```

Result Grid | Filter Rows:

	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
	2015-01-09	21526.4
	2015-01-10	23990.350000000002

