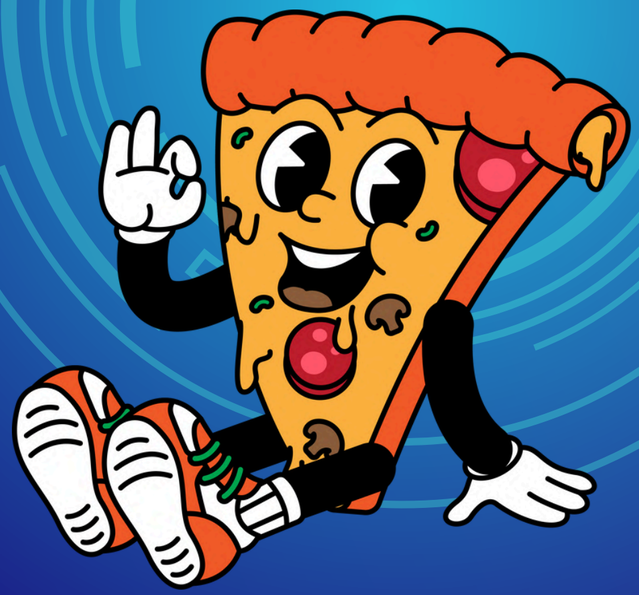
Pizza Hut Sales Analysis - SQL



About the project

This project focuses on analyzing a pizza sales dataset using MySQL.
The goal is to explore ordering patterns, revenue trends, and category-wise insights by writing SQL queries ranging from basic to advanced levels.
The project demonstrates:

* SQL joins
* Aggregations
* Date functions
* Window functions
* CTE
* Real-world problem-solving approach

Dataset Overview

The dataset consists of four tables

1. orders -
Contains details about each order :
- order_id
- date
- time

2. order_details -
Stores line-level order items:
- order_detail_id
- order_id
- pizza_id
- quantity

3. pizzas -
Contains pizza metadata :
- pizza_id
- pizza_type_id
- size
- price

4. pizza_types -
Contains pizza category information:
- pizza_type_id
- name
- category
- ingredients

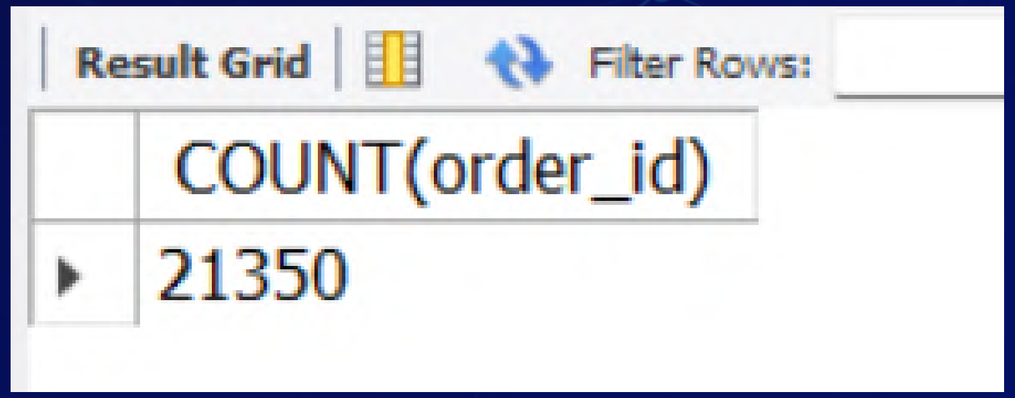
Project Objectives

- Basic
  - Total number of orders
  - Total revenue
  - Highest-priced pizza
  - Most common pizza size
  - Top 5 most-ordered pizza types
- Intermediate
  - Total quantity by pizza category
  - Order distribution by hour
  - Category-wise pizza distribution
  - Average pizzas per day
  - Top 3 pizzas by revenue
- Advanced
  - Revenue % contribution by pizza types
  - Cumulative revenue over time
  - Top pizzas by category revenue

Retrieve the total number of orders placed.

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

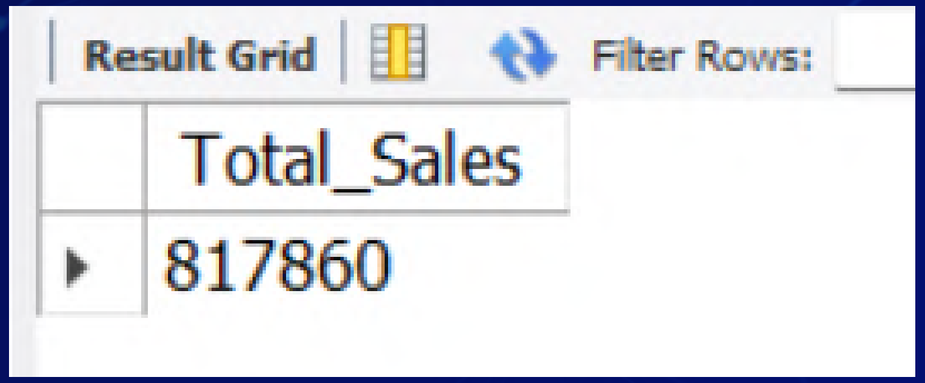
Output



Calculate the total revenue generated from pizza sales.

```sql
SELECT ROUND(SUM(od.quantity * pz.price),0) as Total_Sales
FROM orders_details as od
JOIN pizzas as pz
ON od.pizza_id = pz.pizza_id;
```

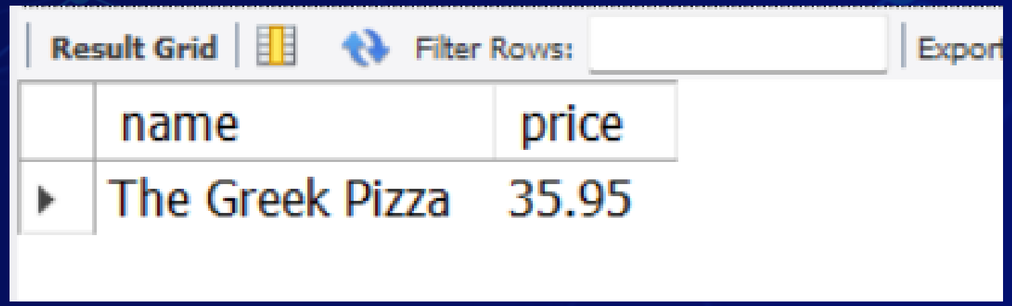
Output



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 price DESC
LIMIT 1;
```

Output

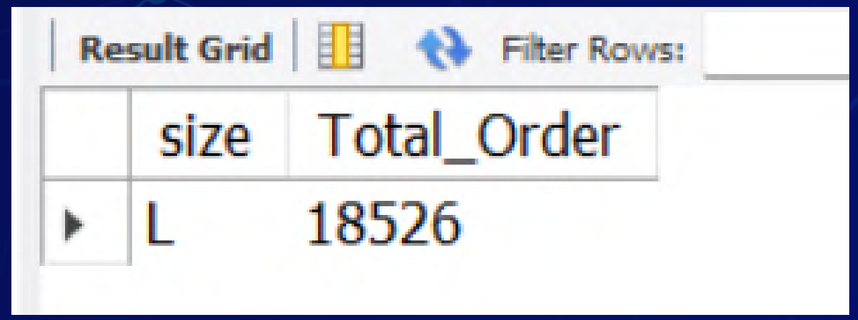


name	price
The Greek Pizza	35.95

Identify the most common pizza size ordered.

```sql
SELECT pizzas.size, COUNT(orders_details.order_detail_id) as Total_Order
FROM orders_details
JOIN pizzas
ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizzas.size
ORDER BY Total_Order DESC LIMIT 1 ;
```

Output

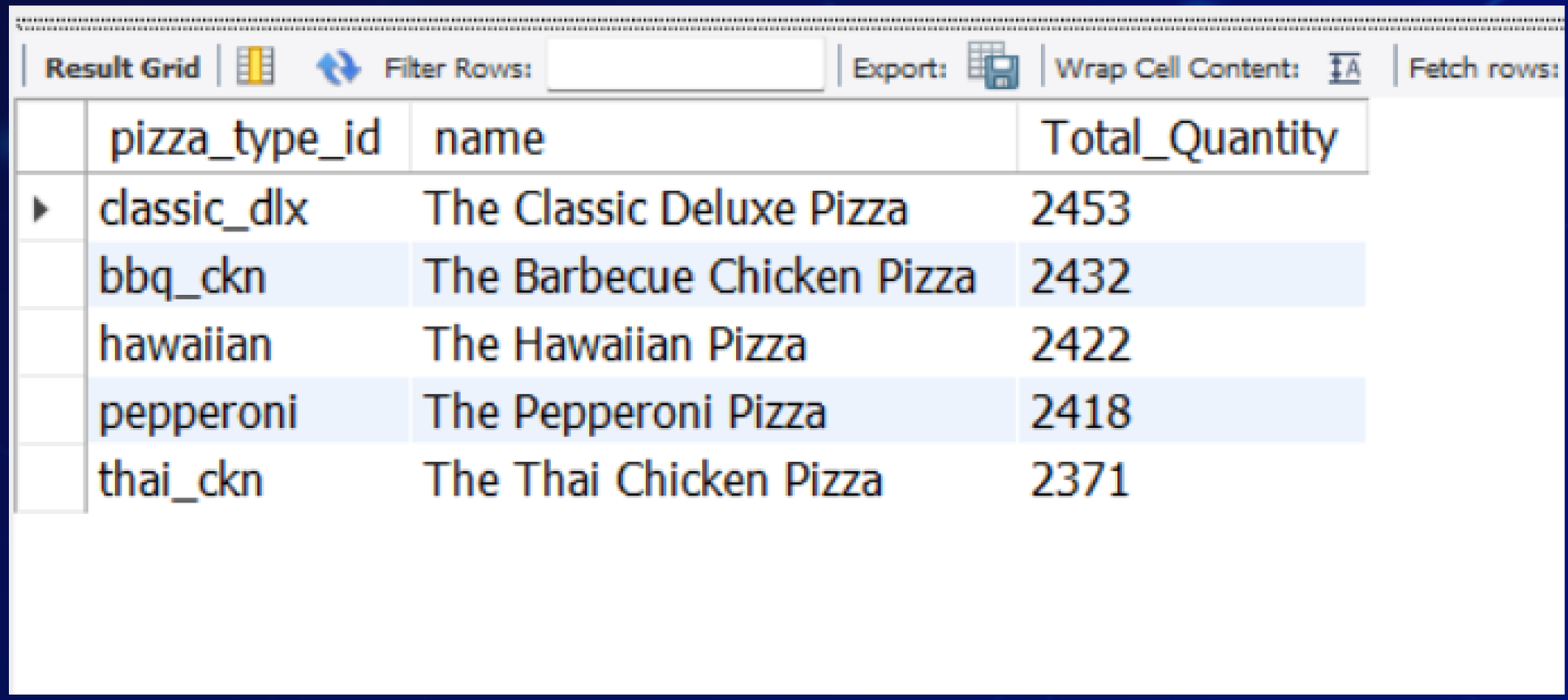


size	Total_Order
L	18526

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

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

Output

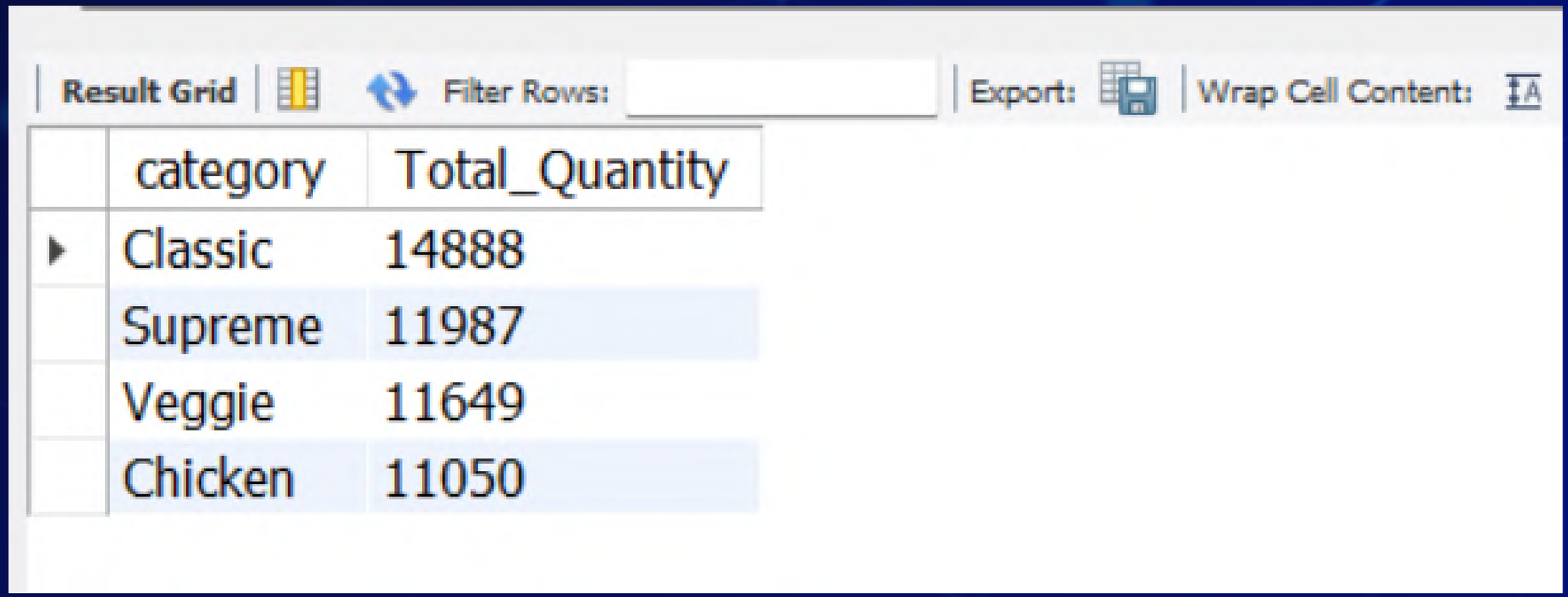


pizza_type_id	name	Total_Quantity
classic_dlx	The Classic Deluxe Pizza	2453
bbq_ckn	The Barbecue Chicken Pizza	2432
hawaiian	The Hawaiian Pizza	2422
pepperoni	The Pepperoni Pizza	2418
thai_ckn	The Thai Chicken Pizza	2371

Total quantity of each pizza category ordered.

```sql
SELECT pizza_types.category, SUM(orders_details.quantity) as Total_Quantity
FROM pizza_types
JOIN pizzas
ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN orders_details
ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY pizza_types.category
ORDER BY Total_Quantity DESC ;
```

Output



category	Total_Quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Determine the distribution of orders by hour of the day.

```sql
SELECT HOUR(`time`), COUNT(order_id) Total_orders
FROM orders
GROUP BY HOUR(`time`)
ORDER BY Total_orders DESC;
```

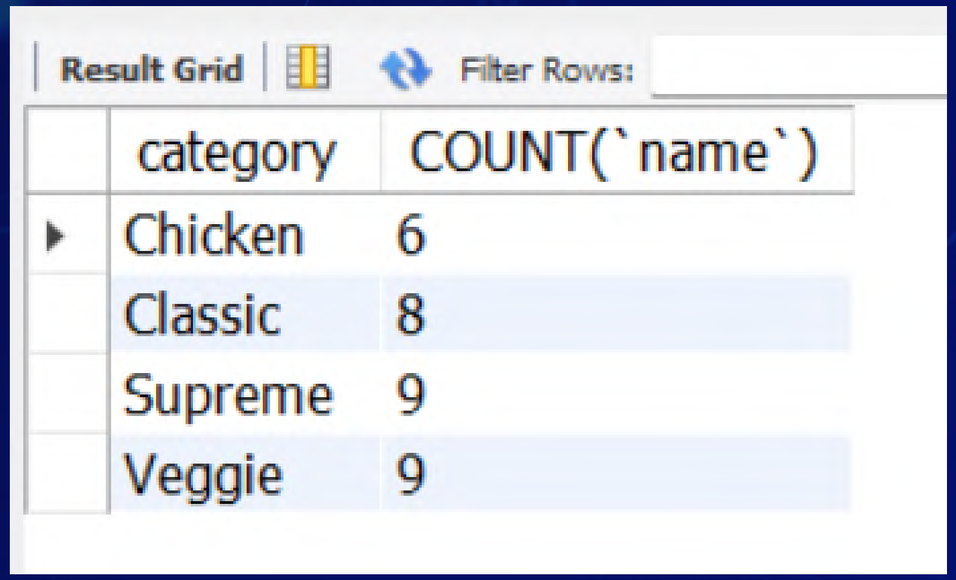
Output

HOUR(`time`)	Total_orders
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

Category-wise distribution of pizzas.

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

Output



Result Grid | Filter Rows:

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

Average number of pizzas ordered per day.

```sql
SELECT ROUND(avg(Total_orders),0) as Average_order FROM
(SELECT orders.`date`, SUM(orders_details.quantity) Total_orders
FROM orders
JOIN orders_details
ON orders.order_id = orders_details.order_id
GROUP BY orders.`date`
ORDER BY Total_orders DESC) as daily_quantity;
```

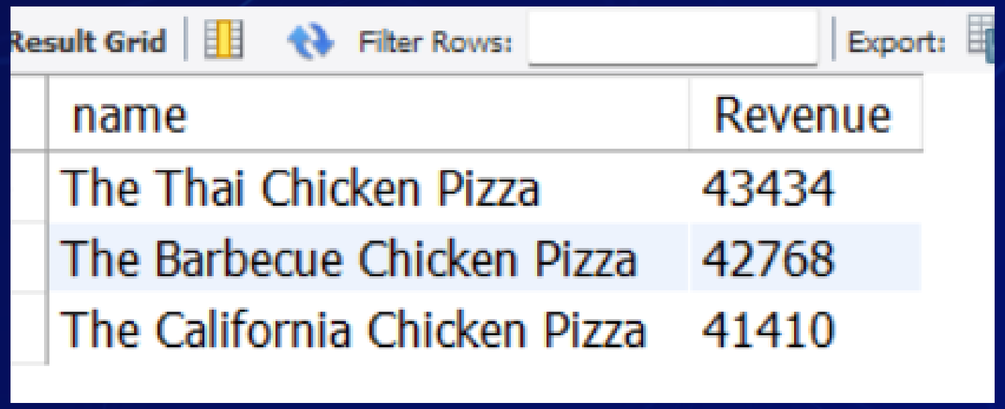
Output

Average_order
138

Top 3 most ordered pizza types based on revenue

```sql
SELECT pizza_types.name, ROUND(SUM(orders_details.quantity* pizzas.price),0) Revenue
FROM orders_details
JOIN pizzas
ON orders_details.pizza_id = pizzas.pizza_id
JOIN pizza_types
ON pizzas.pizza_type_id = pizza_types.pizza_type_id
GROUP BY pizza_types.name
ORDER BY Revenue DESC LIMIT 3;
```

Output

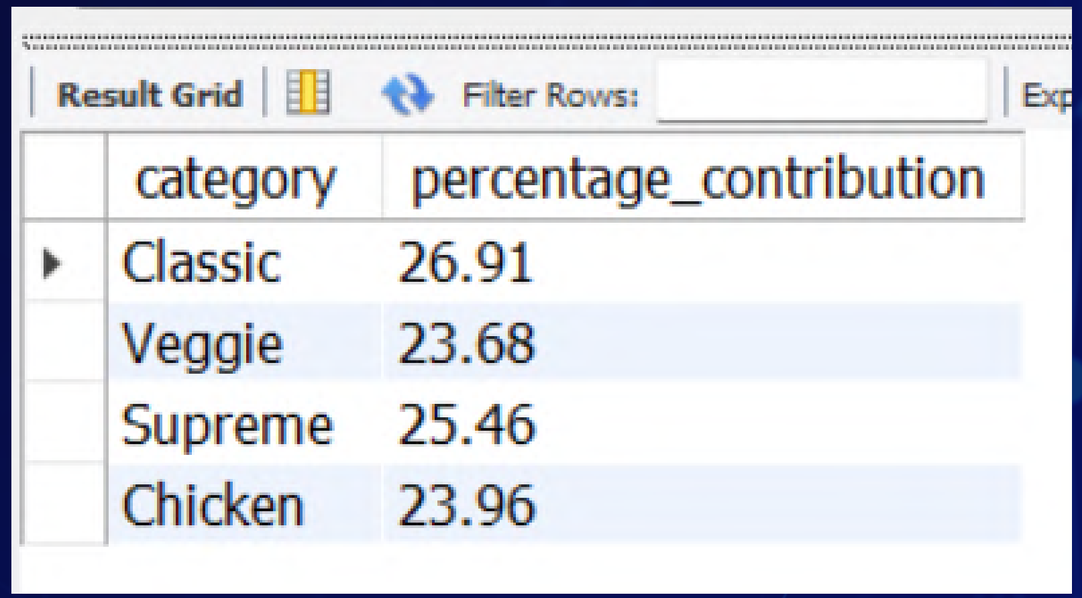


name	Revenue
The Thai Chicken Pizza	43434
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41410

Percentage contribution of each pizza type to total revenue.

```sql
select pizza_types.category, ROUND((SUM(orders_details.quantity * pizzas.price))/
(
select SUM(orders_details.quantity * pizzas.price)
from orders_details
join pizzas
on orders_details.pizza_id = pizzas.pizza_id
) * 100 ,2) as percentage_contribution
from pizzas
join pizza_types
on pizzas.pizza_type_id = pizza_types.pizza_type_id
join orders_details
on pizzas.pizza_id = orders_details.pizza_id
GROUP BY pizza_types.category;
```

Output



category	percentage_contribution
Classic	26.91
Veggie	23.68
Supreme	25.46
Chicken	23.96

Cumulative revenue generated over time.

```sql
WITH Cumulative_revenue AS
(
SELECT orders.date as Dates, ROUND(SUM(orders_details.quantity * pizzas.price),0) Revenue
FROM orders
JOIN orders_details
ON orders.order_id = orders_details.order_id
JOIN pizzas
ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY orders.date
)
SELECT Dates, Revenue, SUM(Revenue) OVER (ORDER BY Dates ASC) as Roll_Sum
FROM Cumulative_revenue
;
```

Output

Result Grid | Filter Rows: | Export:

Dates	Revenue	Roll_Sum
2015-01-01	2714	2714
2015-01-02	2732	5446
2015-01-03	2662	8108
2015-01-04	1755	9863
2015-01-05	2066	11929
2015-01-06	2429	14358
2015-01-07	2202	16560
2015-01-08	2838	19398
2015-01-09	2127	21525
2015-01-10	2464	23989
2015-01-11	1872	25861
2015-01-12	1919	27780
2015-01-13	2050	29830
2015-01-14	2527	32357

The Top 3 most ordered pizza types based on revenue for each pizza category.

```sql
SELECT category, name, revenue, Rank_wise
FROM
(
    WITH Ranking AS
    (
    SELECT pizza_types.category , pizza_types.name, SUM(orders_details.quantity * pizzas.price) Revenue
    FROM pizza_types
        JOIN pizzas
            ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN orders_details
            ON orders_details.pizza_id = pizzas.pizza_id
    GROUP BY pizza_types.category, pizza_types.name
    )
    SELECT category, name, Revenue, dense_rank() OVER (partition by category Order By Revenue DESC) as Rank_wise
    FROM Ranking
) AS a
WHERE Rank_wise <= 3;
```

Output

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

category	name	revenue	Rank_wise
Chicken	The Thai Chicken Pizza	43434.25	1
Chicken	The Barbecue Chicken Pizza	42768	2
Chicken	The California Chicken Pizza	41409.5	3
Classic	The Classic Deluxe Pizza	38180.5	1
Classic	The Hawaiian Pizza	32273.25	2
Classic	The Pepperoni Pizza	30161.75	3
Supreme	The Spicy Italian Pizza	34831.25	1
Supreme	The Italian Supreme Pizza	33476.75	2
Supreme	The Sicilian Pizza	30940.5	3
Veggie	The Four Cheese Pizza	32265.70000000065	1
Veggie	The Mexicana Pizza	26780.75	2
Veggie	The Five Cheese Pizza	26066.5	3

Summary of Insights :

- Total orders placed : 21350
- Total revenue generated : 817860
- Most popular pizza size : L
- Highest revenue-generating pizza : The Thai Chicken Pizza
- Busiest hour of the day : 12
- Category with highest demand : Classic
- Total pizzas sold per day (average) : 138

Conclusion :-

This SQL mini-project helped analyze pizza sales data from multiple angles, including popularity, trends, and revenue distribution. Such analysis is commonly used in:

- Restaurants
- Retail
- E-commerce
- Inventory management
- Business intelligence