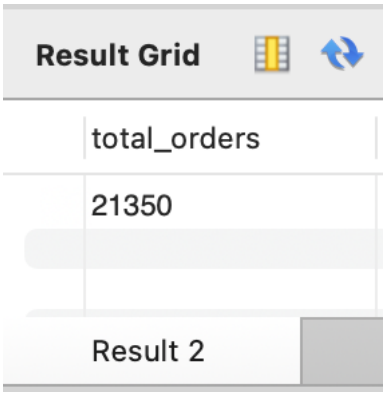
Pizza Sales Analysis Using MYSQL

This project involved analysing pizza sales data using SQL to uncover insights such as total orders, revenue, top-selling pizzas, and customer ordering patterns. By joining and aggregating data across multiple related tables, I answered business questions ranging from basic metrics to advanced revenue breakdowns by category and time.

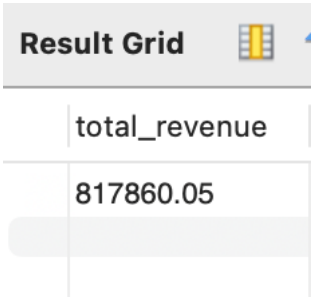
Q1. Retrieve the total number of orders placed.

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



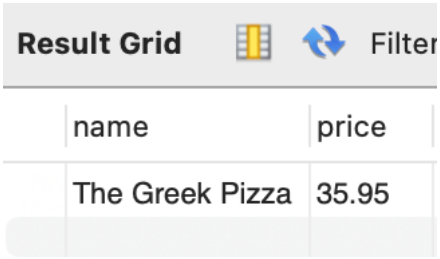
Q2. Calculate the total revenue from pizza sales.

```
SELECT
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS total_revenue
FROM
    order_details
        JOIN
    pizzas ON order_details.pizza_id = pizzas.pizza_id;
```



Q3. 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 price DESC
LIMIT 1;
```

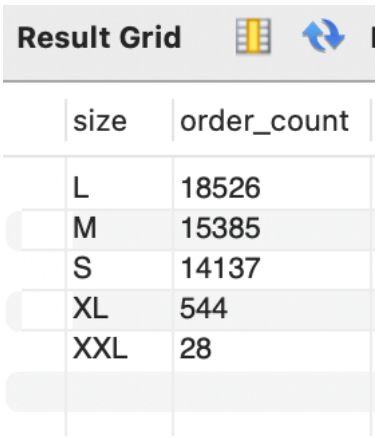


Result Grid Filter

name	price
The Greek Pizza	35.95

Q4. 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;
```

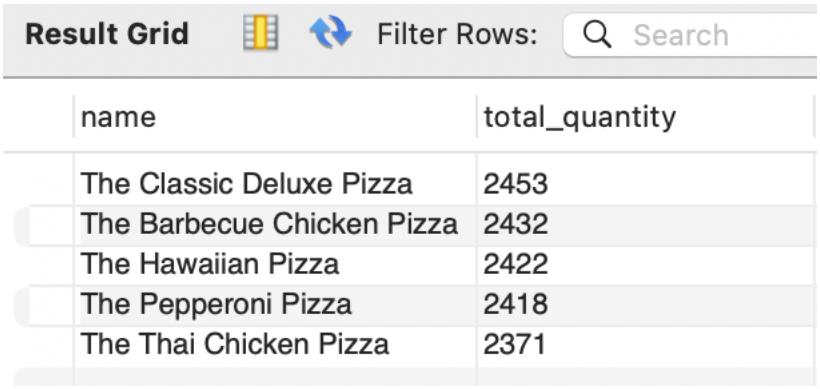


Result Grid

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

Q5. List the top 5 most ordered pizzas along with their quantities.

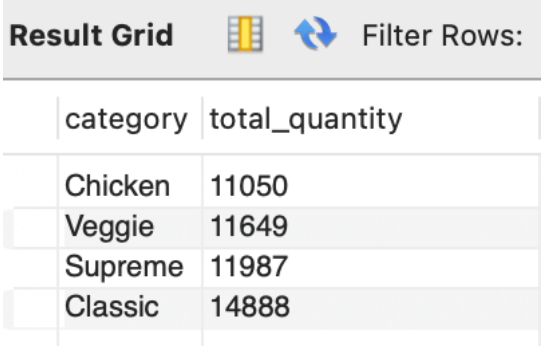
```
SELECT
    pizza_types.name,
    SUM(order_details.quantity) AS total_quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY total_quantity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


Result Grid Filter Rows: Search

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

Q6. Find the quantity of each pizza category ordered.

```
SELECT
    pizza_types.category,
    SUM(order_details.quantity) AS total_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 total_quantity;
```


Result Grid Filter Rows:

category	total_quantity
Chicken	11050
Veggie	11649
Supreme	11987
Classic	14888

Q7. Determine the distribution of orders by hour of the day.

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

Result Grid Filter

hour	COUNT(order_i...
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009
20	1642
21	1198
22	663
23	28
10	8
9	1

Q8. Category wise distribution of pizzas

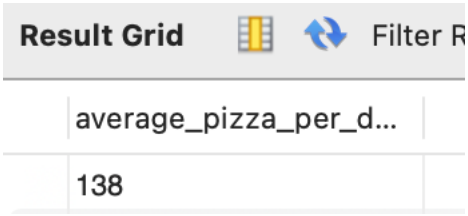
```
SELECT
    category, COUNT(name)
FROM
    pizza_types
GROUP BY category;
```

Result Grid Filter

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

Q9. Group the orders by date and calculate the average number pizzas ordered per day.

```
SELECT
    ROUND(AVG(quantity),0) AS average_pizza_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;
```



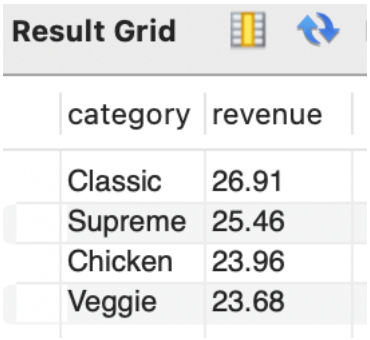
Q10. Determine top 3 pizza types based on revenue.

```
SELECT
    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.name
ORDER BY revenue
LIMIT 3;
```

name	revenue
The Brie Carre Pizza	11588.4999999999
The Green Garden Pizza	13955.75
The Spinach Supreme Pizza	15277.75

Q11. Calculate the percentage distribution 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_revenue
FROM
    order_details
        JOIN
    pizzas ON order_details.pizza_id = pizzas.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;
```

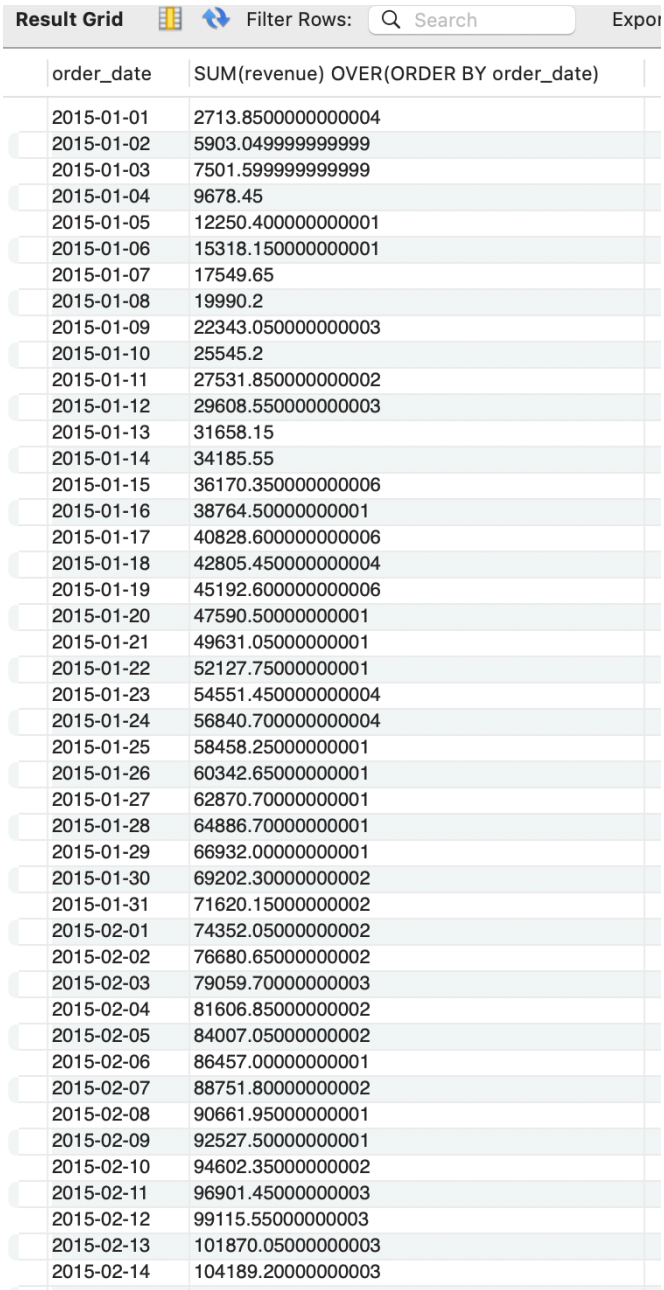


Result Grid

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

Q12. Analyse the cumulative revenue generated over time.

```
SELECT order_date, SUM(revenue) OVER(ORDER BY order_date)
FROM
(SELECT orders.order_date,
SUM(order_details.quantity * pizzas.price) 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;
```


Result Grid Filter Rows: Search Expor

order_date	SUM(revenue) OVER(ORDER BY order_date)
2015-01-01	2713.8500000000004
2015-01-02	5903.049999999999
2015-01-03	7501.599999999999
2015-01-04	9678.45
2015-01-05	12250.400000000001
2015-01-06	15318.150000000001
2015-01-07	17549.65
2015-01-08	19990.2
2015-01-09	22343.050000000003
2015-01-10	25545.2
2015-01-11	27531.850000000002
2015-01-12	29608.550000000003
2015-01-13	31658.15
2015-01-14	34185.55
2015-01-15	36170.350000000006
2015-01-16	38764.50000000001
2015-01-17	40828.600000000006
2015-01-18	42805.450000000004
2015-01-19	45192.600000000006
2015-01-20	47590.50000000001
2015-01-21	49631.05000000001
2015-01-22	52127.75000000001
2015-01-23	54551.450000000004
2015-01-24	56840.700000000004
2015-01-25	58458.25000000001
2015-01-26	60342.65000000001
2015-01-27	62870.70000000001
2015-01-28	64886.70000000001
2015-01-29	66932.00000000001
2015-01-30	69202.30000000002
2015-01-31	71620.15000000002
2015-02-01	74352.05000000002
2015-02-02	76680.65000000002
2015-02-03	79059.70000000003
2015-02-04	81606.85000000002
2015-02-05	84007.05000000002
2015-02-06	86457.00000000001
2015-02-07	88751.80000000002
2015-02-08	90661.95000000001
2015-02-09	92527.50000000001
2015-02-10	94602.35000000002
2015-02-11	96901.45000000003
2015-02-12	99115.55000000003
2015-02-13	101870.05000000003
2015-02-14	104189.20000000003

Q13. Determine the top 3 ordered pizza types based on revenue for each category.

```
SELECT category, 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(order_details.quantity * pizzas.price) AS revenue
        FROM pizzas
        JOIN pizza_types 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.category, pizza_types.name
    ) AS a
) AS b
WHERE rn <= 3;
```

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