

# Pizza Sales Analysis

Robin Raj

## Introduction

### Project Title: Pizza Sales Data Analysis using MySQL

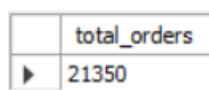
In this project, I utilized MySQL to analyze a pizza sales dataset, which consists of four main tables: **pizza**, **pizza\_types**, **orders**, and **order\_details**. This structured data allowed me to explore different aspects of sales performance, customer preferences, and product popularity. The project was divided into three analytical levels—Basic, Intermediate, and Advanced—each designed to address specific business questions and provide meaningful insights.

In the **Basic** analysis, I retrieved foundational insights, such as the total number of orders placed, overall revenue, the most commonly ordered pizza size, and the top 5 pizzas by order quantity. The **Intermediate** section involved more complex joins and aggregations to explore trends like hourly order distributions, average daily pizza orders, and category-wise order distribution. In the **Advanced** section, I calculated the revenue contributions of each pizza type, analyzed cumulative revenue over time, and identified the top revenue-generating pizzas within each category.

Through this project, I demonstrated my ability to manage SQL queries of increasing complexity to uncover key business insights. This analysis not only showcases my technical proficiency but also provides a comprehensive view of pizza sales dynamics, from basic sales metrics to more advanced revenue patterns.

### 1. Retrieve the total number of orders placed

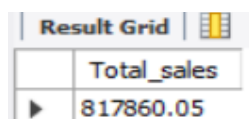
```
select count(order_id) as total_orders from orders;
```



	total_orders
▶	21350

### 2. Calculate the total revenue generated from pizza sales

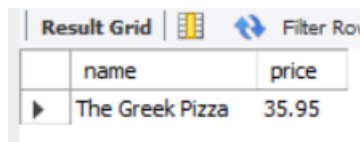
```
SELECT  
    ROUND(SUM(orders_details.quantity * pizzas.price),  
          2) AS Total_sales  
FROM  
    orders_details  
    JOIN  
    pizzas ON pizzas.pizza_id = orders_details.pizza_id;
```



	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;
```

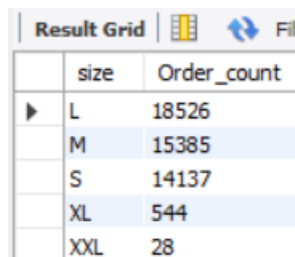


The screenshot shows a 'Result Grid' window with a table containing two columns: 'name' and 'price'. The first row of data shows 'The Greek Pizza' with a price of '35.95'.

	name	price
▶	The Greek Pizza	35.95

### 4. Identify the most common pizza size ordered

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





The screenshot shows a 'Result Grid' window with a table containing two columns: 'size' and 'Order\_count'. The results are ordered by 'Order\_count' in descending order.

	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(orders_details.quantity) AS quantity
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.name
```

ORDER BY quantity DESC  
LIMIT 5;

Result Grid     Filter Rows: <input type="text"/>		
	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

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

```
SELECT
    pizza_types.category,
    SUM(orders_details.quantity) AS quantity
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
ORDER BY quantity DESC
LIMIT 5;
```

	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 Hour, COUNT(order_id) as Order_count
FROM
    orders
GROUP BY HOUR(order_time) order by Order_count desc;
```

	Hour	Order_count
▶	12	2520
	13	2455
	18	2399
	17	2336
	19	2009
	16	1920
	20	1642
	14	1472
	15	1468

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

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

	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_order_perday
FROM
    (SELECT
        orders.order_date,
        COUNT(orders_details.quantity) AS quantity
    FROM
        orders
    JOIN orders_details ON orders.order_id = orders_details.order_id
    GROUP BY orders.order_date) AS order_quantity;
```

	avg_order_perday
▶	136

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

```

SELECT
    pizza_types.name,
    SUM(orders_details.quantity * pizzas.price) AS 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.name
ORDER BY Revenue DESC
LIMIT 3;

```

	name	Revenue
▶	The Thai Chicken Pizza	43434.25
	The Barbecue Chicken Pizza	42768
	The California Chicken Pizza	41409.5

#### 11. Calculate the percentage contribution of each pizza type to total revenue

```

SELECT
    pizza_types.category,
    ROUND((SUM(orders_details.quantity * pizzas.price) / (SELECT
        ROUND(SUM(orders_details.quantity * pizzas.price),
            2) AS Total_sales
    FROM
        orders_details
        JOIN
        pizzas ON pizzas.pizza_id = orders_details.pizza_id)) * 100,
    2) AS 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
ORDER BY Revenue DESC
;

```

	category	Revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

#### 12. Analyze the cumulative revenue generated over time

```

select order_date, sum(revenue) over(order by order_date) as cum_revenue
from(select orders.order_date, sum(orders_details.quantity * pizzas.price) as revenue
from orders_details join pizzas on orders_details.pizza_id = pizzas.pizza_id
join orders on orders.order_id = orders_details.order_id group by orders.order_date) as Sales;

```

	order_date	cum_revenue
►	2015-01-01 00:00:00	2713.8500000000004
	2015-01-02 00:00:00	5445.75
	2015-01-03 00:00:00	8108.15
	2015-01-04 00:00:00	9863.6
	2015-01-05 00:00:00	11929.55
	2015-01-06 00:00:00	14358.5
	2015-01-07 00:00:00	16560.7
	2015-01-08 00:00:00	19399.05
	2015-01-09 00:00:00	21526.4

### 13. 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((orders_details.quantity) * pizzas.price) as 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) as a) as b
where rn<=3 ;

```

	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
	The Sicilian Pizza	30940.5
	The Four Cheese Pizza	32265.700000000065