# PIZZA SALES ANALYSIS USING MYSQL

**Submitted By:**
**Gaurav Diwate**
**(Data Analyst Fresher)**

**Tool Used:**

- **MySQL**

**MAY 2025**

# Introduction & Dataset Overview

## Introduction

This project focuses on analyzing pizza sales data using MySQL.
The goal is to extract key business insights such as total revenue, top-selling pizzas, customer preferences, and monthly trends.
This helps in understanding business performance and making data-driven decisions.

## Dataset Description

| Column Name | Description |
|---|---|
| order_id | Unique ID for each customer order |
| order_date | Date and time when the order was placed |
| pizza_name | Name of the pizza |
| quantity | Number of pizzas sold |
| price | Price per pizza |

# Business Questions Solved Using SQL

**1) RetriEve the total number of orders placed.**

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

Result Grid

| | total_orders |
|---|---|
| ▶ | 21350 |

# 2) Calculate the Total Revenue Generated from Pizza Sales.

```sql
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;
```

Result Grid

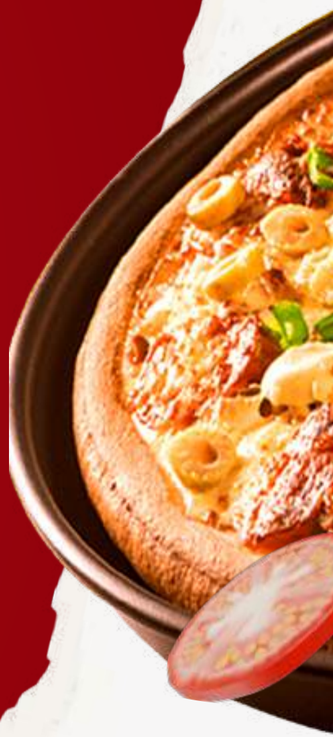
| | total_sales |
|---|---|
| ▶ | 817860.05 |

## 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;
```

Result Grid | Filter Ro

| | name | price |
|---|---|---|
| ▶ | The Greek Pizza | 35.95 |



# 4) Identify the most Frequently Ordered Pizza Size.

```sql
SELECT quantity, count(order_details_id)
FROM order_details GROUP BY quantity;


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 |

# 5) List the Top 5 Most Ordered Pizza Types 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 |

# 6) 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 | Filte

| category | quantity |
|---|---|
| Classic | 14888 |
| Supreme | 11987 |
| Veggie | 11649 |
| Chicken | 11050 |

# 7) Determine the distribution of orders by hour of the day.

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

Result Grid | Filter Rows:

| hour | order_count |
|---|---|
| 11 | 1231 |
| 12 | 2520 |
| 13 | 2455 |
| 14 | 1472 |
| 15 | 1468 |
| 16 | 1920 |
| 17 | 2336 |
| 18 | 2399 |

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

```
SELECT category, count(name) from pizza_types
group by category
```

Result Grid | Filter Row

| | 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.

```sql
SELECT
    ROUND(AVG(quantity), 0) as avg_pizza_orderd_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_orderd_per_day |
|---|---|
| ▶ | 138 |

# 10) 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 |

# 11) 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 | Filter

| | category | revenue |
|---|---|---|
| ▶ | Veggie | 23.68 |
| | Chicken | 23.96 |
| | Supreme | 25.46 |
| | Classic | 26.91 |

# 12) Analyze the cumulative revenue generated over time

```sql
select order_date,
sum(revenue) over(order by order_date) as cum_revenue
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:

| 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 |

# 13) Determine the top 3 most ordered pizza types based on revenue for each pizza category.

```sql
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((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.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 |
| The Italian Supreme Pizza | 33476.75 |
| The Sicilian Pizza | 30940.5 |
| The Four Cheese Pizza | 32265.70000000065 |
| The Mexicana Pizza | 26780.75 |
| The Five Cheese Pizza | 26066.5 |

# THANK YOU!

PIZZA WITH GAURAV


- **Name :** Gaurav Diwate
- **Email :** gdiwate7210@gmailcom
- **LinkedIn :** https://www.linkedin.com/in/gauravdiwate/
- **GitHub :** https://github.com/Gauravdiwate
- **Contact :** 9766549521