

Pizza Sales Database Management System

Payal Gupta
Math & Computing , IIT Guwahati





CONTENTS

01 Introduction

02 Database Design

03 Database Schema

04 Key Queries & Codes

05 Data Analysis & Conclusion



01

Introduction

This project involves creating a MySQL database to manage and analyze pizza sales data, focusing on customer orders, sales trends, and inventory management.

The objective is to streamline sales data management and provide insights into sales trends, popular pizzas, and customer preferences.

Database Design



Entity Relationship (ER) Diagram:



dbdiagram.io

The ER diagram illustrates the relationships between four tables: *Pizzas*, *Pizza_Types*, *Orders*, and *Order_Details*. *Pizzas* are categorized by *Pizza_Types*, *Orders* contain multiple *Order_Details*, and each *Order_Detail* links a specific pizza to an order. This structure ensures efficient data management for tracking pizza sales and order details. Primary keys are shown as 

Database Schema



- *pizzas (pizza_id : text, pizza_type_id : text, size : text, price : double)*
- *pizza_types (pizza_type_id : text, name : text , category : text, ingredients : text)*
- *orders (order_id : int , order_date : date , order_time : time)*
- *orders_details (order_details_id : int ,order_id : int , pizza_id : text ,quantity : int)*

Key Queries & Codes



Retrieve the total number of orders placed.

```
1 -- Retrieve the total number of orders placed  
2 • SELECT COUNT(order_id) AS total_order FROM orders;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	total_order			
▶	21350			

Key Queries & Codes



Calculate the total revenue generated from pizza sales.

```
1 -- Calculate the total revenue generated from pizza sales.  
2  
3 • SELECT  
4     ROUND(SUM(pizzas.price * orders_details.quantity),2)  
5         AS total_revenue  
6 FROM  
7 pizzas  
8     JOIN  
9 orders_details ON pizzas.pizza_id = orders_details.pizza_id;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	total_revenue			
▶	817860.05			

Key Queries & Codes



Identify the highest-priced pizza.

```
1 -- Identify the highest-priced pizza.  
2  
3 • SELECT  
4     pizza_types.name, pizzas.price  
5 FROM  
6     pizza_types  
7         JOIN  
8     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id  
9 ORDER BY pizzas.price DESC  
10 LIMIT 1;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
name	price				
The Greek Pizza	35.95				

Key Queries & Codes



Identify the most common pizza size ordered.

```
1 -- Identify the most common pizza size ordered.  
2  
3 • SELECT  
4     pizzas.size,  
5     COUNT(orders_details.order_details_id) AS order_count  
6 FROM  
7     pizzas|  
8     JOIN  
9     orders_details ON pizzas.pizza_id = orders_details.pizza_id  
10 GROUP BY pizzas.size  
11 ORDER BY order_count DESC;
```

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

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

Key Queries & Codes



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

```
1 -- List the top 5 most ordered pizza types along with their quantities.
2
3 • SELECT
4     pizza_types.name, SUM(orders_details.quantity) AS counts
5 FROM
6     pizza_types
7     JOIN
8     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
9     JOIN
10    orders_details ON pizzas.pizza_id = orders_details.pizza_id
11   GROUP BY pizza_types.name
12   ORDER BY counts DESC
13   LIMIT 5;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
	name	counts			
▶	The Classic Deluxe Pizza	2453			
	The Barbecue Chicken Pizza	2432			
	The Hawaiian Pizza	2422			
	The Pepperoni Pizza	2418			
	The Thai Chicken Pizza	2371			

Key Queries & Codes



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

```
1  -- Join the necessary tables to find the total quantity of each pizza category ordered.
2
3 • SELECT
4      pizza_types.category,
5      SUM(orders_details.quantity) AS total_quantity
6  FROM
7      pizza_types
8      JOIN
9      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
10     JOIN
11     orders_details ON orders_details.pizza_id = pizzas.pizza_id
12 GROUP BY pizza_types.category
13 ORDER BY total_quantity DESC;
```

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

category	total_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Key Queries & Codes



Determine the distribution of orders by hour of the day.

```
1 -- Determine the distribution of orders by hour of the day.  
2  
3 • SELECT  
4     HOUR(order_time) AS hours, COUNT(order_id) AS total_order  
5 FROM  
6     orders  
7 GROUP BY HOUR(order_time);
```

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

hours	total_order
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

Key Queries & Codes



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

```
1 -- Join relevant tables to find the category-wise distribution of pizzas.  
2  
3 • SELECT  
4     category, COUNT(name) AS total_types  
5 FROM  
6     pizza_types  
7 GROUP BY category;
```

Result Grid	
category	total_types
Chicken	6
Classic	8
Supreme	9
Veggie	9

Key Queries & Codes



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

```
1  -- Group the orders by date and calculate the average number of pizzas ordered per day.
2
3 • SELECT
4      ROUND(AVG(qty), 0) AS avg_per_day
5  FROM
6    (SELECT
7        orders.order_date, SUM(orders_details.quantity) AS qty
8      FROM
9        orders
10     JOIN orders_details ON orders.order_id = orders_details.order_id
11     GROUP BY orders.order_date) AS order_qty;
```

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

avg_per_day
138

Key Queries & Codes



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

```
1  -- Determine the top 3 most ordered pizza types based on revenue.
2
3 • SELECT
4      pizza_types.name,
5      SUM(pizzas.price * orders_details.quantity) AS revenue
6  FROM
7      pizza_types
8      JOIN
9      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
10     JOIN
11     orders_details ON orders_details.pizza_id = pizzas.pizza_id
12  GROUP BY pizza_types.name
13  ORDER BY revenue DESC
14  LIMIT 3;
```

Result Grid | Filter Rows: _____ | Export: _____ | Wrap Cell Content: _____ | Fetch rows: _____

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

Data Analysis Summary



- **Total Number of Orders Placed:** 21,350
- **Total Revenue:** \$817,860.05
- **Highest Priced Pizza:** The Greek Pizza at \$35.95
- **Most Ordered Pizza Size:** Large (L) size is the most ordered, while XXL size is the least ordered.
- **Most Ordered Pizza:** The Classic Deluxe Pizza
- **Most Popular Pizza Category:** Classic category
- **Peak Ordering Times:** Most pizzas are ordered between 12:00 PM to 1:00 PM and 5:00 PM to 7:00 PM.
- **Most Liked Pizza Type:** Chicken pizzas are the most popular among customers
- **Average Number of Pizzas Ordered Per Day:** 138
- **Top 3 Ordered Pizzas:**
 - I. The Thai Chicken Pizza
 - II. The Barbecue Chicken Pizza
 - III. The California Chicken Pizza

Conclusion:



- I successfully created a MySQL database to manage and analyze pizza sales.
- The database design captures important details about pizzas, orders, and sales.
- SQL queries helped us understand which pizzas are most popular and when sales are highest.
- Future improvements could include adding customer feedback and real-time data analysis.

THANK YOU

