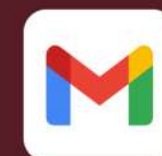


PIZZA SALES REPORT

19 AUG, 2025

Presented by:
Saima Shaikh



AGENDA

- Introduction
- Problem Statement
- SQL Concepts Applied
- Summarized Insights
- Schema Diagram
- Questions
- Conclusion
- Recommendations





pizza sales

INTRODUCTION

I'm Saima Shaikh, a Data Analyst skilled in SQL and data visualization. In this project ,I utilized SQL queries to answer key questions related to pizza sales, uncovering trends, customer preferences, and sales patterns. This analysis provides insights that can support data-driven decisions, optimize inventory, and enhance business performance.



pizza sales



PROBLEM STATEMENT

The project aims to answer essential business questions about pizza sales, such as:

- Identifying top-selling pizzas
- Understanding customer preferences
- Evaluating order patterns

These insights empower stakeholders to make data-driven choices to boost sales and refine marketing strategies



SQL CONCEPTS APPLIED

JOINS

Extensively used to integrate data across multiple tables, enabling thorough analysis of customer orders and sales trends.

AGGREGATION

Functions like COUNT, SUM, and AVG support tasks such as identifying top-performing pizzas, customer frequency, and sales metrics.

SUBQUERIES

Subqueries are used to answer more complex questions, such as identifying peak sales times or analyzing customer demographics.

COMMON TABLE EXPRESSIONS (CTES)

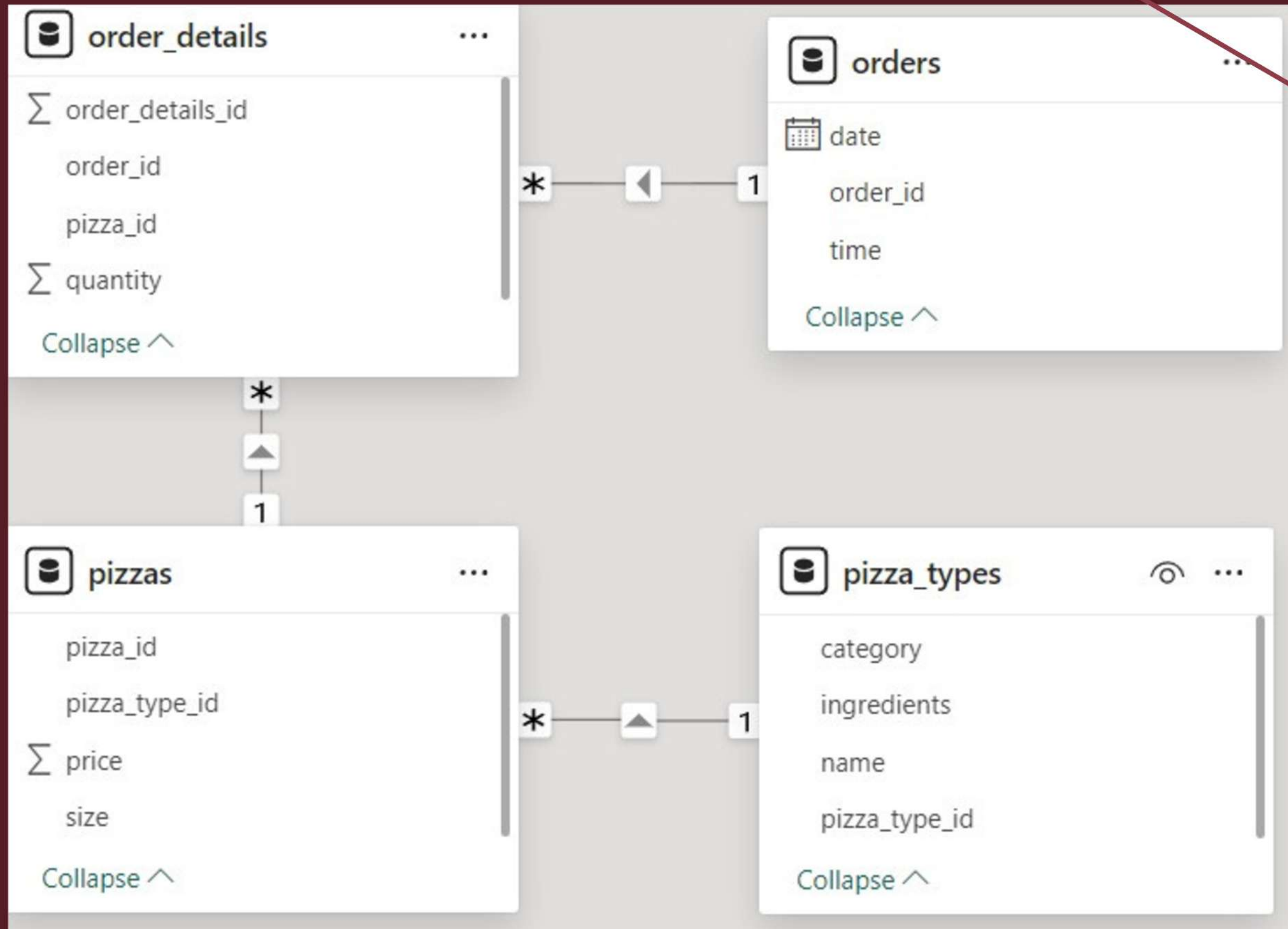
CTEs help simplify complex queries, improving readability and enabling streamlined analysis across different sales dimensions.



SUMMARIZED INSIGHTS

- **Top-Selling Pizzas:**
Identified most popular pizza types for marketing.
- **Customer Preferences:**
Order frequency & pizza preferences helped target customers.
- **Sales Trends:**
Found peak order times for better scheduling.
- **Inventory Optimization:**
Ingredient demand insights guided stock planning.
- **Revenue Patterns:**
Identified high-value pizzas & customers for focus.

SCHEMA DIAGRAM



1. RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

```
SELECT
    COUNT(order_id) AS Total_orders
FROM
    pizzahut.orders;
```

Result Grid	
	Total_orders
▶	21350



2. CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

```
SELECT
    ROUND(SUM(order_details.quantity * pizzas.price),
          2) AS total_sales
FROM
    order_details
    JOIN
    pizzas ON order_details.pizza_id = pizzas.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 Rows:		
	name	price
▶	The Greek Pizza	35.95



4. 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   Filter F		
	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(order_details.quantity) AS quantity
FROM
    pizza_types
    JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
    JOIN
    order_details ON pizzas.pizza_id = order_details.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(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			Filter
	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);
```

Result Grid			Filter
	hour	order_count	
▶	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	



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 Rows: <input type="text"/>		
	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_pizzas_ordered_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_pizzas_ordered_per_day		
▶	138		

10. DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```
SELECT
    pizza_types.name,
    SUM(order_details.quantity * pizzas.price) AS rewanue
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 rewanue DESC
LIMIT 3;
```

Result Grid			Filter Rows:
	name	rewanue	
▶	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(order_details.quantity * pizzas.price) /  
(SELECT ROUND(SUM(order_details.quantity * pizzas.price),2) AS total_sales  
FROM order_details JOIN pizzas  
ON order_details.pizza_id = pizzas.pizza_id) * 100,2) AS revanue  
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;
```

Result Grid			Filter
	category	revanue	
▶	Classic	26.91	
	Veggie	23.68	
	Supreme	25.46	
	Chicken	23.96	

12. ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```
SELECT order_date,  
SUM(revanue) OVER(ORDER BY order_date) AS cum_revanue  
FROM (SELECT orders.order_date, SUM(order_details.quantity * pizzas.price)  
AS revanue 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: <input type="text"/>		
	order_date	cum_revanue
▶	2015-01-01	2713.850000000000004
	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	18888.85

13. DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA 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 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:	Export:	Wrap Cell C
	category	name	revenue			
▶	Chicken	The Thai Chicken Pizza	43434.25			
	Chicken	The Barbec The Thai Chicken Pizza	768			
	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	32265.700000000065			
	Veggie	The Mexicana Pizza	26780.75			
	Veggie	The Five Cheese Pizza	26066.5			



pizza sales



CONCLUSION

The Pizza Sales Database Analysis project demonstrates how SQL can transform raw data into valuable business insights — from customer preferences and top-selling pizzas to peak order times and revenue drivers.

This data-centric approach enhances decision-making, supports stakeholders in boosting sales, improving customer satisfaction, and optimizing operations.

RECOMMENDATIONS



- Maintain higher stock of Large-size pizzas to match demand.
- Align staff schedules with peak order hours for smoother operations.
- Highlight and market best-selling pizzas to attract more customers.
- Encourage upselling of premium pizzas that generate higher revenue.
- Introduce special deals during lunch hours to capture office crowd.
- Promote Classic pizzas (highest revenue category).



THANK YOU

19,AUG 2025