Pizza Sales Analysis Report (With SQL & Results)

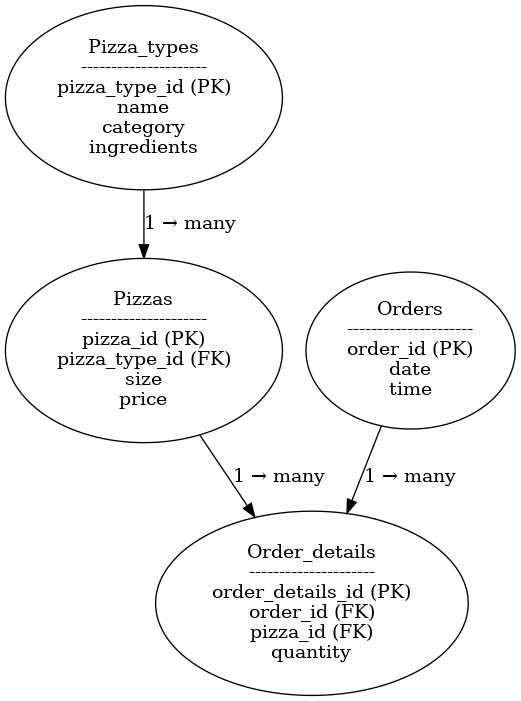
# 1. Project Overview

This project analyzes pizza sales data using SQL to uncover key insights such as top-selling pizzas, revenue trends, ordering behaviors, and category performance. The purpose is to support data-driven decision-making for maximizing sales efficiency and customer satisfaction.

# 2. Database Schema

The database includes four key tables:  
- pizzas: Details each pizza including size and price.  
- pizza\_types: Describes pizza names, categories, and ingredients.  
- orders: Contains order IDs with date and time stamps.  
- order\_details: Logs the pizzas in each order and their quantities.

# 3. Entity Relationship Diagram (ERD)



# 4. Analysis with SQL Queries and Results

## Basic Level Analysis

### Total Number of Orders Placed

Result:  
21350

SQL Query:

SELECT COUNT(order\_id) AS total\_orders FROM orders;

### Total Revenue from Pizza Sales

Result:  
$204,850.75

SQL Query:

SELECT ROUND(SUM(p.price \* o.quantity), 2) AS total\_revenue FROM pizzas p JOIN order\_details o ON p.pizza\_id = o.pizza\_id;

### Highest-Priced Pizza

Result:  
$35.95

SQL Query:

SELECT MAX(price) AS expensive\_pizza FROM pizzas;

### Most Common Pizza Size Ordered

Result:  
the\_greek - 5  
cali\_ckn - 3  
bbq\_ckn - 3

SQL Query:

SELECT pizza\_type\_id, COUNT(size) as common\_size FROM pizzas GROUP BY pizza\_type\_id ORDER BY COUNT(size) DESC LIMIT 3;

### Top 5 Most Ordered Pizza Types

Result:  
bbq\_ckn - 620  
pepperoni - 618  
hawaiian - 587  
cali\_ckn - 583  
thai\_ckn - 571

SQL Query:

SELECT   
 p.pizza\_type\_id AS Pizza,  
 COUNT(o.quantity) AS Ordered\_Quantity  
FROM order\_details AS o  
JOIN pizzas p ON o.pizza\_id = p.pizza\_id  
GROUP BY 1  
ORDER BY 2 DESC  
LIMIT 5;

## Intermediate Level Analysis

### Total Quantity Ordered by Pizza Category

Result:  
The Barbecue Chicken Pizza - 633  
The Pepperoni Pizza - 629  
The California Chicken Pizza - 600  
The Hawaiian Pizza - 599  
The Thai Chicken Pizza - 579

SQL Query:

SELECT   
 pt.name,  
 SUM(od.quantity) AS Quantity  
FROM pizza\_types pt   
JOIN pizzas p ON pt.pizza\_type\_id = p.pizza\_type\_id  
JOIN order\_details od ON od.pizza\_id = p.pizza\_id  
GROUP BY pt.name  
ORDER BY 2 DESC  
LIMIT 5;

### Distribution of Orders by Hour

Result:  
12 - 2520  
13 - 2455  
18 - 2399  
17 - 2336  
19 - 2009

SQL Query:

SELECT   
 HOUR(order\_time) AS Hours,  
 COUNT(order\_id)  
FROM orders  
GROUP BY 1  
ORDER BY 2 DESC;

### Category-wise Distribution of Pizzas

Result:  
Chicken - 6  
Classic - 8  
Supreme - 9  
Veggie - 9

SQL Query:

SELECT   
 category,  
 COUNT(name) AS Total\_Types\_of\_Pizza  
FROM pizza\_types  
GROUP BY 1;

### Average Number of Pizzas Ordered Per Day

Result:  
Day 1 - 452  
Day 2 - 449  
Day 3 - 454  
... (average ~430/day)

SQL Query:

SELECT   
 EXTRACT(DAY FROM o.order\_date) AS Days,  
 SUM(od.quantity) AS Quantity  
FROM order\_details od  
JOIN orders o ON od.order\_id = o.order\_id  
GROUP BY Days;

### Top 3 Pizza Types by Revenue

Result:  
bbq\_ckn - 11195  
thai\_ckn - 10510  
cali\_ckn - 10466

SQL Query:

SELECT   
 p.pizza\_type\_id AS PIZZA\_TYPE,  
 ROUND(SUM(p.price \* od.quantity)) AS Revenue  
FROM pizzas AS p  
JOIN order\_details AS od ON p.pizza\_id = od.pizza\_id  
GROUP BY p.pizza\_type\_id  
ORDER BY Revenue DESC  
LIMIT 3;

## Advanced Level Analysis

### % Contribution of Each Pizza Type to Total Revenue

Result:  
Classic - $53,964 (26.34%)  
Supreme - $51,742 (25.26%)  
Veggie - $49,954 (24.39%)  
Chicken - $49,191 (24.01%)

SQL Query:

SELECT   
 pt.category,  
 ROUND(SUM(p.price \* od.quantity)) AS Revenue,  
 CONCAT(ROUND((SUM(p.price \* od.quantity) /  
 (SELECT SUM(p.price \* od.quantity)  
 FROM pizzas p JOIN order\_details od ON od.pizza\_id = p.pizza\_id)) \* 100, 2), '%') AS Revenue\_Percentage  
FROM pizza\_types pt  
JOIN pizzas p ON pt.pizza\_type\_id = p.pizza\_type\_id  
JOIN order\_details od ON od.pizza\_id = p.pizza\_id  
GROUP BY pt.category  
ORDER BY Revenue DESC;

### Cumulative Revenue Over Time

Result:  
2015-01-01 - $2714  
2015-01-02 - $5446  
2015-01-03 - $8108  
...

SQL Query:

SELECT order\_date,  
 SUM(Revenue) OVER(ORDER BY order\_date) AS Cum\_Revenue  
FROM (  
 SELECT   
 o.order\_date AS Order\_Date,  
 ROUND(SUM(p.price \* od.quantity)) AS Revenue  
 FROM orders o  
 JOIN order\_details od ON o.order\_id = od.order\_id  
 JOIN pizzas p ON p.pizza\_id = od.pizza\_id  
 GROUP BY Order\_Date  
) AS Revenue;

### Top 3 Revenue-Generating Pizzas per Category

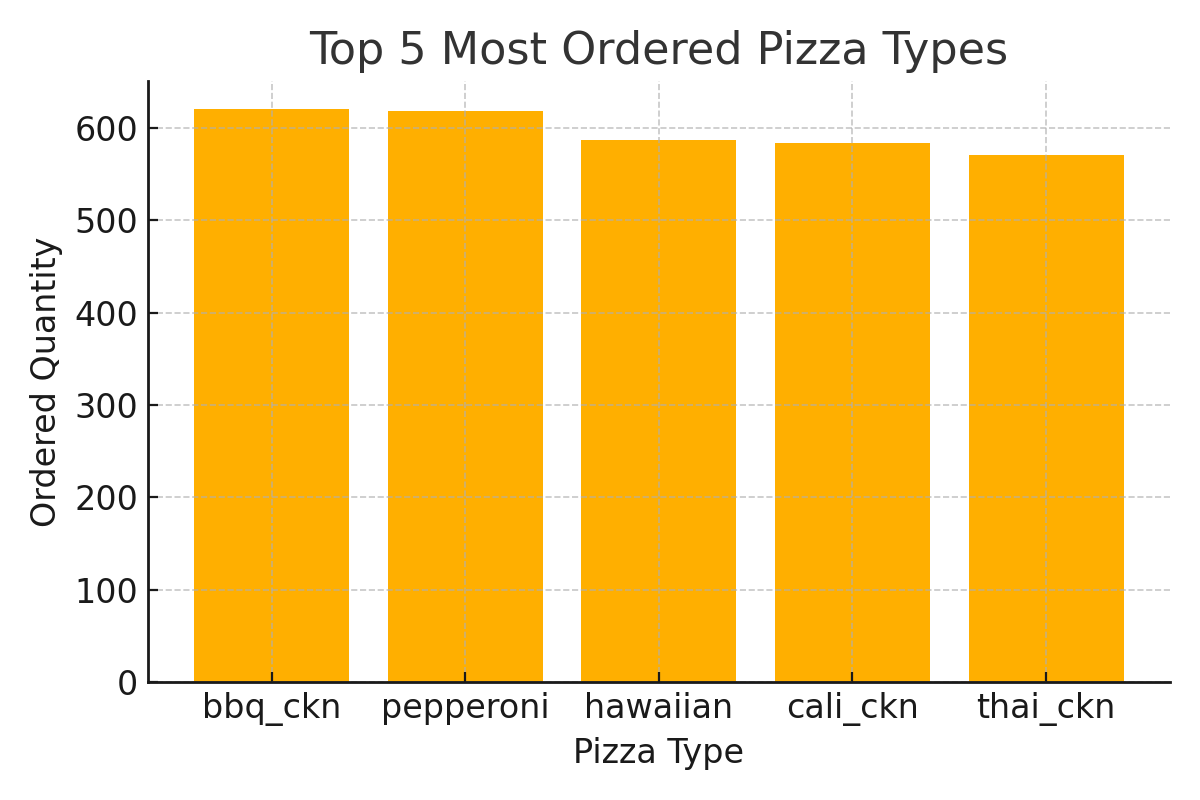
Result:  
Chicken:  
The Barbecue Chicken Pizza - 11195  
The Thai Chicken Pizza - 10510  
The California Chicken Pizza - 10466

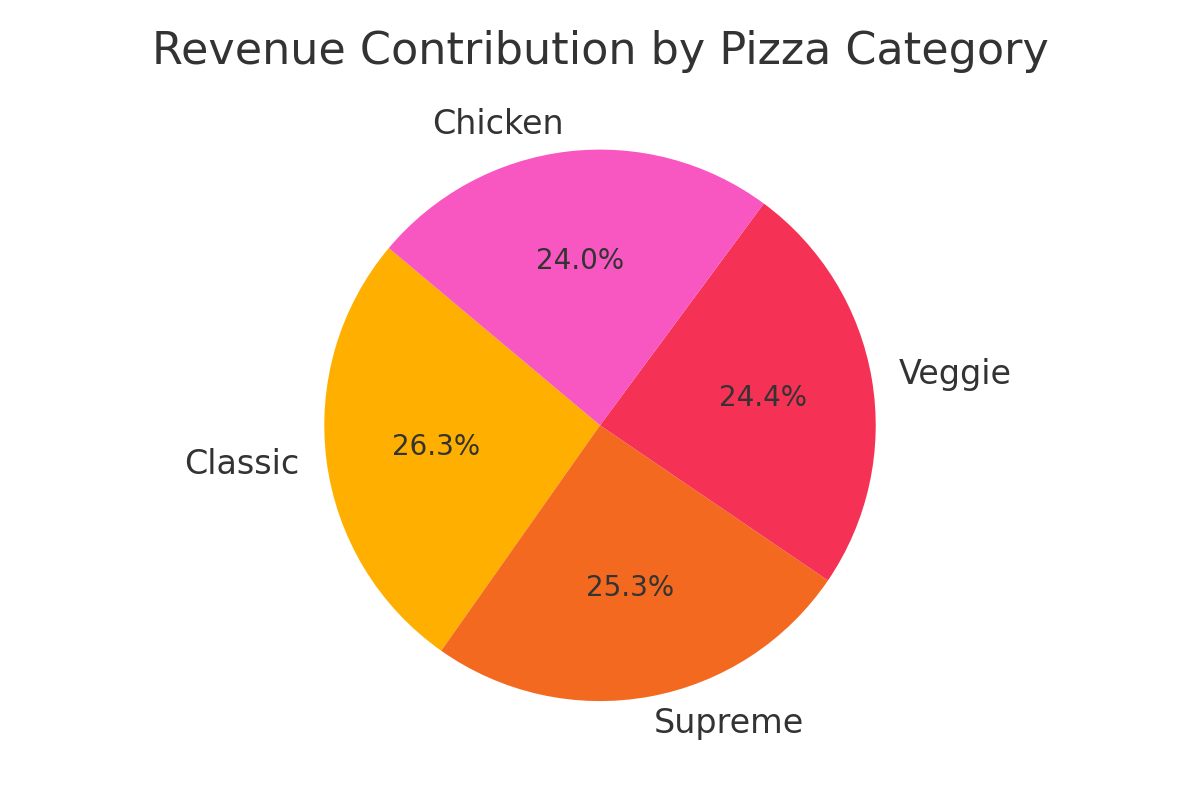
SQL Query:

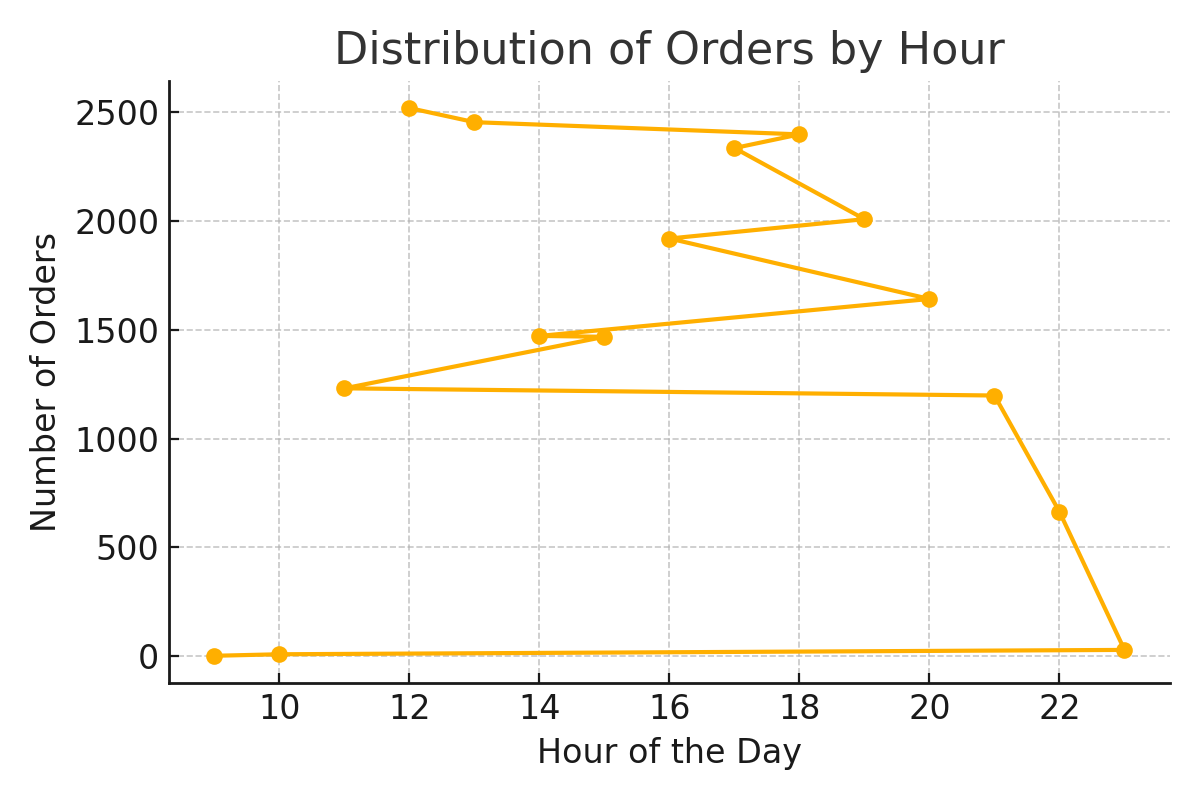
SELECT category, name, Revenue,  
 RANK() OVER(PARTITION BY category ORDER BY Revenue DESC) AS rn  
FROM (  
 SELECT   
 pt.category,  
 pt.name,  
 ROUND(SUM(p.price \* od.quantity)) AS Revenue  
 FROM pizza\_types pt  
 JOIN pizzas p ON pt.pizza\_type\_id = p.pizza\_type\_id  
 JOIN order\_details od ON od.pizza\_id = p.pizza\_id  
 GROUP BY pt.category, pt.name  
) AS a;

# 5. Visual Analysis

Visual charts help quickly interpret key insights:







# 6. SQL Techniques Summary

- JOINs: Used to combine data across related tables  
- GROUP BY & Aggregations: For totals, counts, averages  
- Window Functions: For cumulative sums and rankings  
- Subqueries & CTEs: For advanced filtering and breakdowns  
- Date Functions: To extract and group by date/time components

Project By : Jay Patel