**Pizza Hut Sales Analysis – SQL Project**

# Pizza Hut Sales Analysis

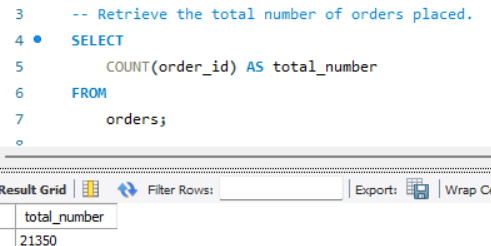
**Project Overview:**

Analysed Pizza Hut sales data using MySQL to uncover trends in orders, revenue, and popular pizzas. Includes category-wise insights, peak order hours, and revenue analysis to support data-driven business decisions.

## Q1: Retrieve the total number of orders placed

**SQL Query:**

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

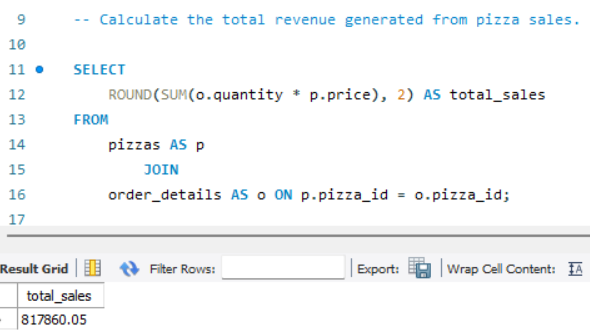


**Insight:** The total number of orders reflects overall customer engagement. High order volume suggests strong demand and potential for upselling or promotions.

## Q2: Calculate the total revenue generated from pizza sales

**SQL Query:**

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

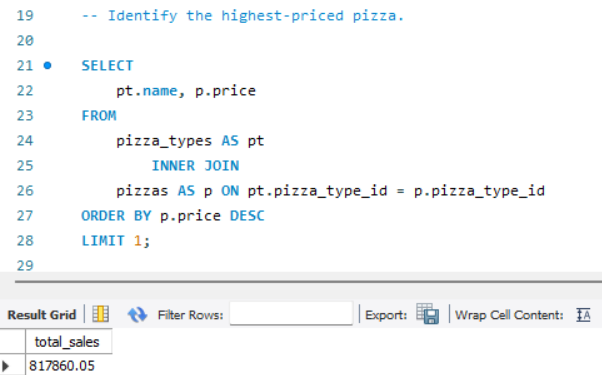


**Insight:** Total revenue indicates store financial performance. This metric serves as a benchmark for growth tracking and profitability evaluation.

## Q3: Identify the highest-priced pizza

**SQL Query:**

SELECT pt.name, p.price  
FROM pizza\_types AS pt  
INNER JOIN pizzas AS p ON pt.pizza\_type\_id = p.pizza\_type\_id  
ORDER BY p.price DESC  
LIMIT 1;

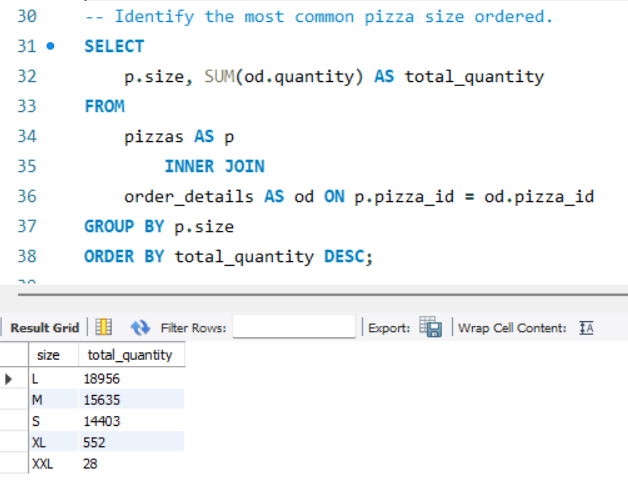


**Insight:** The highest-priced pizza represents the premium segment. Insights here guide marketing, pricing strategies, and promotions targeting high-value customers.

## Q4: Identify the most common pizza size ordered

**SQL Query:**

SELECT p.size, SUM(od.quantity) AS total\_quantity  
FROM pizzas AS p  
INNER JOIN order\_details AS od ON p.pizza\_id = od.pizza\_id  
GROUP BY p.size  
ORDER BY total\_quantity DESC;

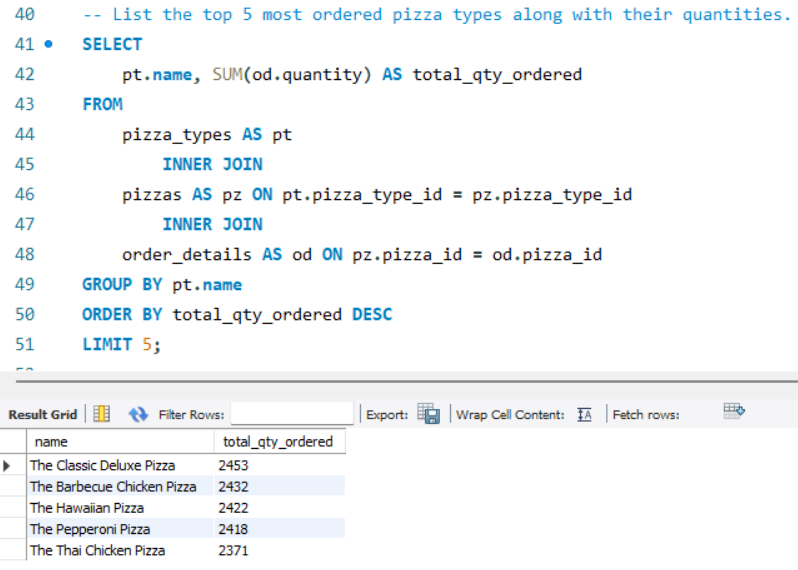


**Insight:** Shows customer preference for pizza size. Dominance of a particular size, e.g., “Large”, informs inventory planning and marketing focus.

## Q5: List the top 5 most ordered pizza types along with their quantities

**SQL Query:**

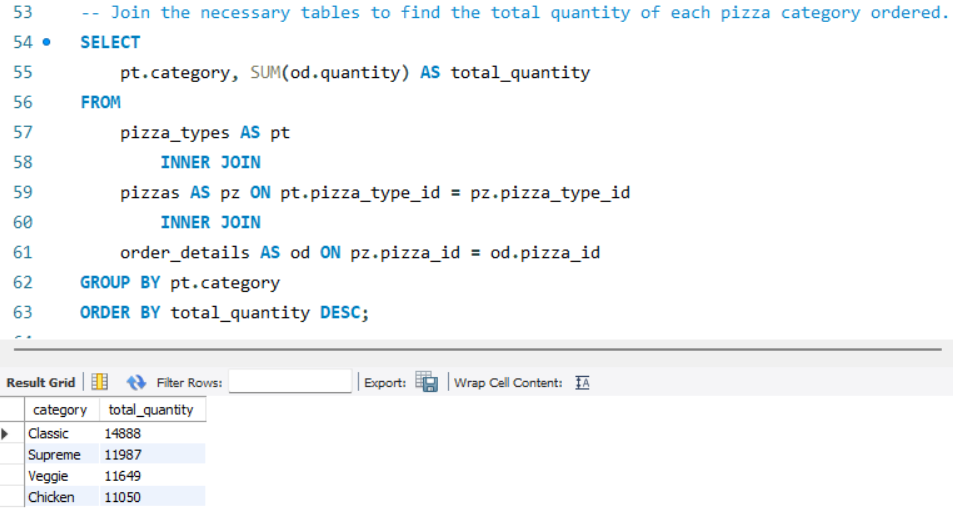
SELECT pt.name, SUM(od.quantity) AS total\_qty\_ordered  
FROM pizza\_types AS pt  
INNER JOIN pizzas AS pz ON pt.pizza\_type\_id = pz.pizza\_type\_id  
INNER JOIN order\_details AS od ON pz.pizza\_id = od.pizza\_id  
GROUP BY pt.name  
ORDER BY total\_qty\_ordered DESC  
LIMIT 5;

 **Insight:** Identifies best-selling pizzas driving sales. Key for stock planning, promotional campaigns, and menu optimization.

## Q6: Total quantity of each pizza category ordered

**SQL Query:**

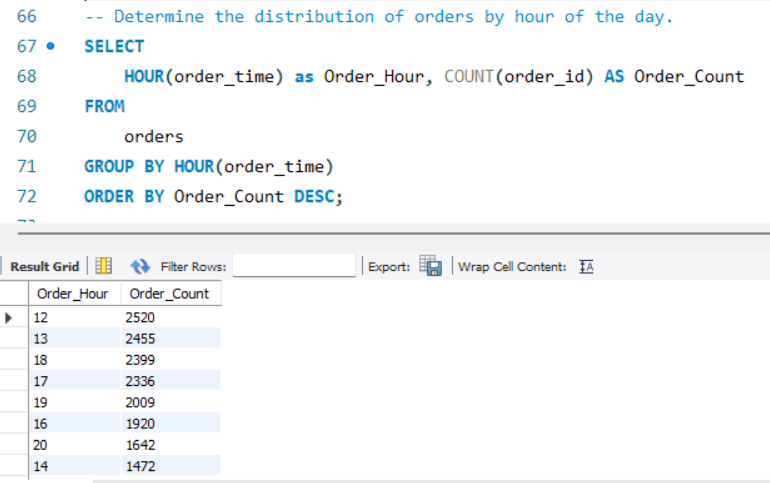
SELECT pt.category, SUM(od.quantity) AS total\_quantity  
FROM pizza\_types AS pt  
INNER JOIN pizzas AS pz ON pt.pizza\_type\_id = pz.pizza\_type\_id  
INNER JOIN order\_details AS od ON pz.pizza\_id = od.pizza\_id  
GROUP BY pt.category  
ORDER BY total\_quantity DESC;

 **Insight:** Category-wise orders highlight the popularity of each pizza category. This supports inventory allocation and category-focused marketing.

## Q7: Distribution of orders by hour of the day

**SQL Query:**

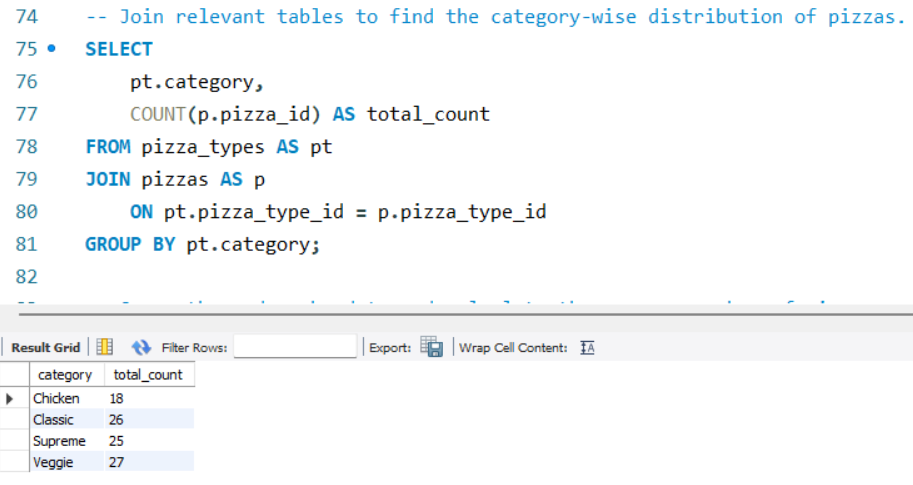
SELECT HOUR(order\_time) as Order\_Hour, COUNT(order\_id) AS Order\_Count  
FROM orders  
GROUP BY HOUR(order\_time)  
ORDER BY Order\_Count DESC;

 **Insight:** Peak ordering hours indicate when customers are most active. Useful for staffing, delivery scheduling, and time-based promotions.

## Q8: Category-wise distribution of pizzas

**SQL Query:**

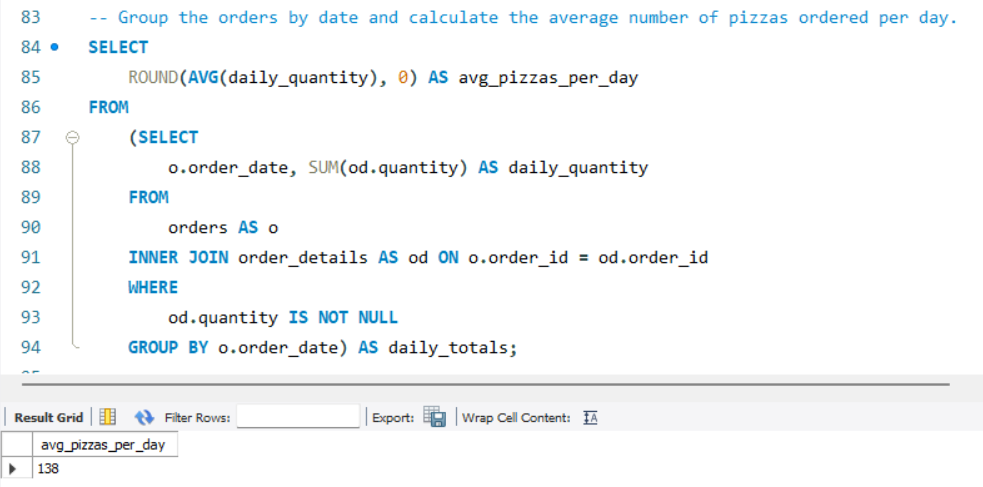
SELECT pt.category, COUNT(p.pizza\_id) AS total\_count  
FROM pizza\_types AS pt  
JOIN pizzas AS p ON pt.pizza\_type\_id = p.pizza\_type\_id  
GROUP BY pt.category;

 **Insight:** Shows the total number of pizzas in each category, reflecting menu composition and variety.

## Q9: Average number of pizzas ordered per day

**SQL Query:**

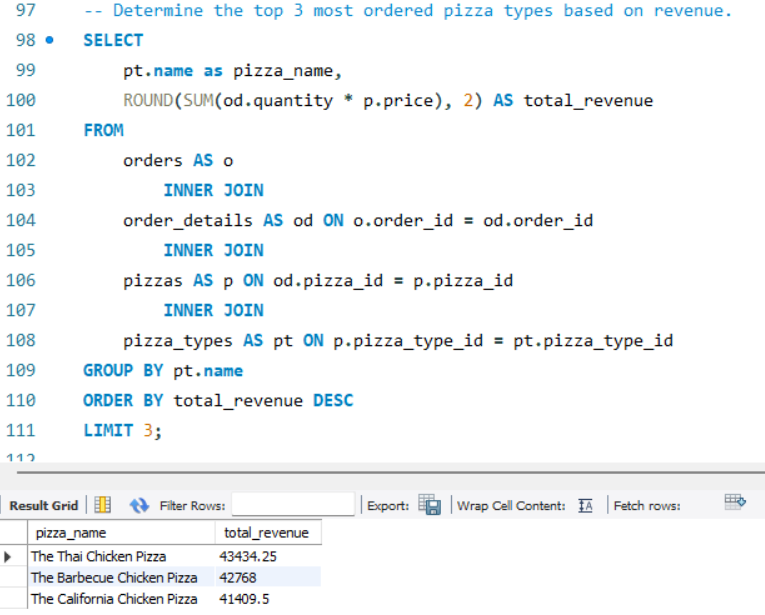
SELECT ROUND(AVG(daily\_quantity), 0) AS avg\_pizzas\_per\_day  
FROM (  
 SELECT o.order\_date, SUM(od.quantity) AS daily\_quantity  
 FROM orders AS o  
 INNER JOIN order\_details AS od ON o.order\_id = od.order\_id  
 WHERE od.quantity IS NOT NULL  
 GROUP BY o.order\_date  
) AS daily\_totals;

 **Insight:** Provides a baseline daily demand for inventory and forecasting. Useful for detecting seasonal trends.

## Q10: Top 3 most ordered pizza types based on revenue

**SQL Query:**

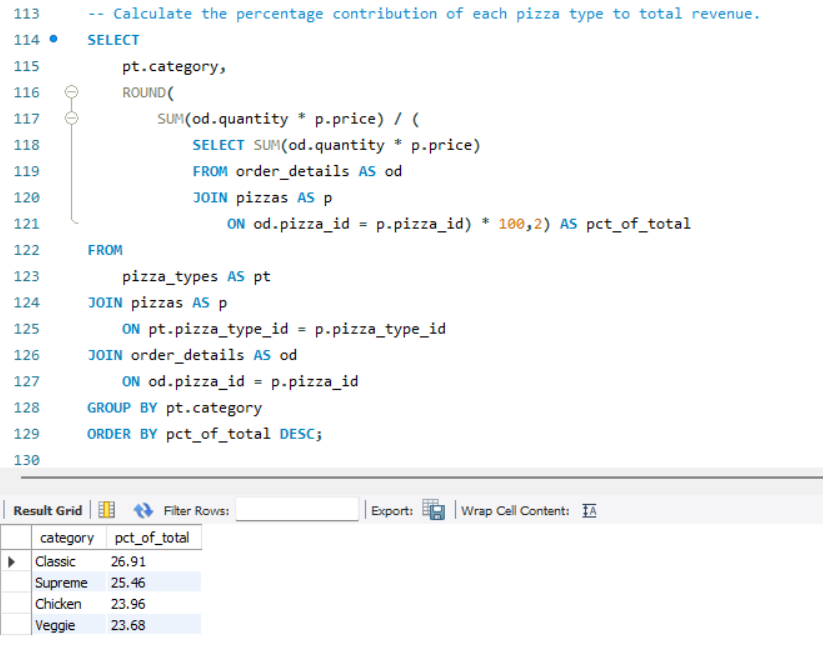
SELECT pt.name AS pizza\_name, ROUND(SUM(od.quantity \* p.price), 2) AS total\_revenue  
FROM orders AS o  
INNER JOIN order\_details AS od ON o.order\_id = od.order\_id  
INNER JOIN pizzas AS p ON od.pizza\_id = p.pizza\_id  
INNER JOIN pizza\_types AS pt ON p.pizza\_type\_id = pt.pizza\_type\_id  
GROUP BY pt.name  
ORDER BY total\_revenue DESC  
LIMIT 3;

 **Insight:** Identifies revenue-driving pizzas. These items are crucial for profitability and marketing focus.

## Q11: Percentage contribution of each pizza type to total revenue

**SQL Query:**

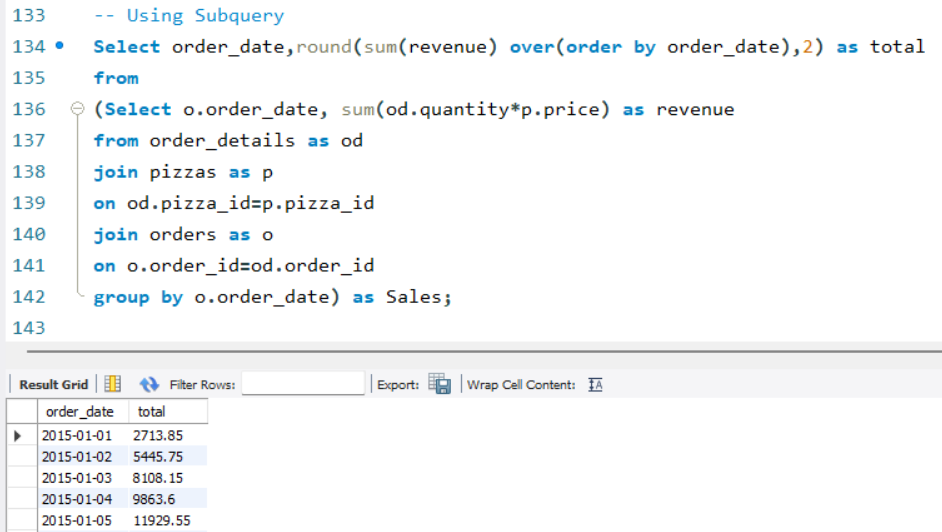
SELECT pt.category, ROUND(  
 SUM(od.quantity \* p.price) / (  
 SELECT SUM(od.quantity \* p.price)  
 FROM order\_details AS od  
 JOIN pizzas AS p ON od.pizza\_id = p.pizza\_id  
 ) \* 100, 2) AS pct\_of\_total  
FROM pizza\_types AS pt  
JOIN pizzas AS p ON pt.pizza\_type\_id = p.pizza\_type\_id 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

**Insight:** Shows revenue concentration by category. High-percentage categories can be prioritized for promotions and inventory management.

## Q12: Analyze cumulative revenue over time (Subquery)

**SQL Query:**

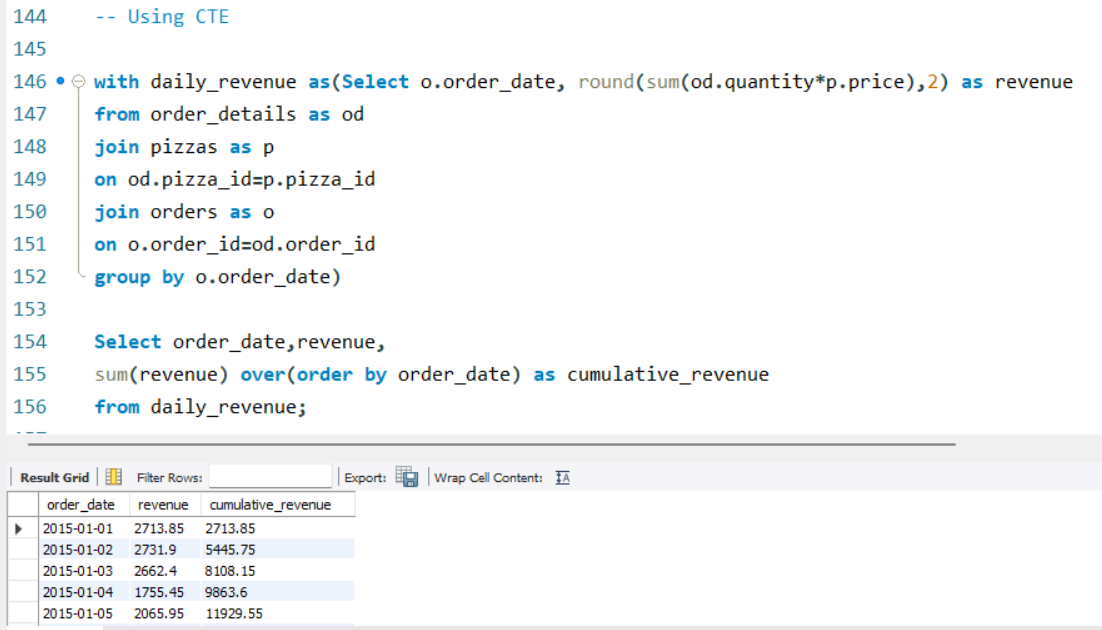
SELECT order\_date, SUM(revenue) OVER(ORDER BY order\_date) AS total  
FROM (  
 SELECT o.order\_date, SUM(od.quantity \* p.price) AS revenue  
 FROM order\_details AS od  
 JOIN pizzas AS p ON od.pizza\_id = p.pizza\_id  
 JOIN orders AS o ON o.order\_id = od.order\_id  
 GROUP BY o.order\_date  
) AS Sales;

 **Insight:** Tracks growth of revenue cumulatively. Helps monitor trends and measure the impact of campaigns or promotions over time.

## Q13: Analyze cumulative revenue over time (CTE)

**SQL Query:**

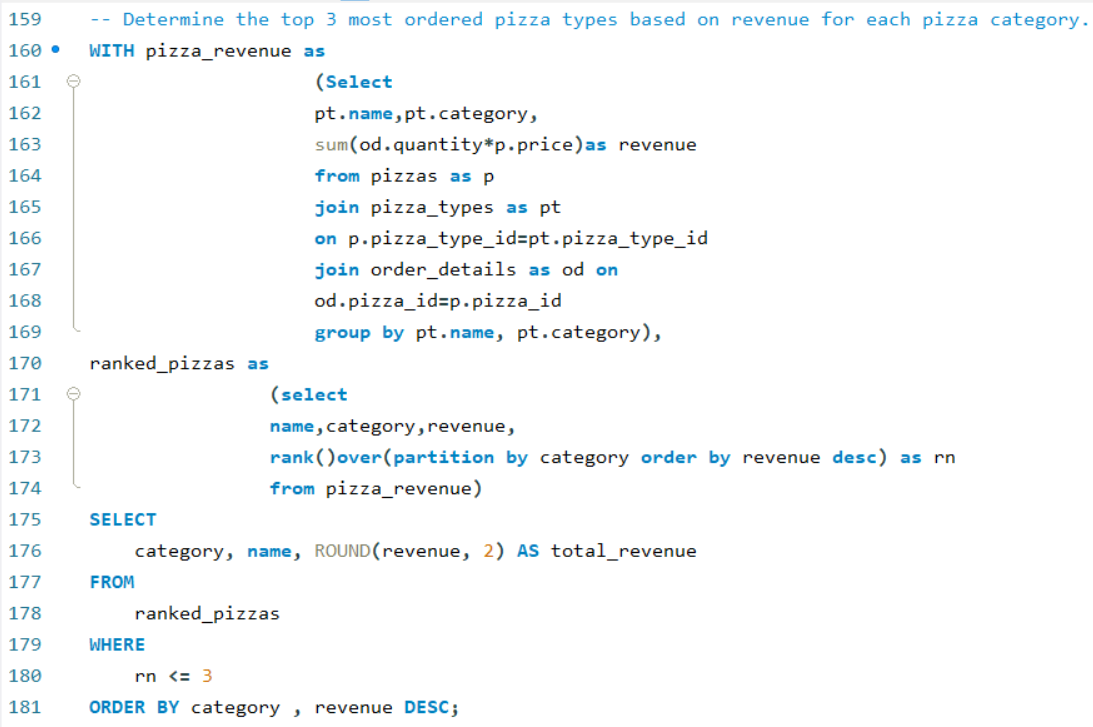
WITH daily\_revenue AS (  
 SELECT o.order\_date, ROUND(SUM(od.quantity \* p.price), 2) AS revenue  
 FROM order\_details AS od  
 JOIN pizzas AS p ON od.pizza\_id = p.pizza\_id  
 JOIN orders AS o ON o.order\_id = od.order\_id  
 GROUP BY o.order\_date  
)  
SELECT order\_date, revenue, SUM(revenue) OVER(ORDER BY order\_date) AS cumulative\_revenue  
FROM daily\_revenue;

**Insight:** Same as above, but modular using CTEs. Cleaner structure for reporting cumulative revenue trends.

## Q14: Top 3 most ordered pizza types based on revenue for each pizza category

**SQL Query:**

WITH pizza\_revenue AS (  
 SELECT pt.name, pt.category, SUM(od.quantity \* p.price) AS revenue  
 FROM pizzas AS p  
 JOIN pizza\_types AS pt ON p.pizza\_type\_id = pt.pizza\_type\_id  
 JOIN order\_details AS od ON od.pizza\_id = p.pizza\_id  
 GROUP BY pt.name, pt.category  
),  
ranked\_pizzas AS (  
 SELECT name, category, revenue,  
 RANK() OVER(PARTITION BY category ORDER BY revenue DESC) AS rn  
 FROM pizza\_revenue  
)  
SELECT category, name, ROUND(revenue, 2) AS total\_revenue  
FROM ranked\_pizzas  
WHERE rn <= 3  
ORDER BY category, revenue DESC;

 **Insight:** Provides category-specific top performers. Useful for targeted marketing and inventory optimization per pizza category.