Pizza Hut Sales Analysis

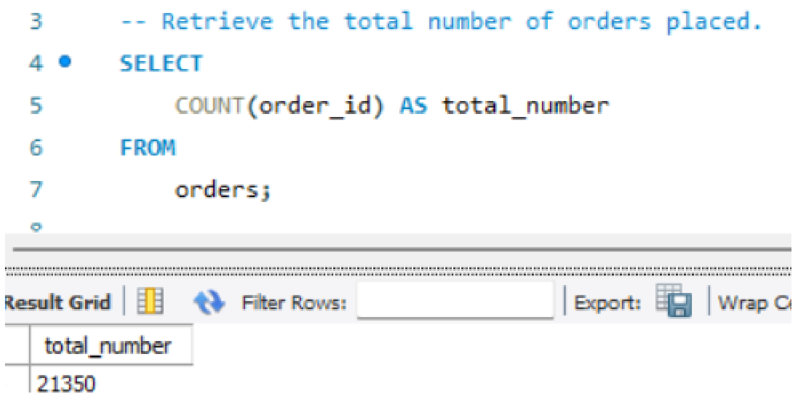
Project Overview:

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;
```



```
-- Retrieve the total number of orders placed.
SELECT
    COUNT(order_id) AS total_number
FROM
    orders;
```

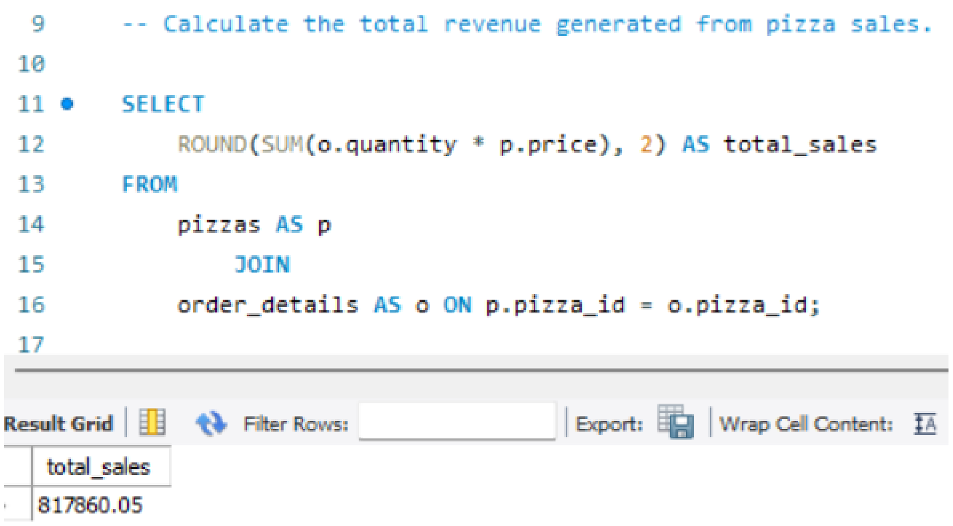
```
total_number
21350
```

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;
```



```
-- Calculate the total revenue generated from pizza sales.

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;
```

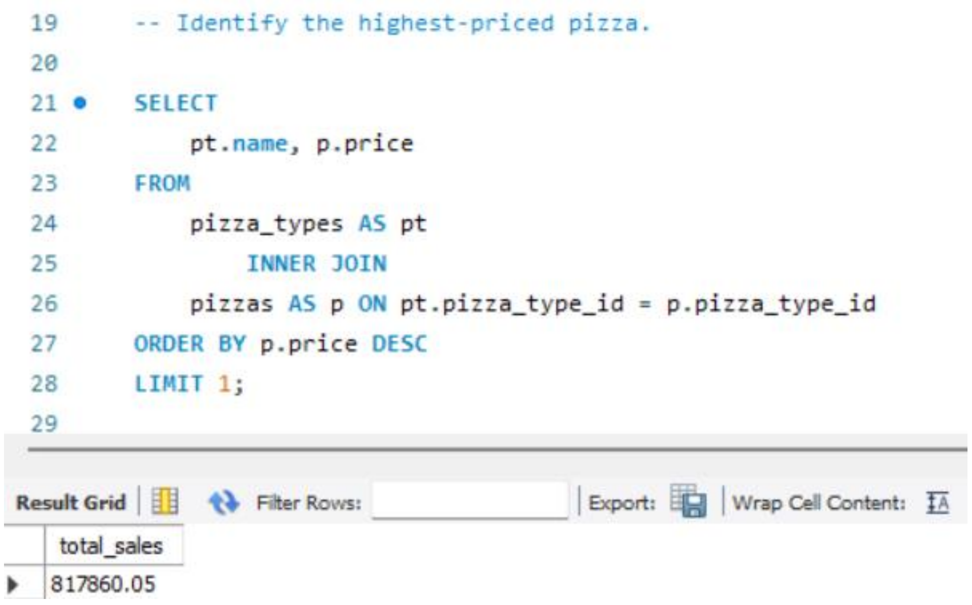
```
total_sales
817860.05
```

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;
```



```
-- Identify the highest-priced pizza.

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;
```

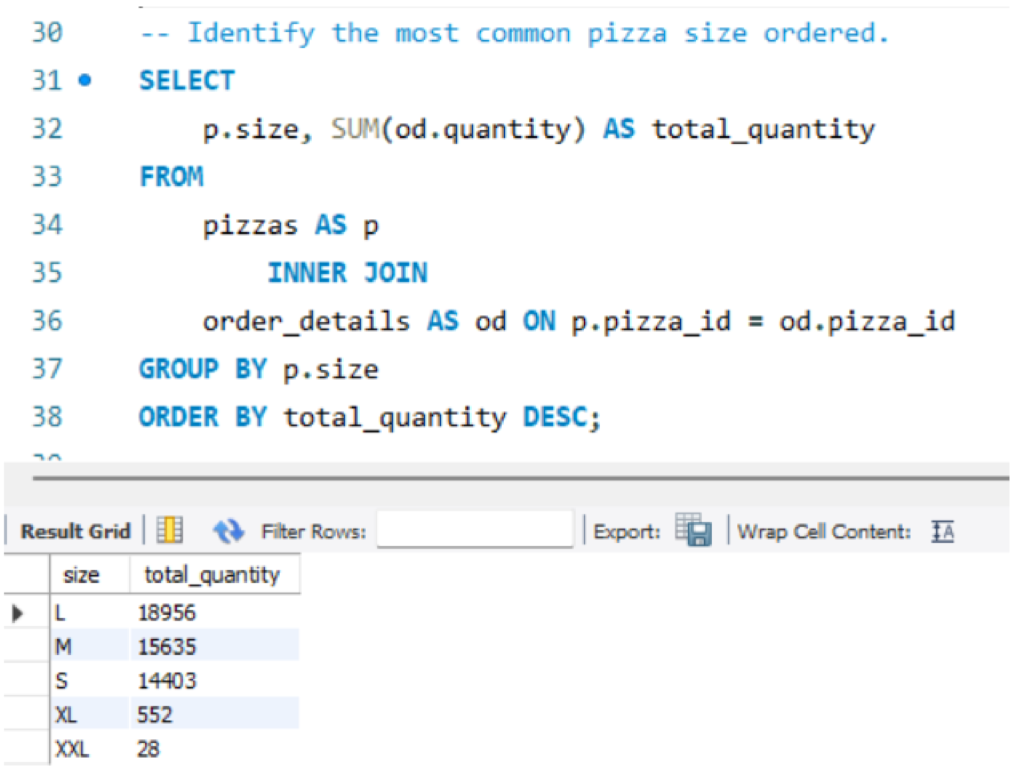
```
total_sales
817860.05
```

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;
```



```
-- Identify the most common pizza size ordered.
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;
```

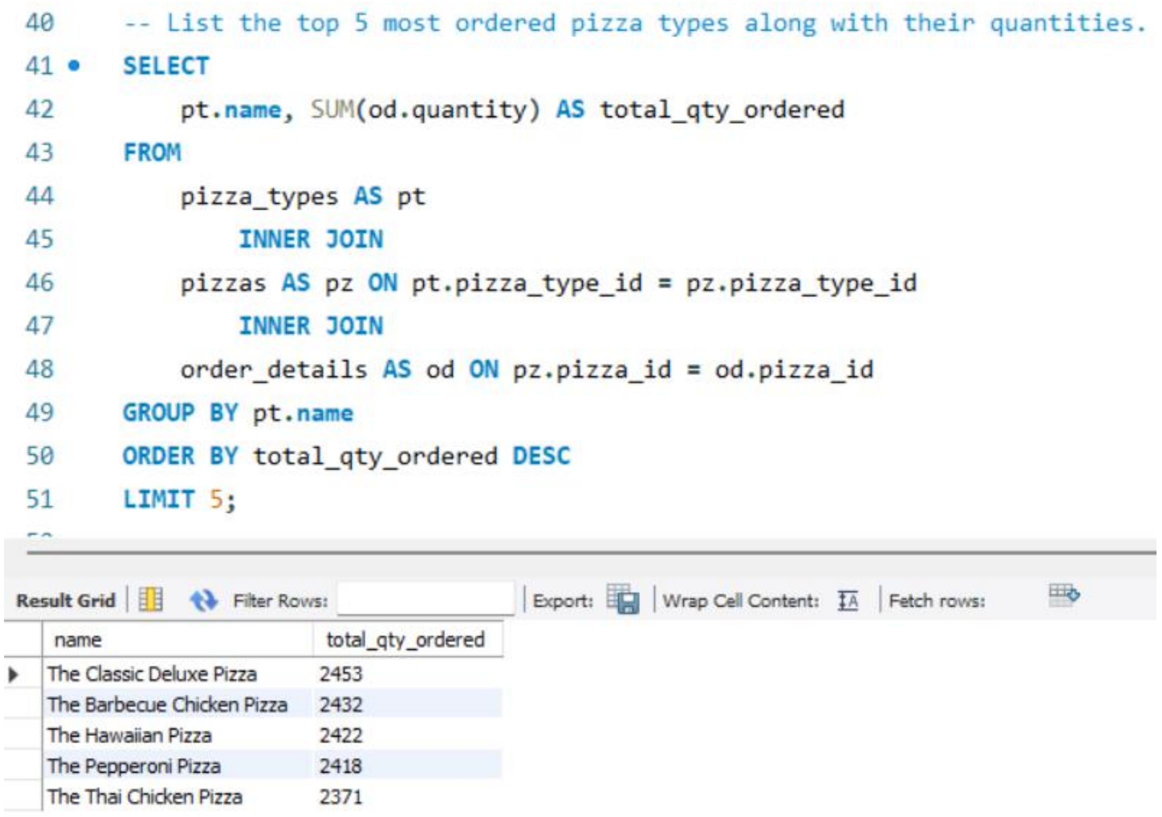
size	total_quantity
L	18956
M	15635
S	14403
XL	552
XXL	28

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;
```



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

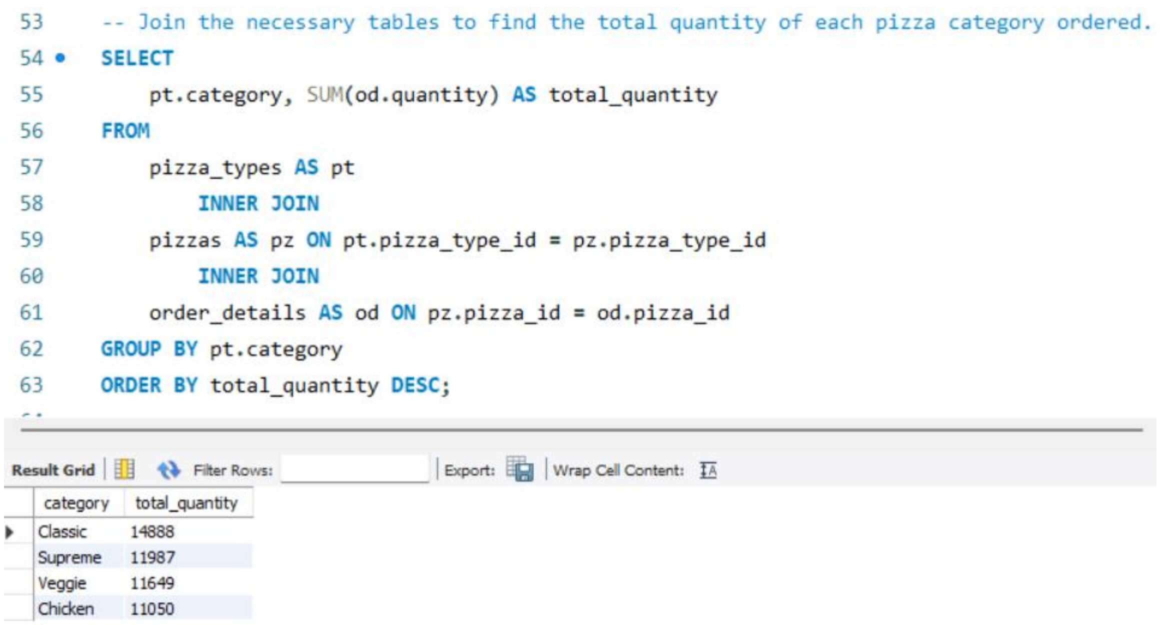
name	total_qty_ordered
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

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;
```


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

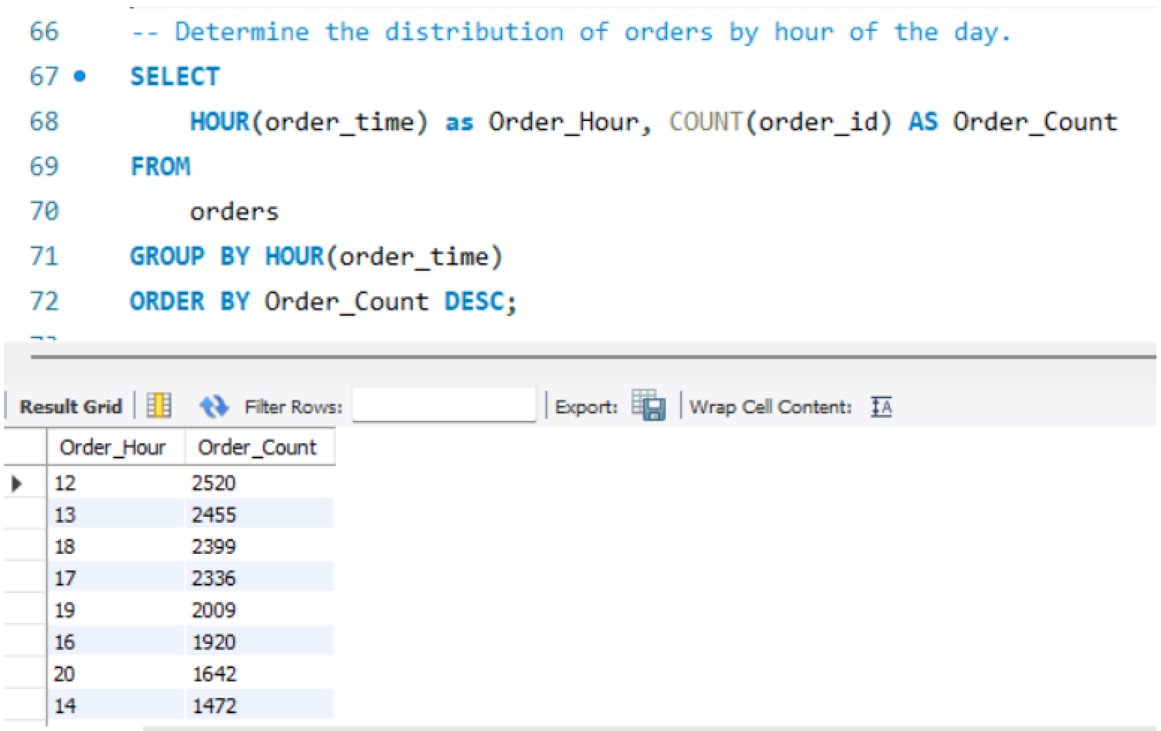
category	total_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

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;
```



```
-- Determine the distribution of orders by hour of the day.
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;
```

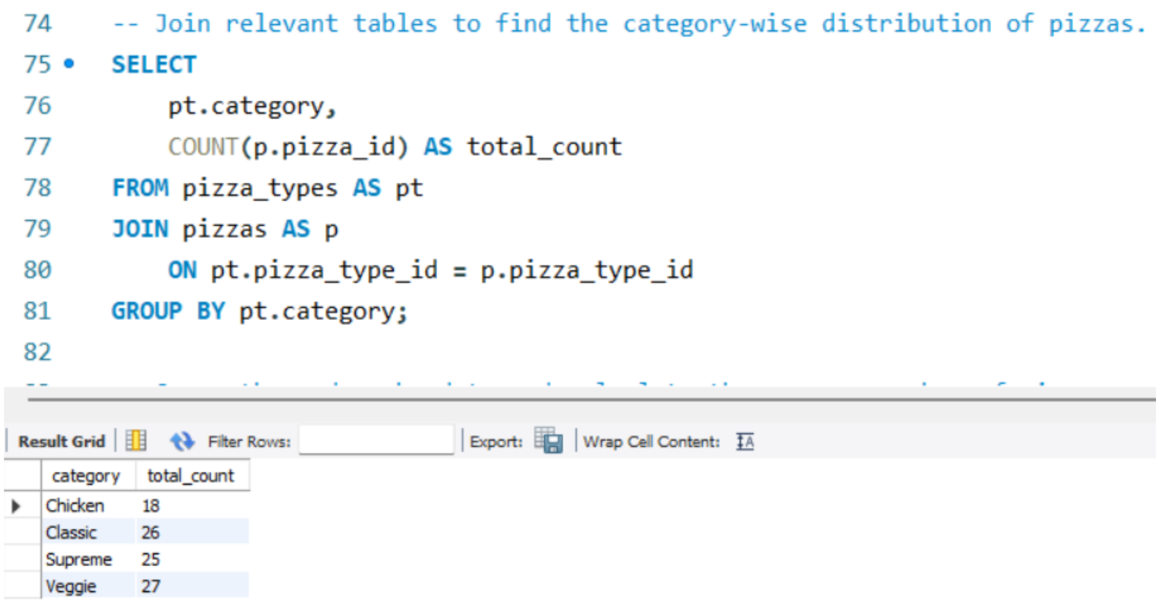
Order_Hour	Order_Count
12	2520
13	2455
18	2399
17	2336
19	2009
16	1920
20	1642
14	1472

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;
```



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

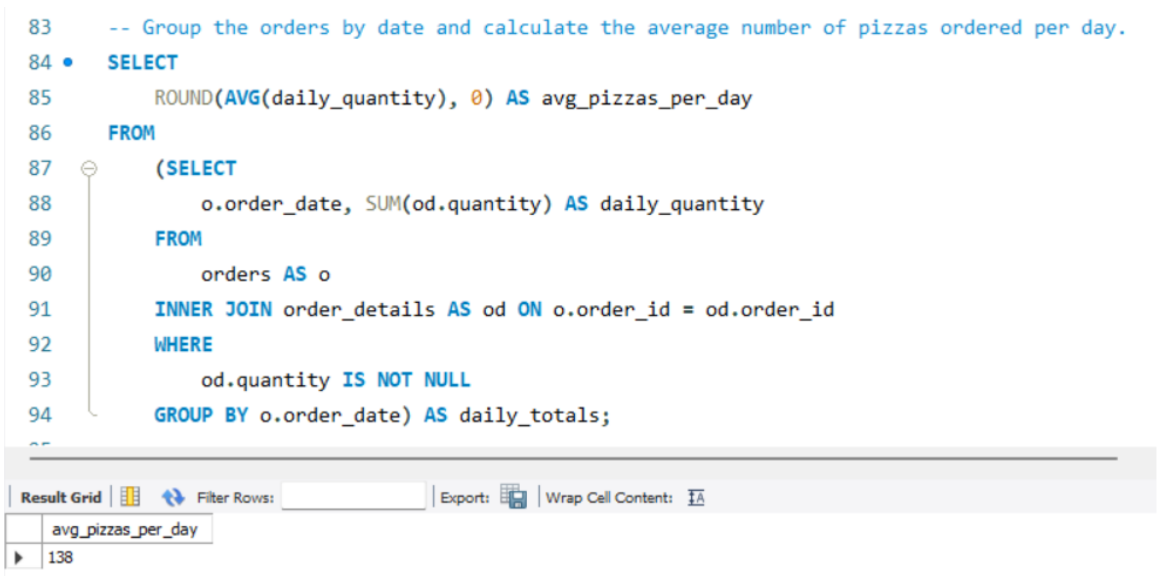
category	total_count
Chicken	18
Classic	26
Supreme	25
Veggie	27

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;
```



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

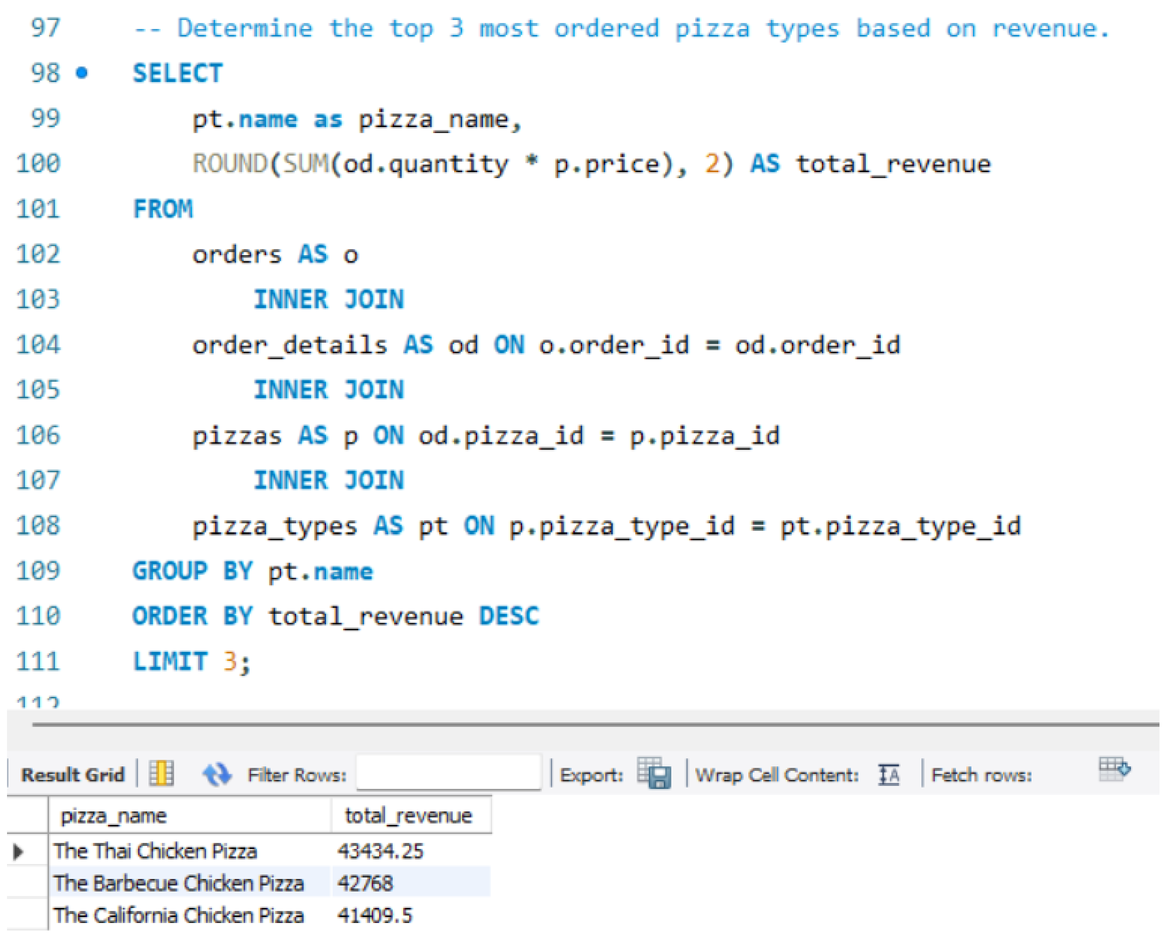
avg_pizzas_per_day
138

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_r
evenue
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;
```



```sql
-- Determine the top 3 most ordered pizza types based on revenue.
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;
```

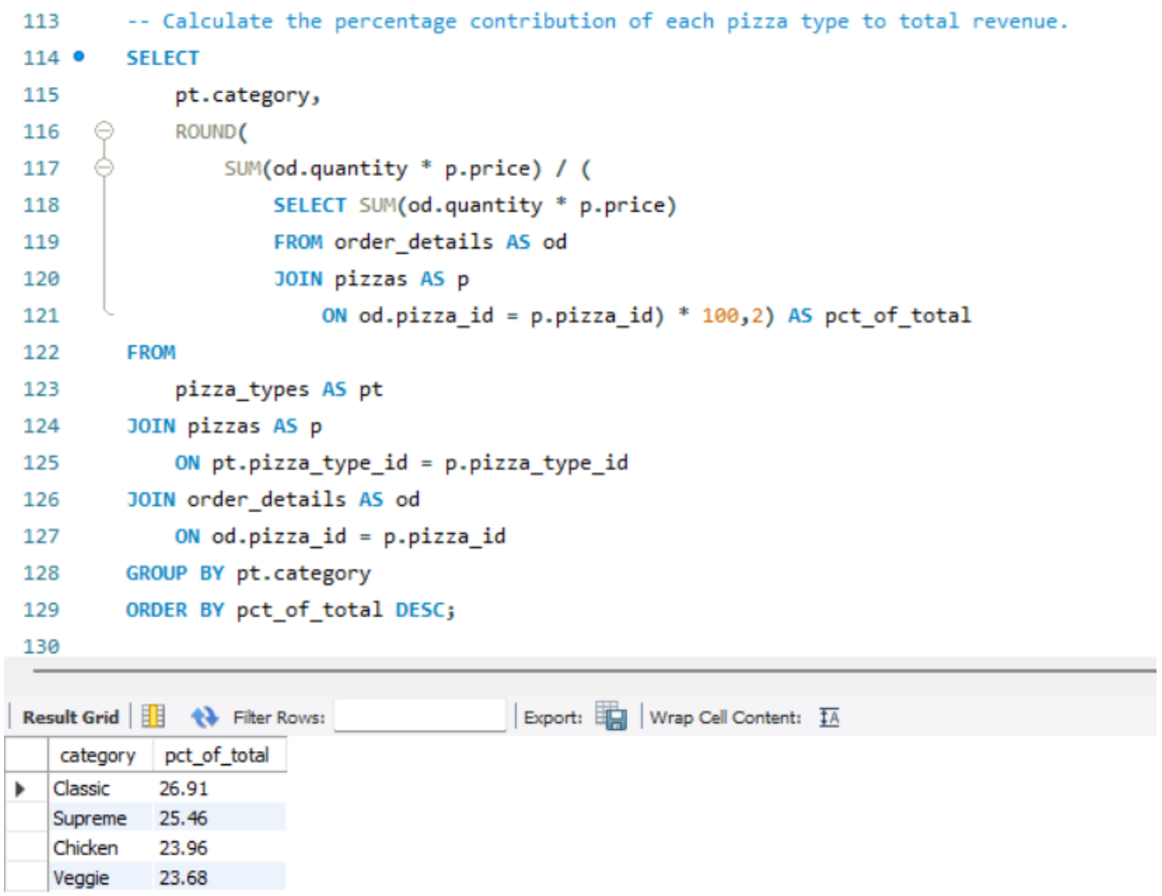
pizza_name	total_revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

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:

```sql
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;
```



```
-- Calculate the percentage contribution of each pizza type to total revenue.
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;
```

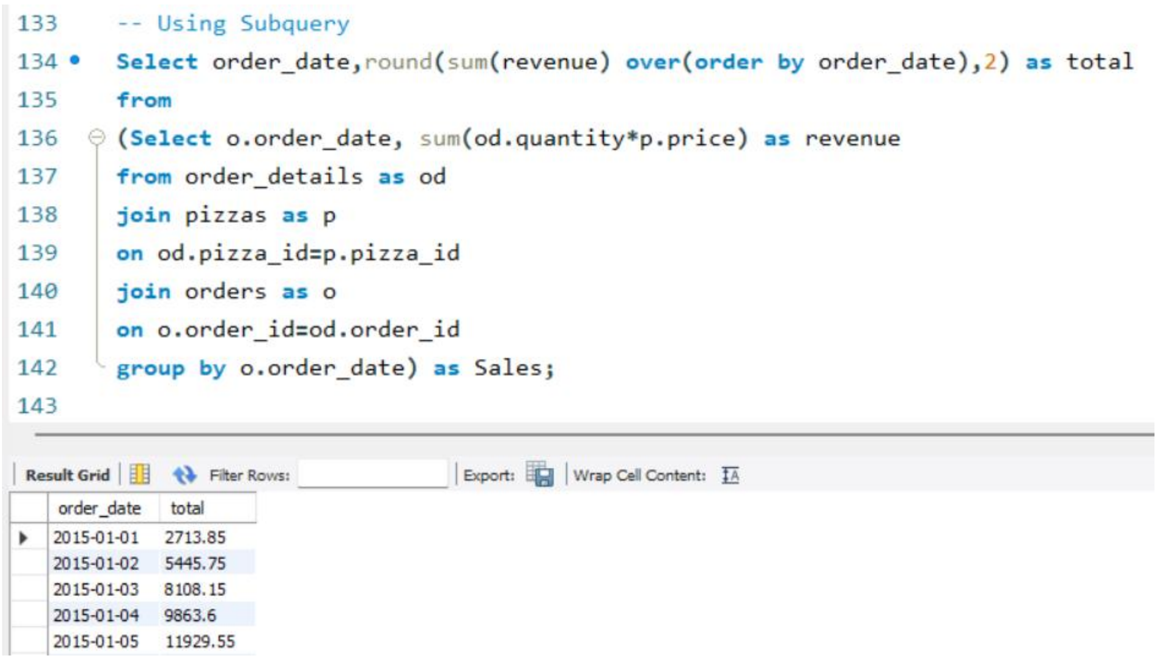
category	pct_of_total
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

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;
```



```
-- Using Subquery
Select order_date,round(sum(revenue) over(order by order_date),2) 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;
```

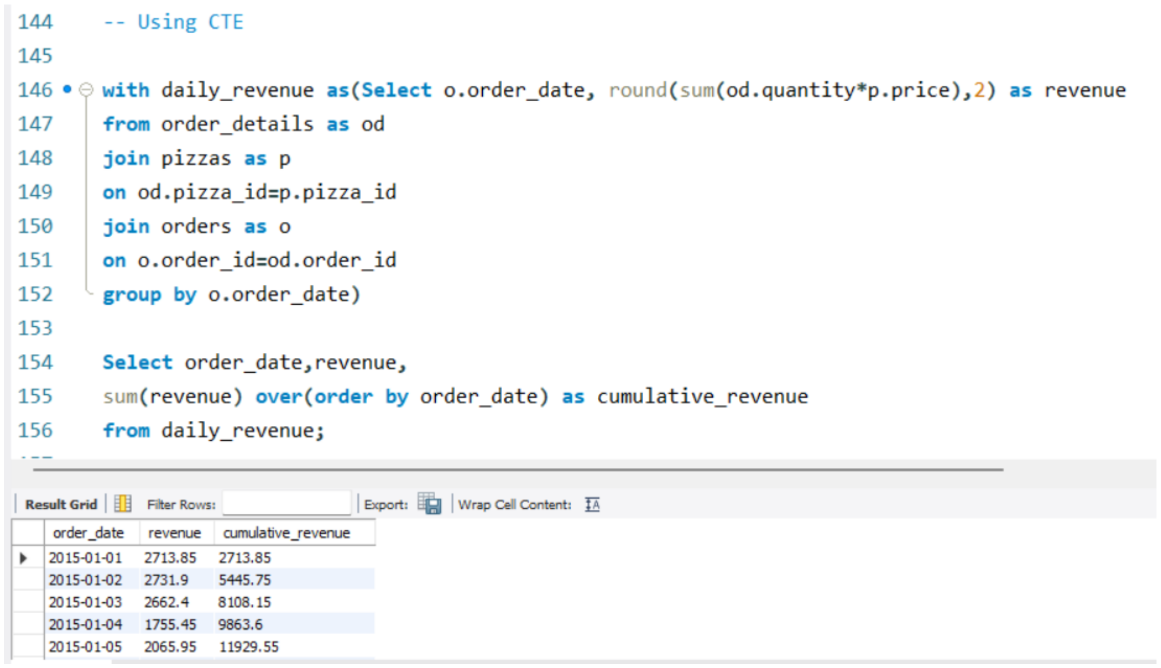
order_date	total
2015-01-01	2713.85
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55

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 cumulat
ive_revenue
FROM daily_revenue;
```



```
-- Using CTE

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;
```

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

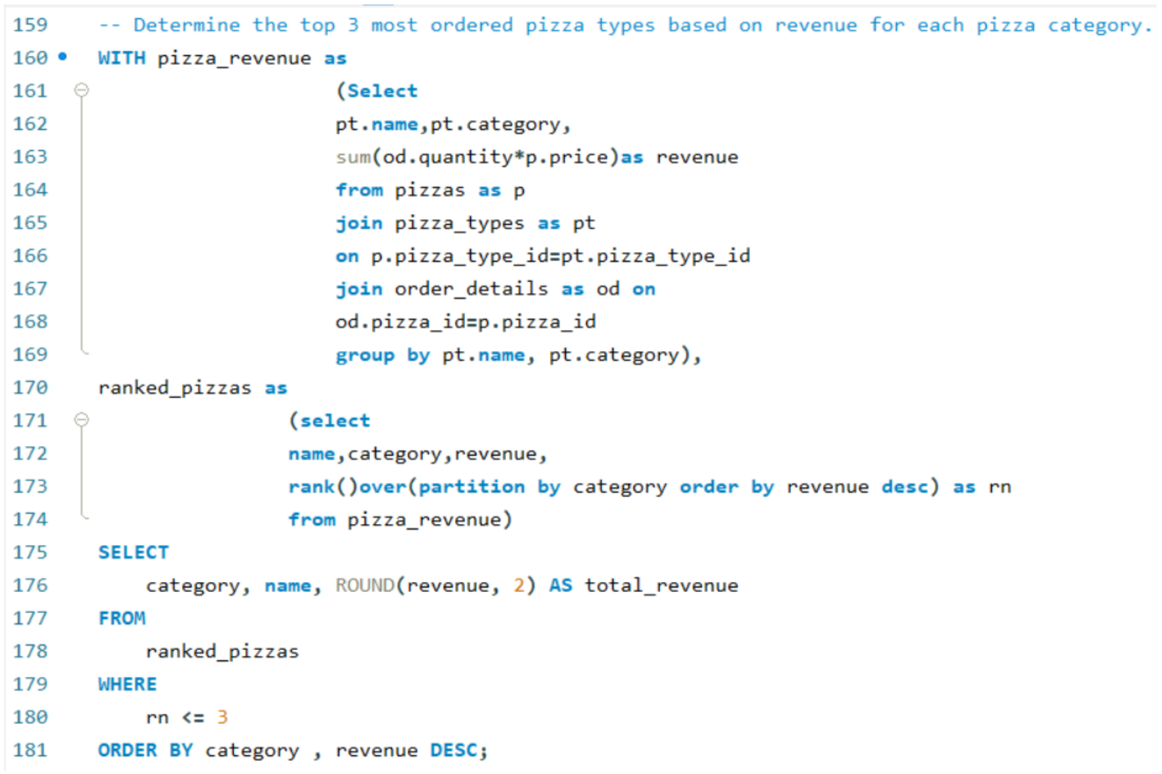
order_date	revenue	cumulative_revenue
2015-01-01	2713.85	2713.85
2015-01-02	2731.9	5445.75
2015-01-03	2662.4	8108.15
2015-01-04	1755.45	9863.6
2015-01-05	2065.95	11929.55

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;
```



```sql
-- Determine the top 3 most ordered pizza types based on revenue for each pizza category.
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.