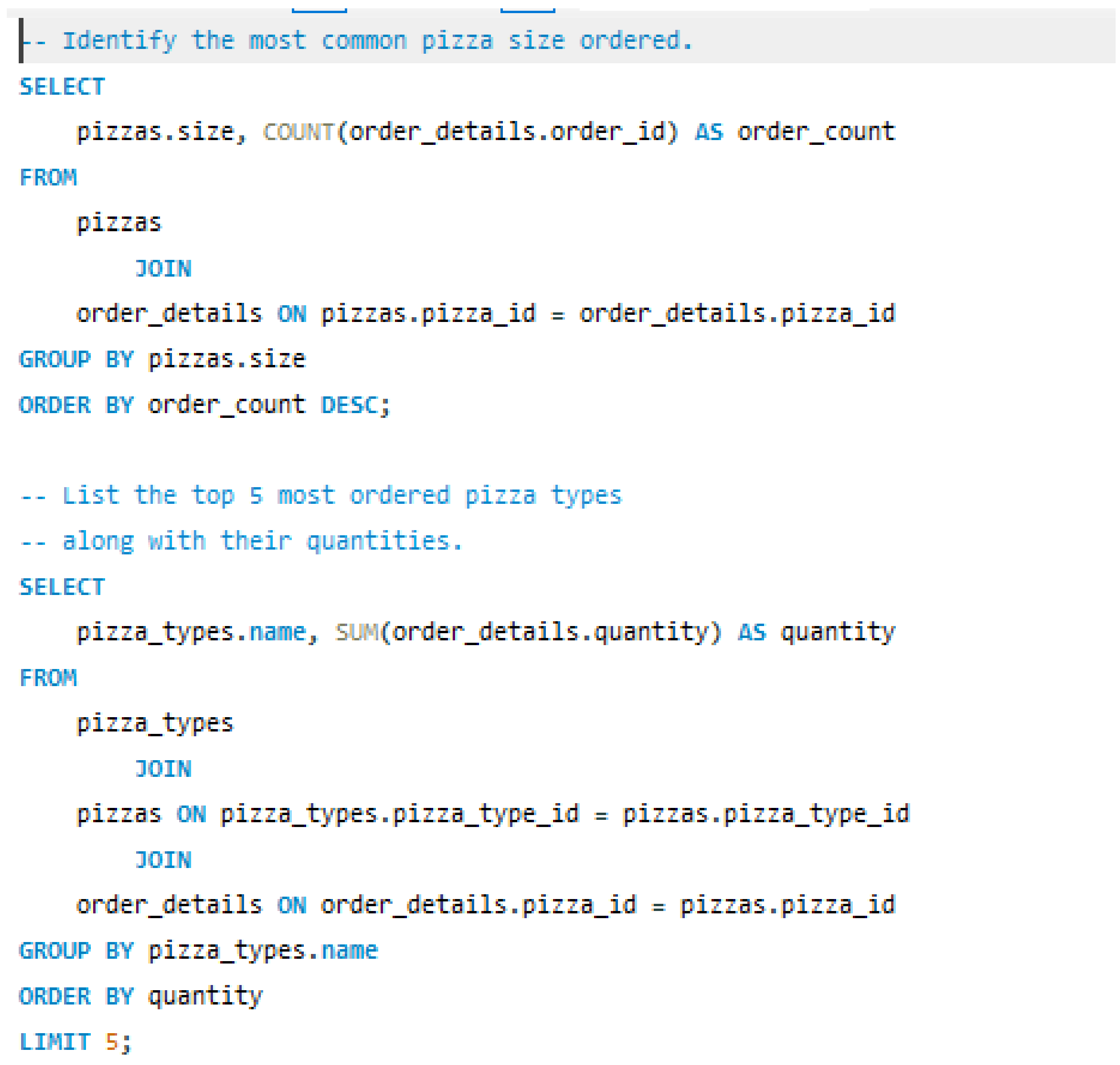
pizzahut

Pizza Sales Analysis: Uncovering Insights with SQL

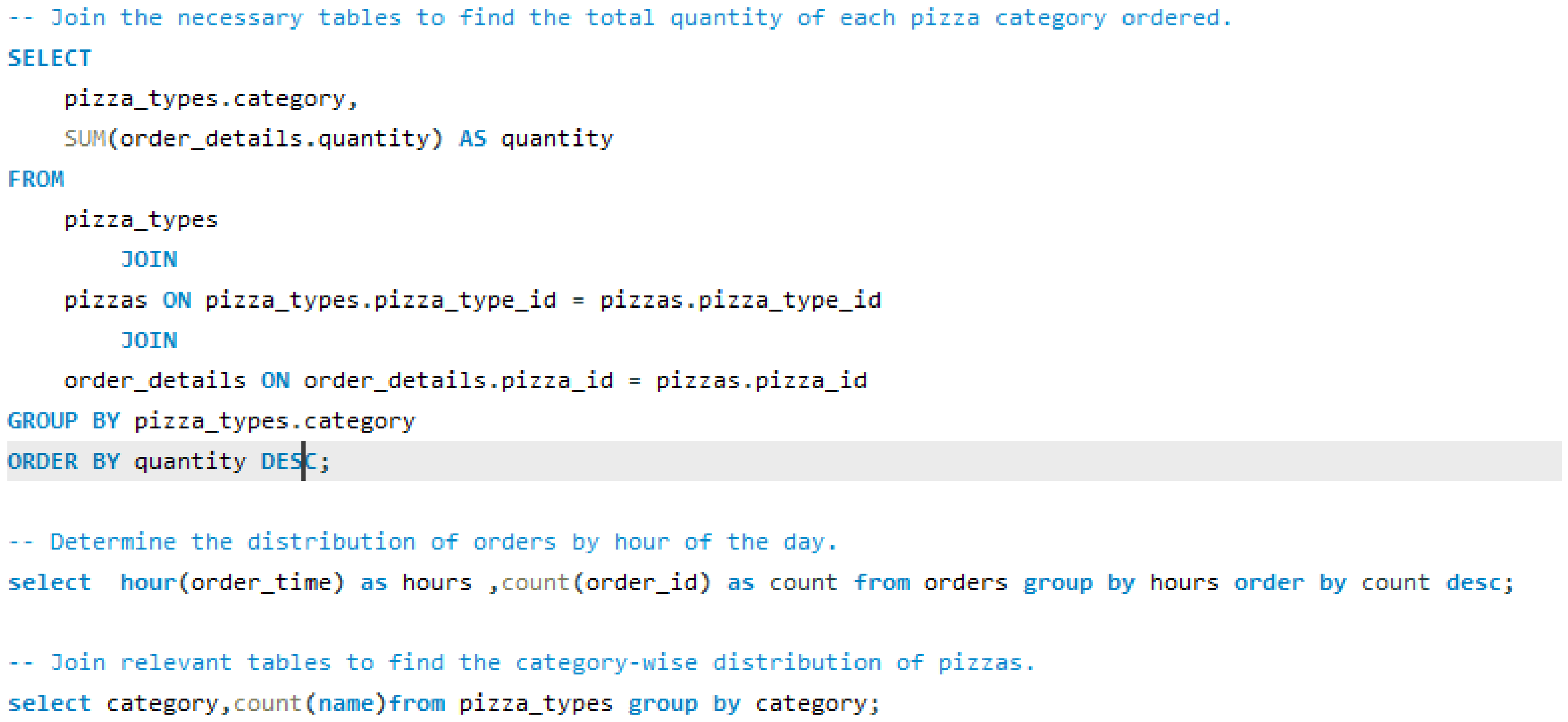
Project Description:

In this project, I utilized SQL to analyze pizza sales data to uncover key insights into sales performance. By querying the sales database, I focused on identifying the top-performing pizzas, analyzing revenue by pizza category, and calculating daily sales trends. The analysis involved ranking pizzas based on revenue, determining sales contributions per category, and extracting top sellers within each category. This project provides actionable insights for decision-making in inventory management, marketing strategies, and overall business performance improvement.

```sql
-- Retrieve the total number of orders placed.
select count(order_id)as total_orders from orders;

-- Calculate the total revenue generated from pizza sales.
SELECT
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS total_revenue
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id;

-- 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
ORDER BY pizzas.price DESC
LIMIT 1;
```

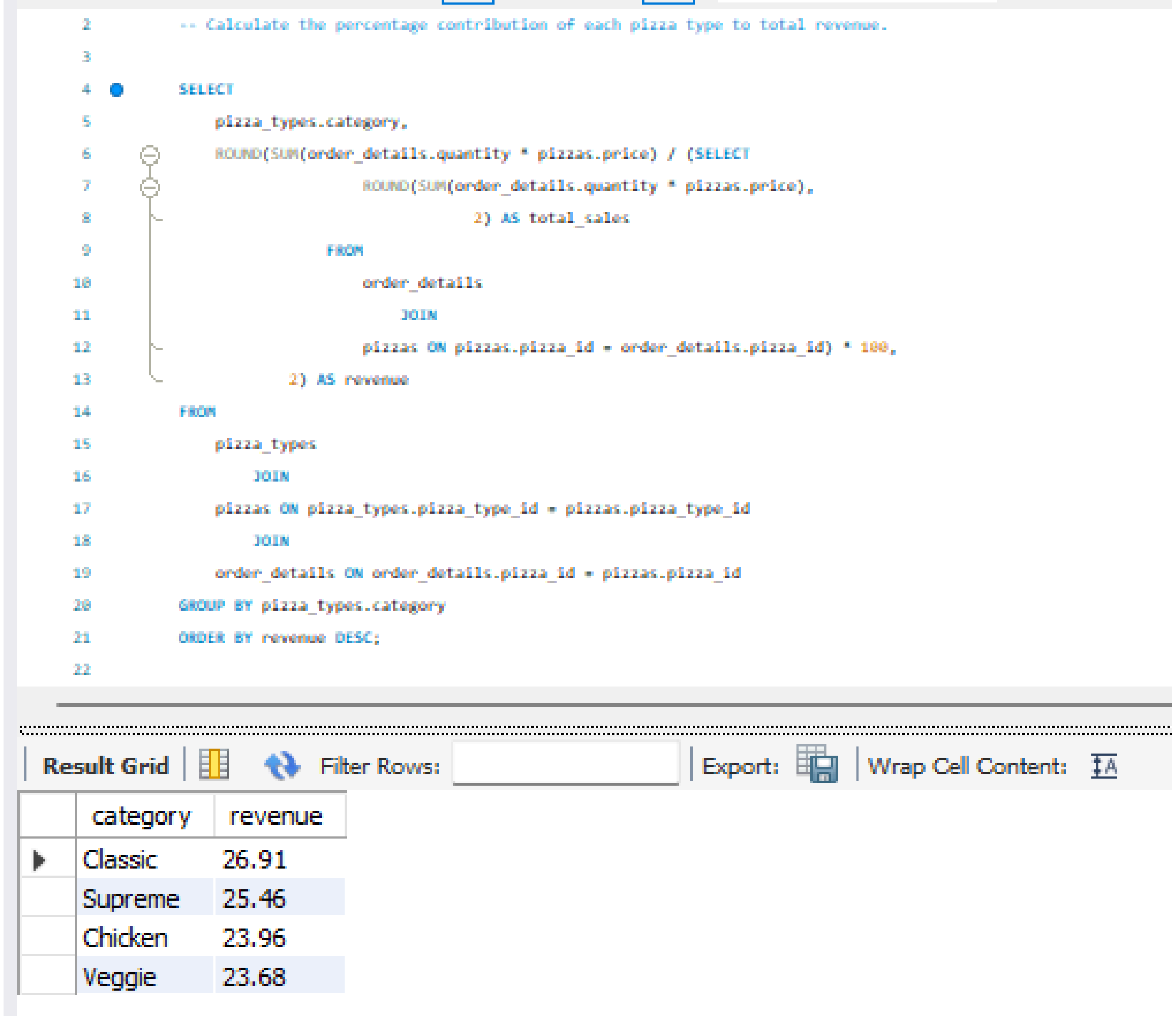


```sql
-- Identify the most common pizza size ordered.
SELECT
    pizzas.size, COUNT(order_details.order_id) AS order_count
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC;

-- 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 order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY quantity
LIMIT 5;
```



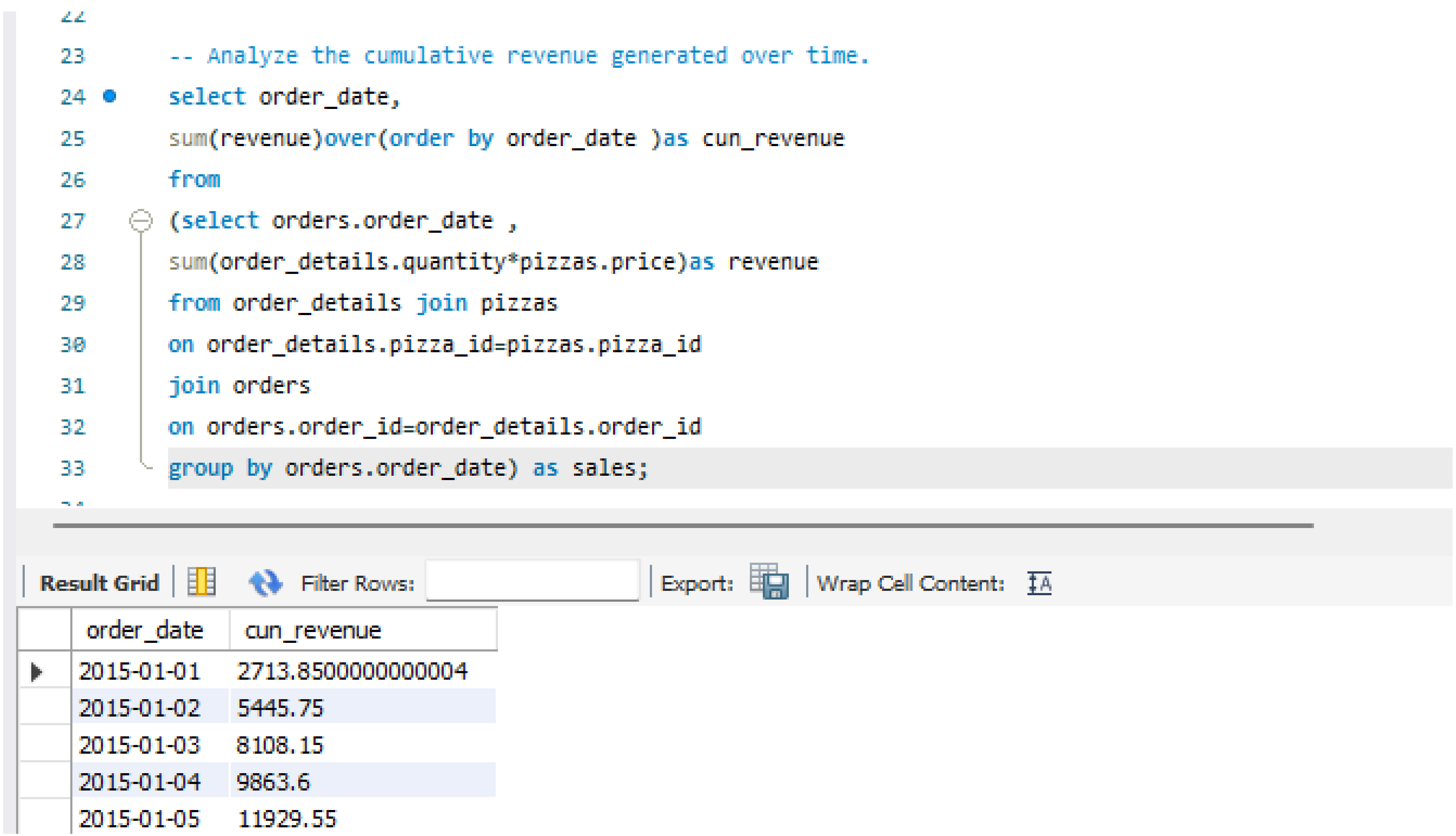
```sql
-- Join the necessary tables to find the total quantity of each pizza category ordered.
SELECT
    pizza_types.category,
    SUM(order_details.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity DESC;

-- Determine the distribution of orders by hour of the day.
select  hour(order_time) as hours ,count(order_id) as count from orders group by hours order by count desc;

-- Join relevant tables to find the category-wise distribution of pizzas.
select category,count(name)from pizza_types group by category;
```

```
-- Group the orders by date
-- and calculate the average number of pizzas ordered per day.
SELECT
    ROUND(AVG(quantity), 0) AS avg_quantity_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;

-- Determine the top 3 most ordered pizza types based on revenue.
SELECT
    pizza_types.name,
    SUM(order_details.quantity * pizzas.price) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.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 revenue DESC
LIMIT 3;
```



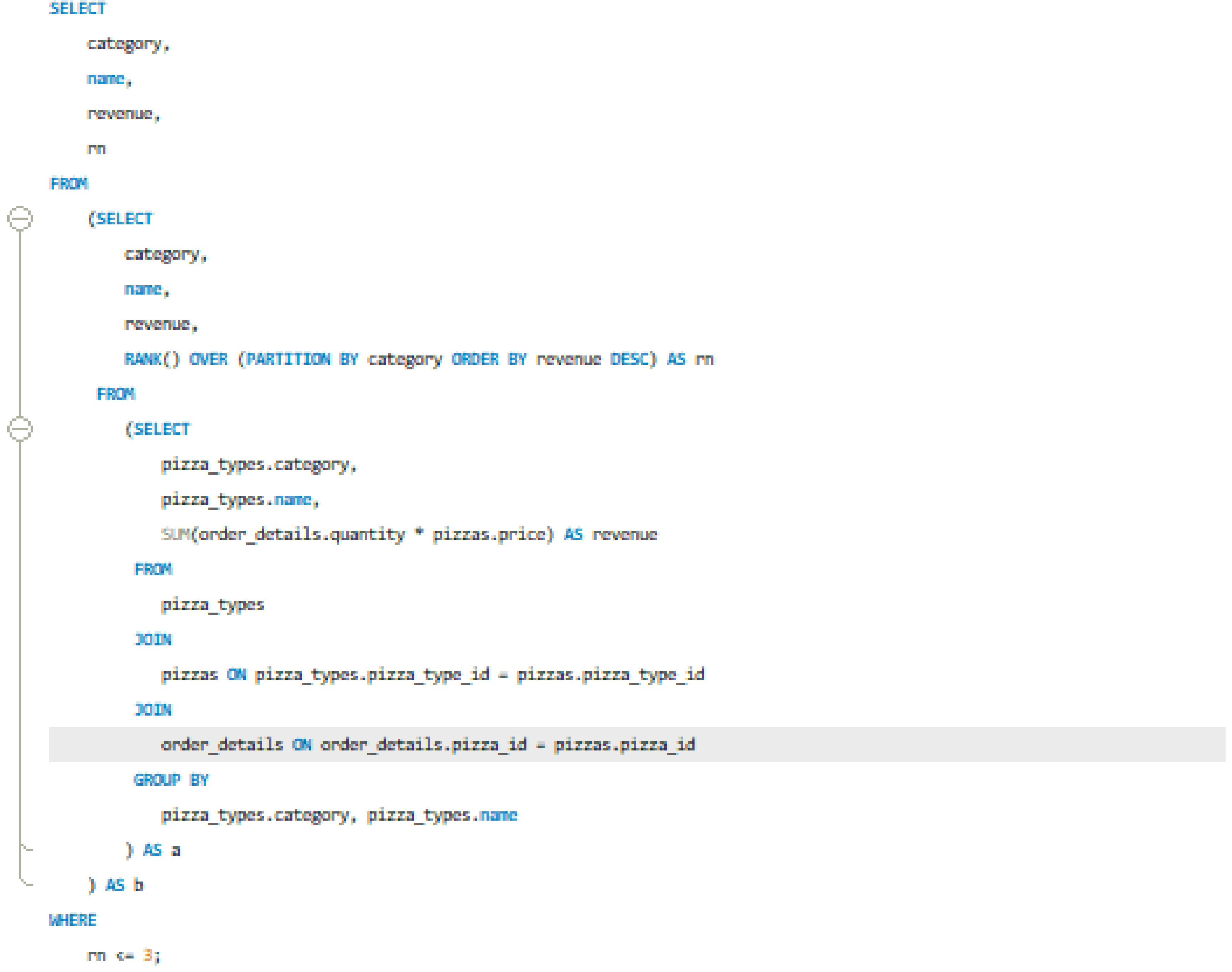
```sql
-- Calculate the percentage contribution of each pizza type to total revenue.

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 pizzas.pizza_id = order_details.pizza_id) * 100,
            2) 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
ORDER BY revenue DESC;
```

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68



```
-- Analyze the cumulative revenue generated over time.
select order_date,
sum(revenue)over(order by order_date )as cun_revenue
from
(select orders.order_date ,
sum(order_details.quantity*pizzas.price)as revenue
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;
```

order_date	cun_revenue
2015-01-01	2713.8500000000004
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55



```
SELECT
    category,
    name,
    revenue,
    rn
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;
```