PIZZA SALES ANALYSIS

BY ANAMIKA USING MYSQL

FOUR CSV FILES IMPORTED IN MYSQL

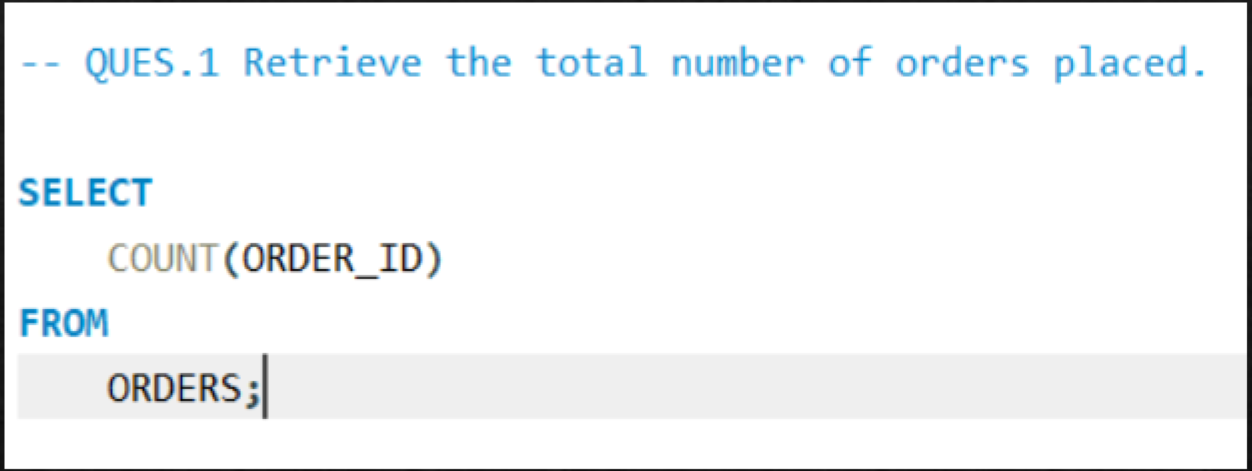
- The files and their respective columns are as follows:

- **order_details** : order_details_id, order_id, pizza_id, quantity
- Orders : order_id, date, time
- Pizzas : pizza_type_id, name, category, ingredients
- pizza_types : pizza_id, pizza_type_id, size, price

Data-Driven Insights on Pizza Sales Using MySQL

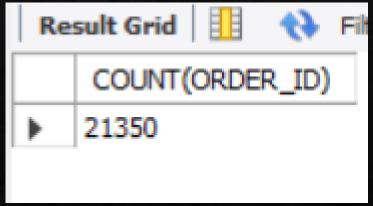
- Introduce the project:
- This analysis focuses on retrieving key insights from pizza sales data, answering questions ranging from basic queries to advanced analytics using MySQL.

BASIC QUESTIONS

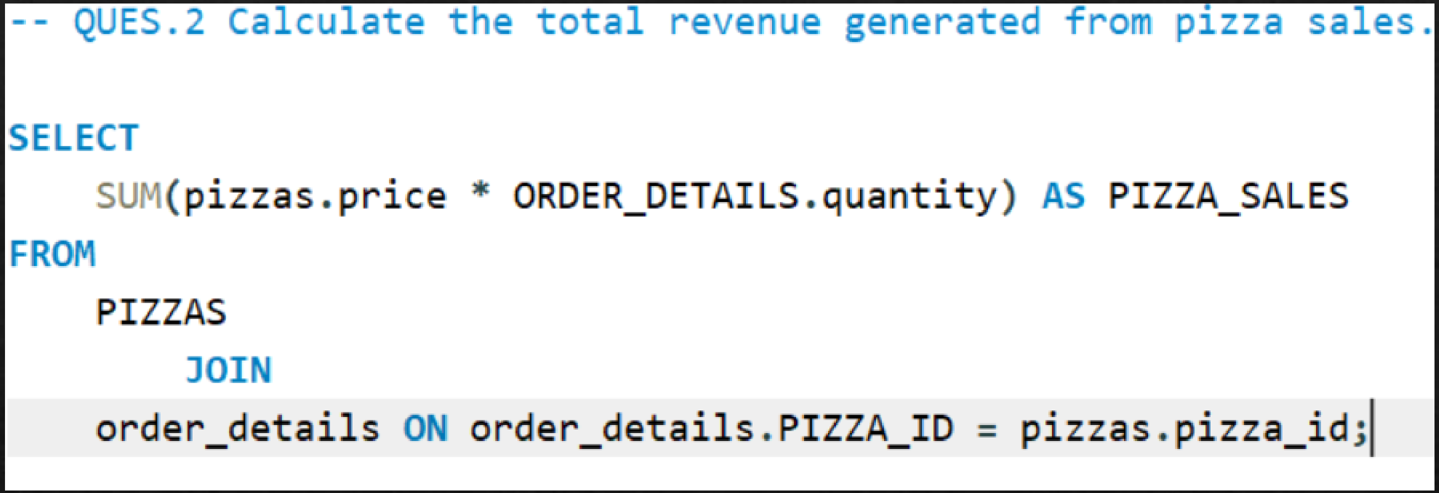


```
-- QUES.1 Retrieve the total number of orders placed.

SELECT
    COUNT(ORDER_ID)
FROM
    ORDERS;
```

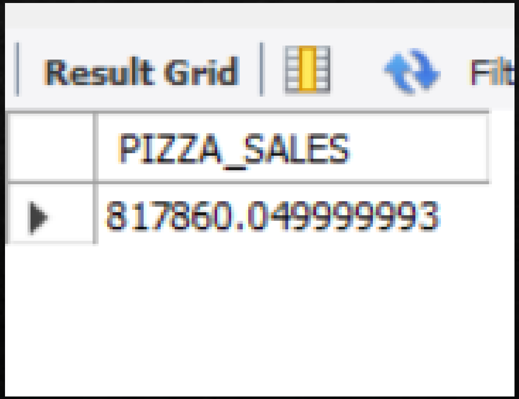


```
Result Grid
COUNT(ORDER_ID)
21350
```

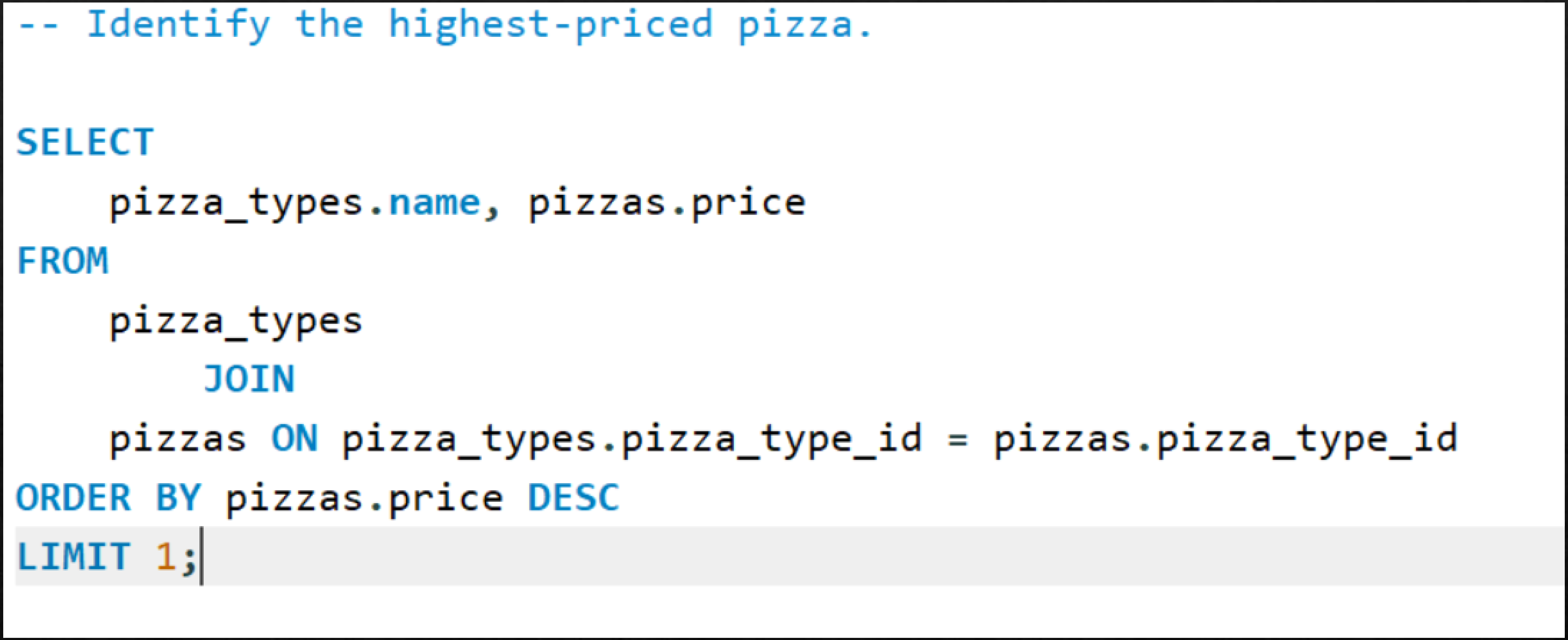


```
-- QUES.2 Calculate the total revenue generated from pizza sales.

SELECT
    SUM(pizzas.price * ORDER_DETAILS.quantity) AS PIZZA_SALES
FROM
    PIZZAS
        JOIN
    order_details ON order_details.PIZZA_ID = pizzas.pizza_id;
```

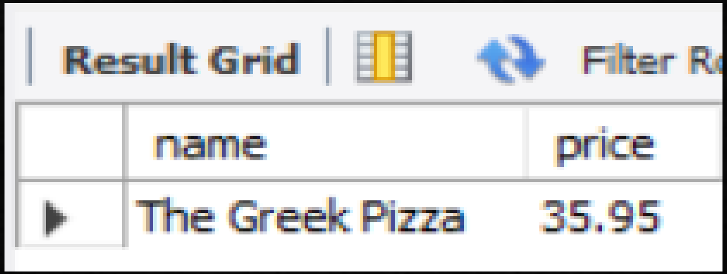


```
PIZZA_SALES
817860.049999993
```

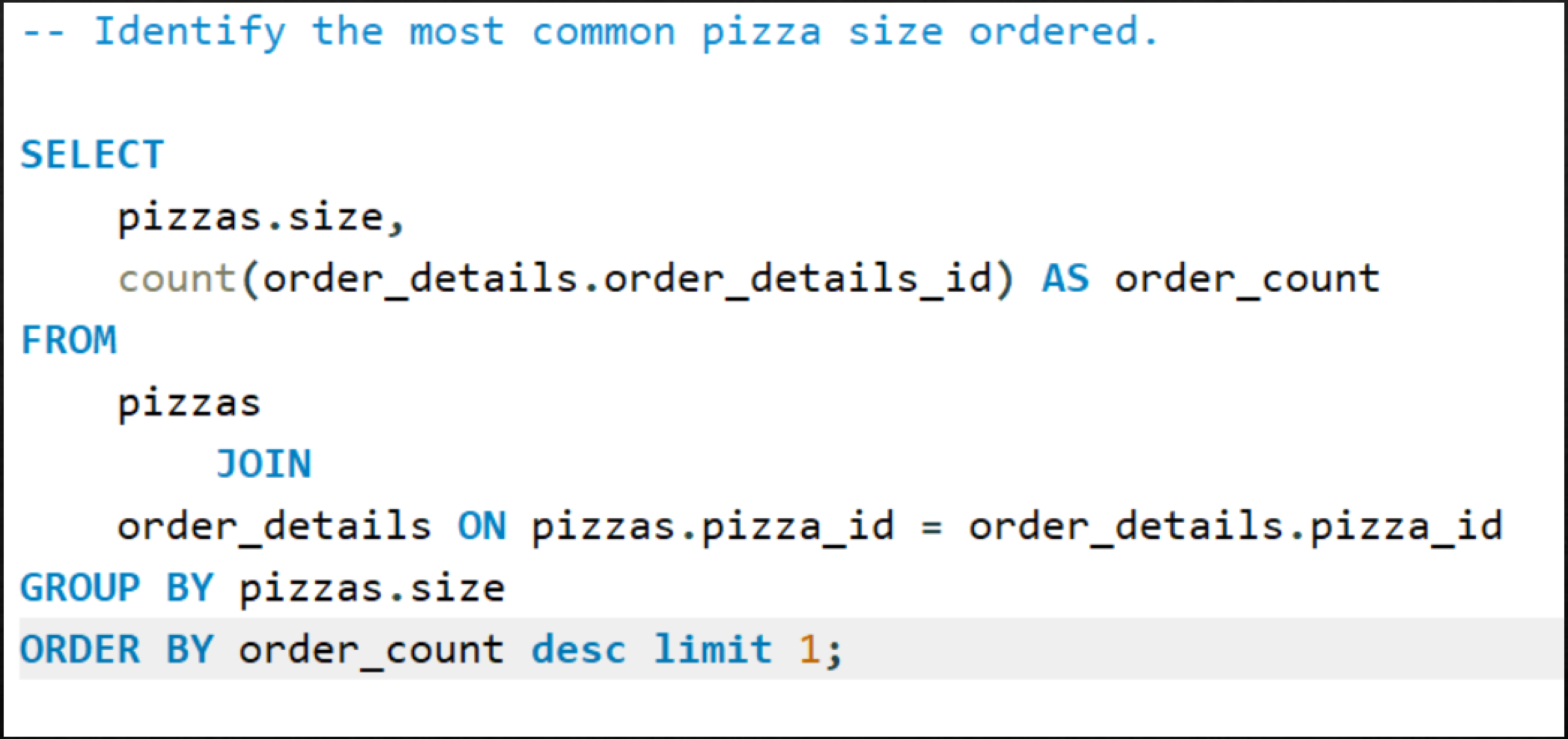


```sql
-- Identify the highest-priced pizza.

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;
```

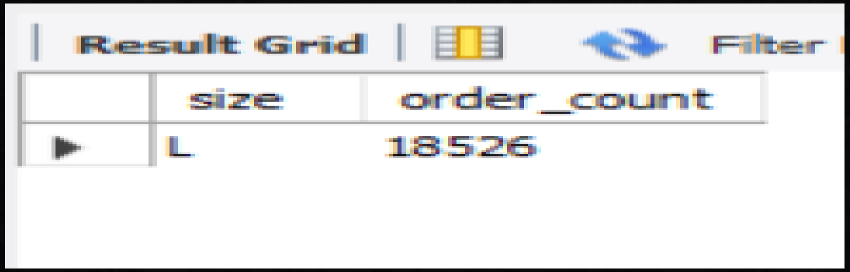


name	price
The Greek Pizza	35.95



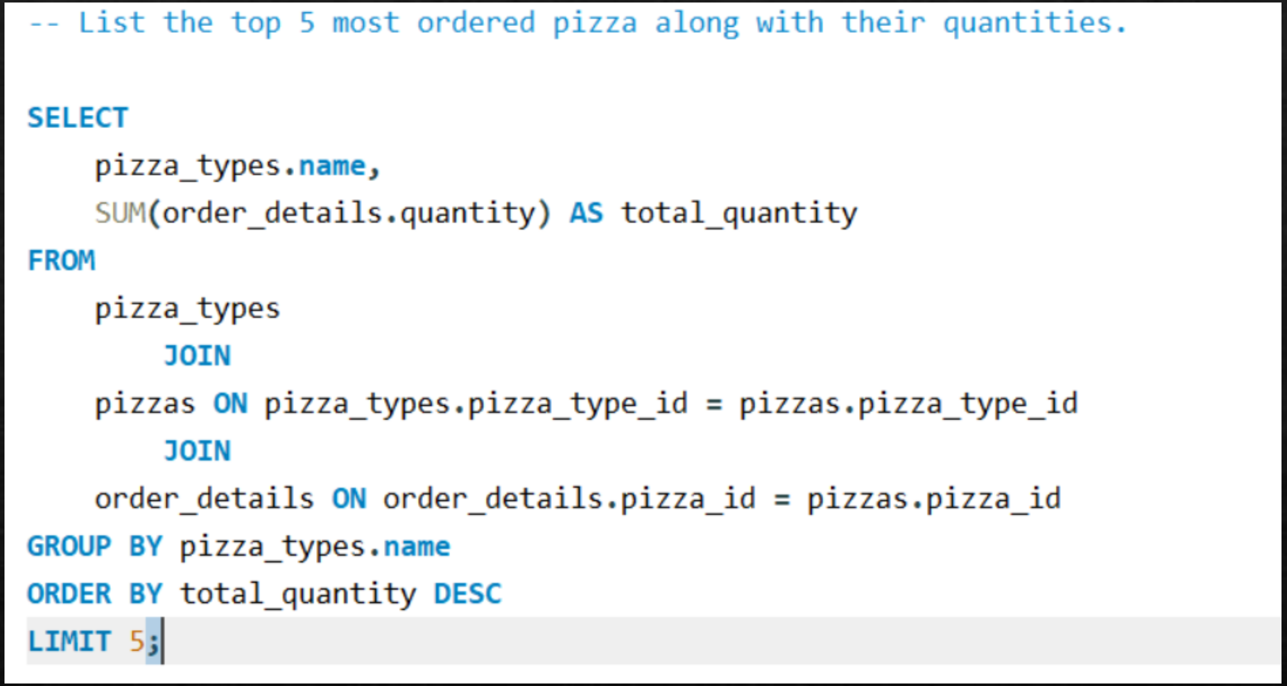
```sql
-- Identify the most common pizza size ordered.

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 limit 1;
```



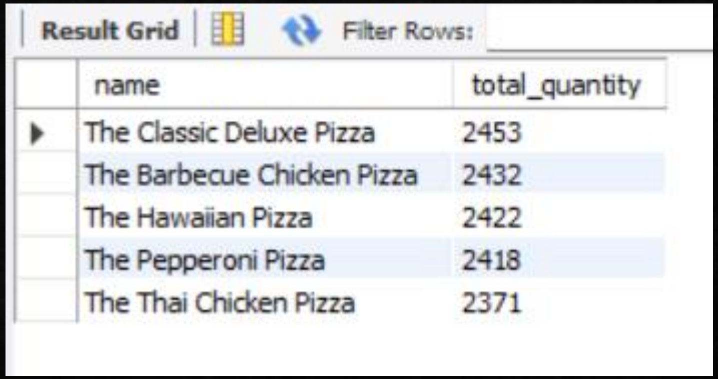
```
size  order_count
L     18526
```

INTERMEDIATE QUESTION

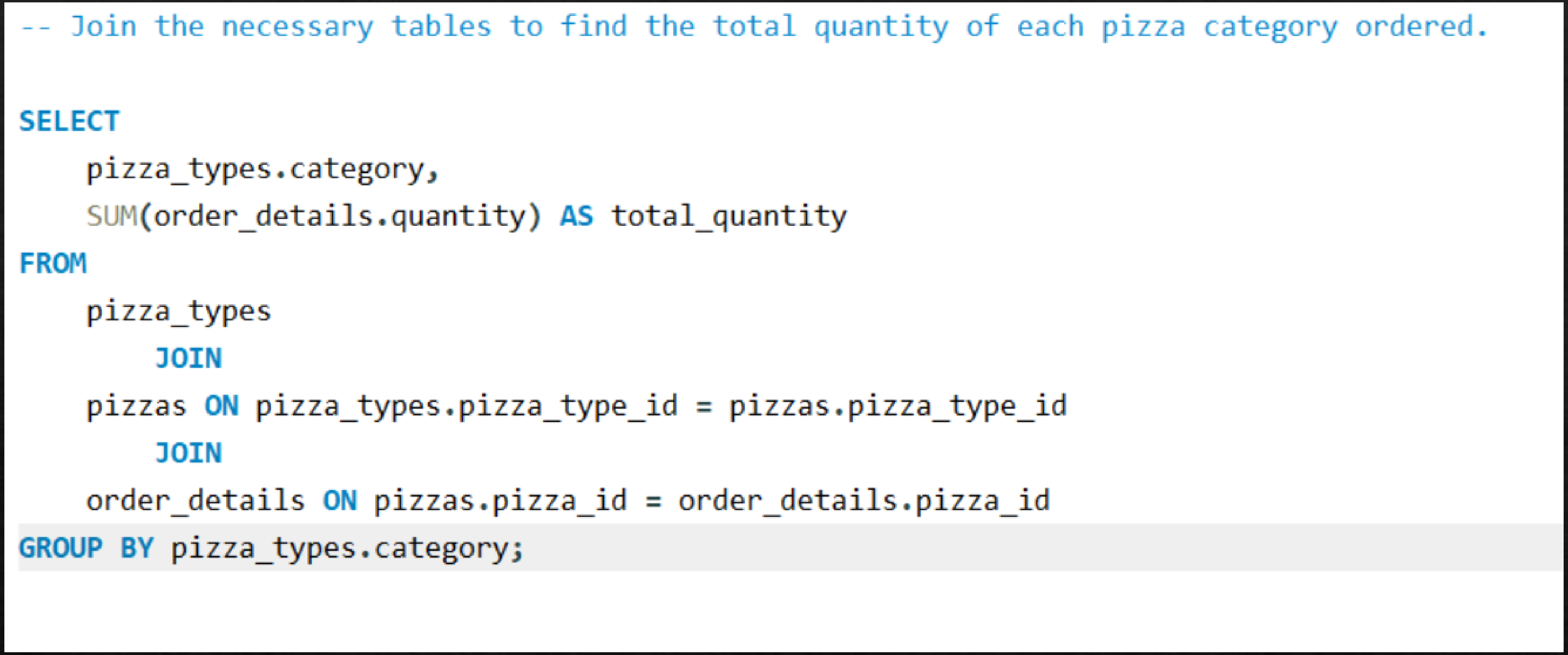


```sql
-- List the top 5 most ordered pizza along with their quantities.

SELECT
    pizza_types.name,
    SUM(order_details.quantity) AS total_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 total_quantity DESC
LIMIT 5;
```

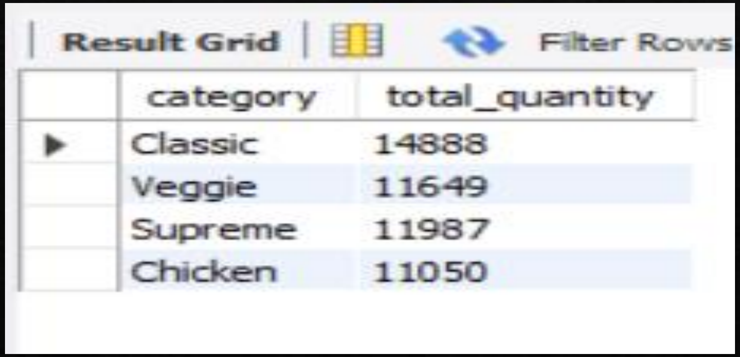


name	total_quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

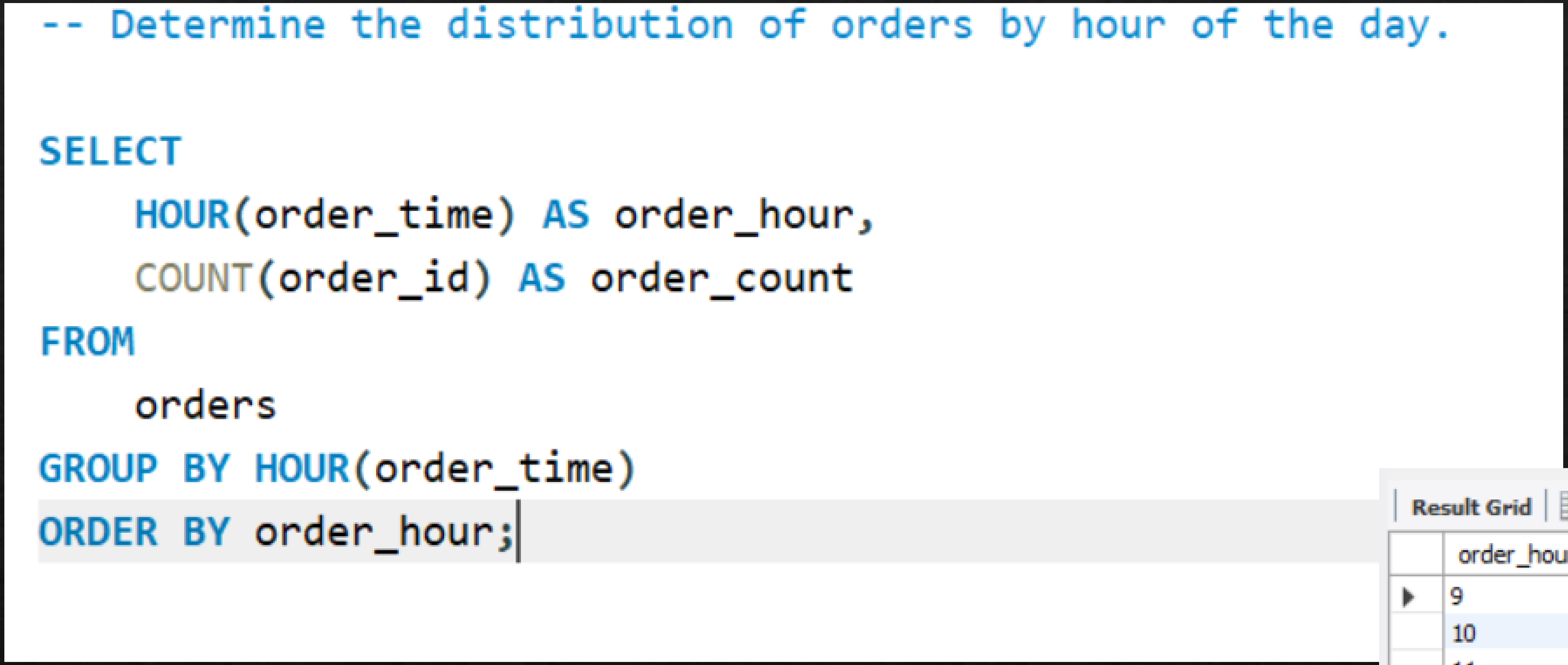


```sql
-- Join the necessary tables to find the total quantity of each pizza category ordered.

SELECT
    pizza_types.category,
    SUM(order_details.quantity) AS total_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.category;
```

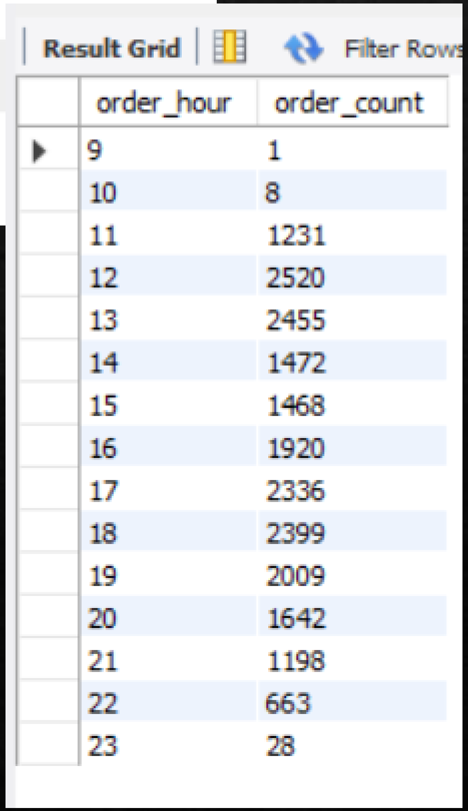


```
category   total_quantity
Classic    14888
Veggie     11649
Supreme    11987
Chicken    11050
```

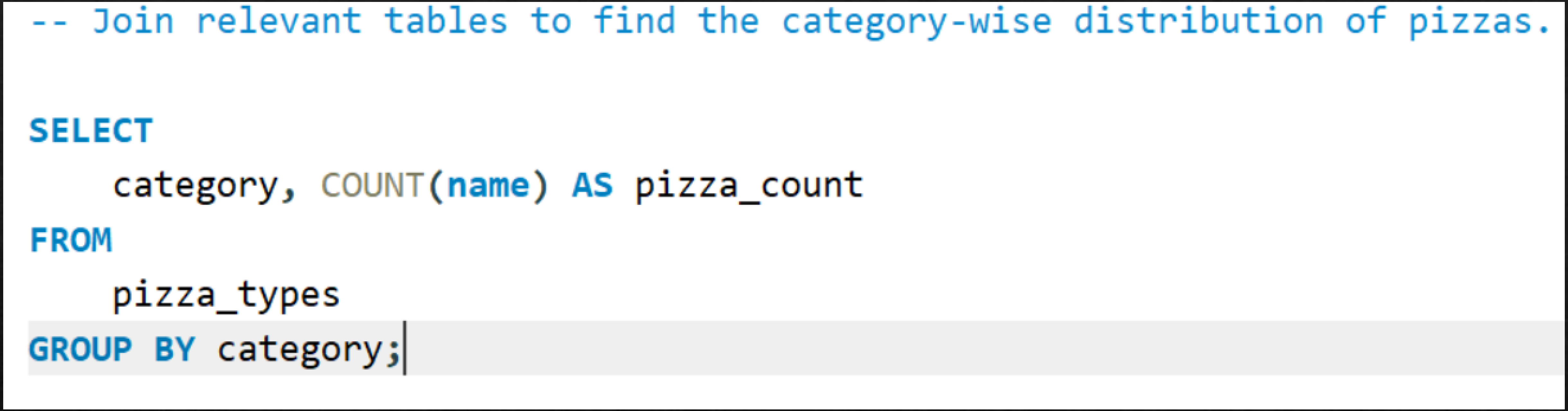


```sql
-- Determine the distribution of orders by hour of the day.

SELECT
    HOUR(order_time) AS order_hour,
    COUNT(order_id) AS order_count
FROM
    orders
GROUP BY HOUR(order_time)
ORDER BY order_hour;
```

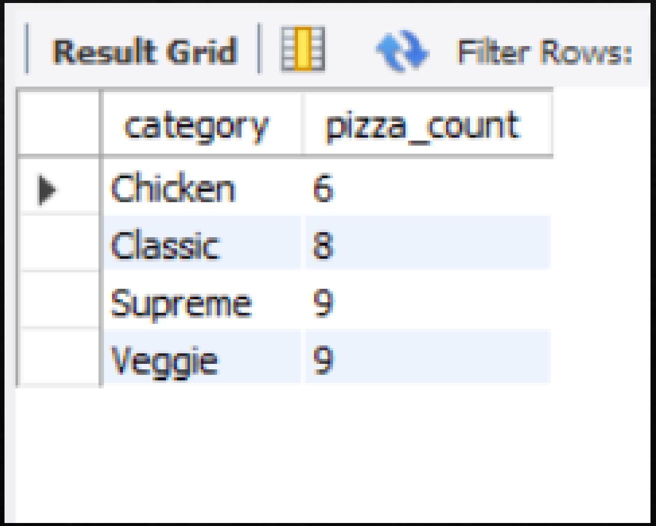


order_hour	order_count
9	1
10	8
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009
20	1642
21	1198
22	663
23	28

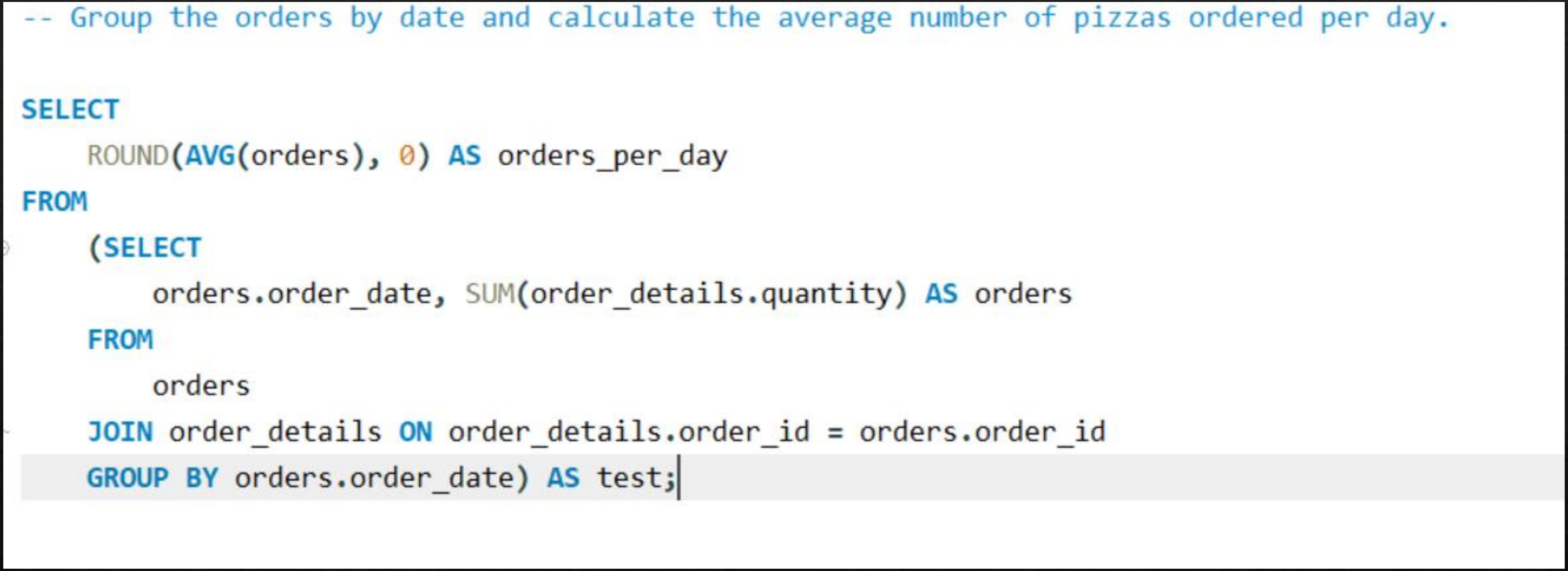


```
-- Join relevant tables to find the category-wise distribution of pizzas.

SELECT
    category, COUNT(name) AS pizza_count
FROM
    pizza_types
GROUP BY category;
```

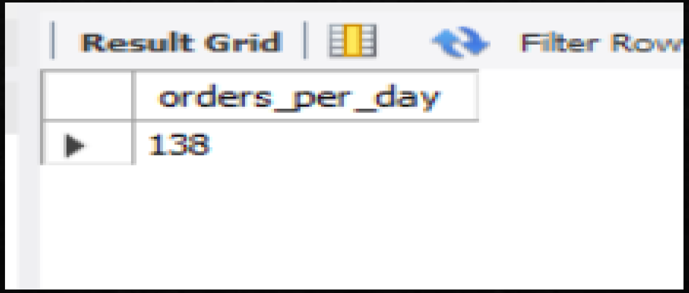


category	pizza_count
Chicken	6
Classic	8
Supreme	9
Veggie	9

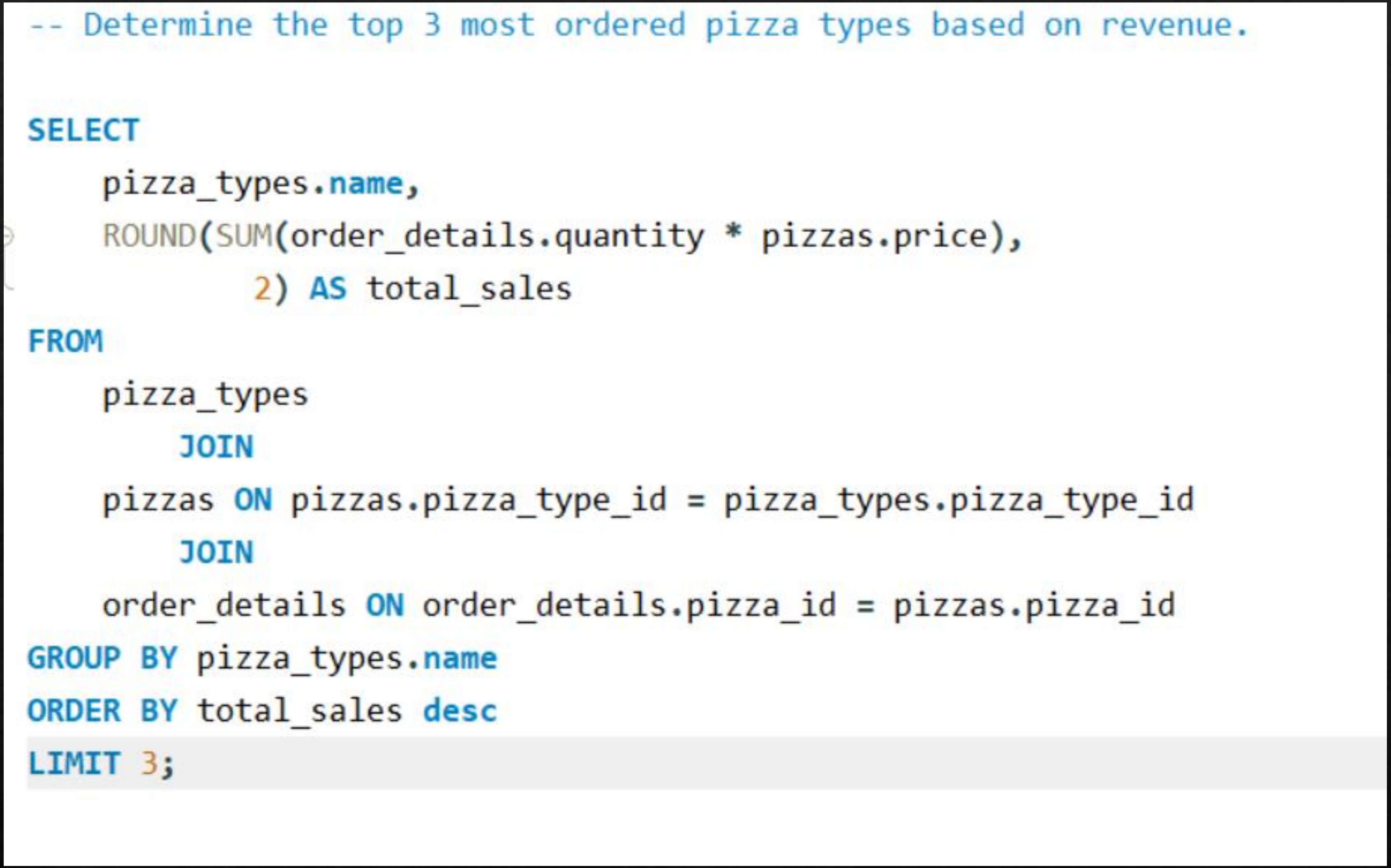


```
-- Group the orders by date and calculate the average number of pizzas ordered per day.

SELECT
    ROUND(AVG(orders), 0) AS orders_per_day
FROM
    (SELECT
        orders.order_date, SUM(order_details.quantity) AS orders
    FROM
        orders
    JOIN order_details ON order_details.order_id = orders.order_id
    GROUP BY orders.order_date) AS test;
```

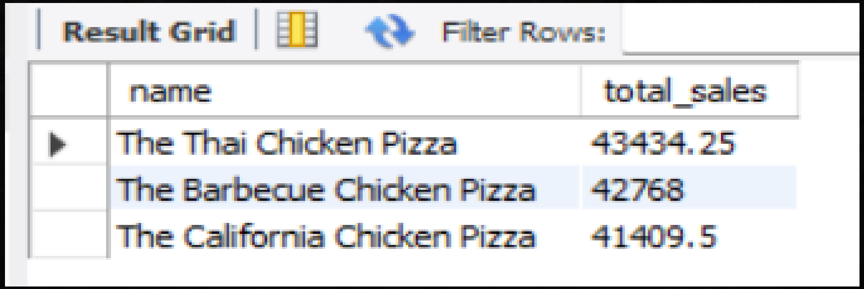


```
orders_per_day
138
```



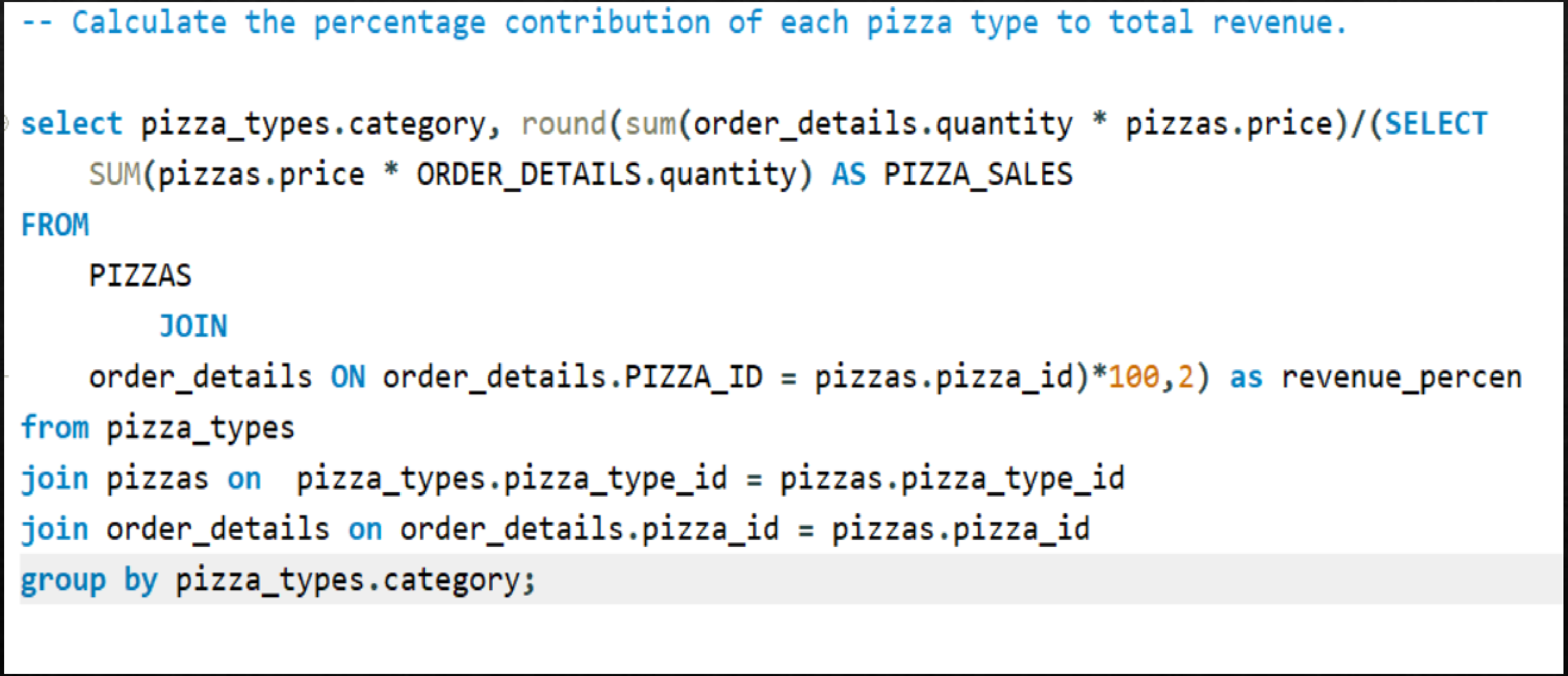
```
-- Determine the top 3 most ordered pizza types based on revenue.

SELECT
    pizza_types.name,
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS total_sales
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY total_sales desc
LIMIT 3;
```



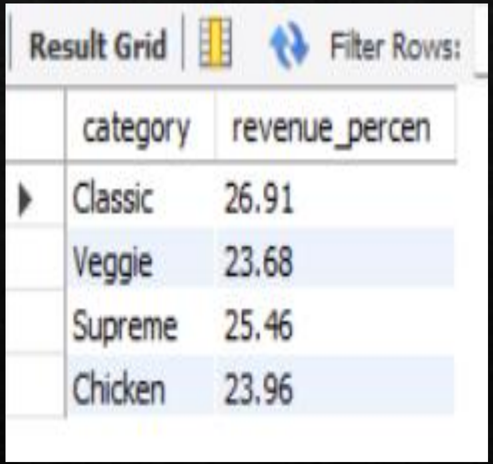
name	total_sales
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

ADVANCED QUESTIONS:

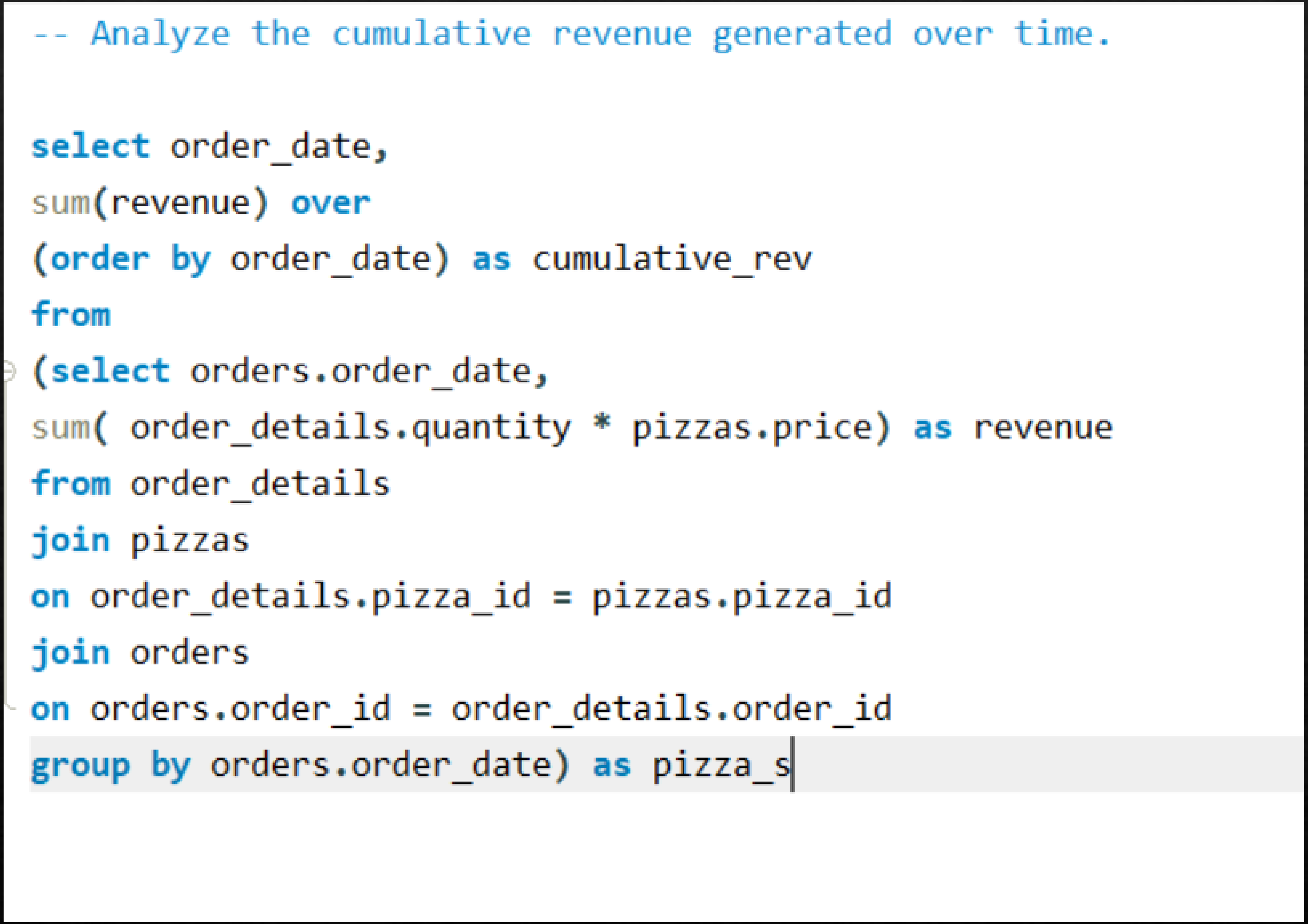


```
-- Calculate the percentage contribution of each pizza type to total revenue.

select pizza_types.category, round(sum(order_details.quantity * pizzas.price)/(SELECT
    SUM(pizzas.price * ORDER_DETAILS.quantity) AS PIZZA_SALES
FROM
    PIZZAS
        JOIN
    order_details ON order_details.PIZZA_ID = pizzas.pizza_id)*100,2) as revenue_percen
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;
```

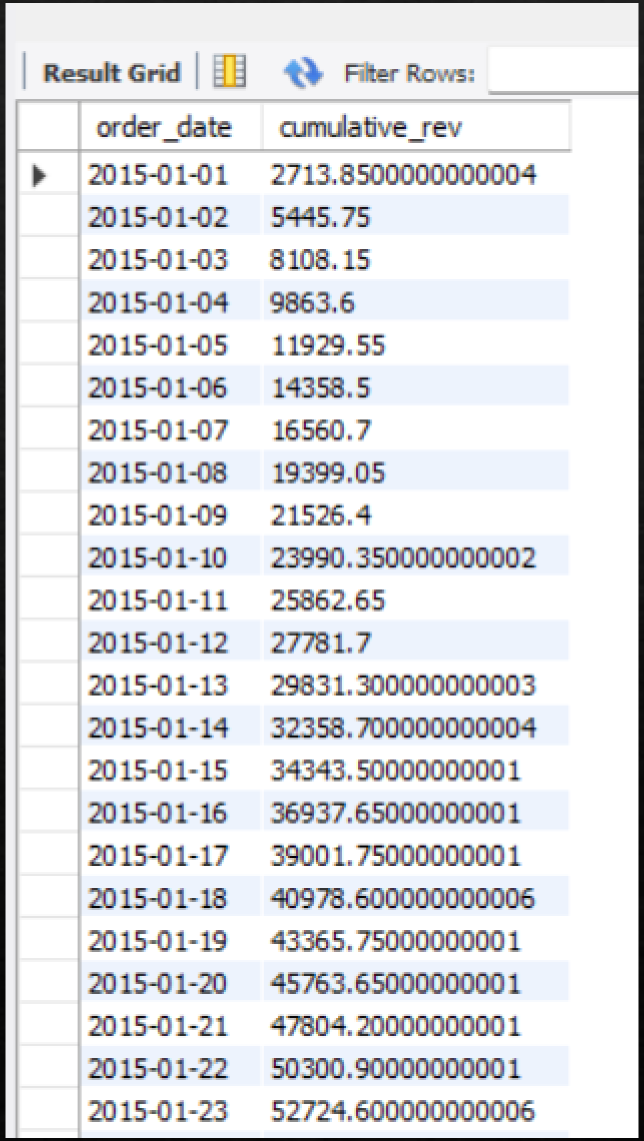


category	revenue_percen
Classic	26.91
Veggie	23.68
Supreme	25.46
Chicken	23.96

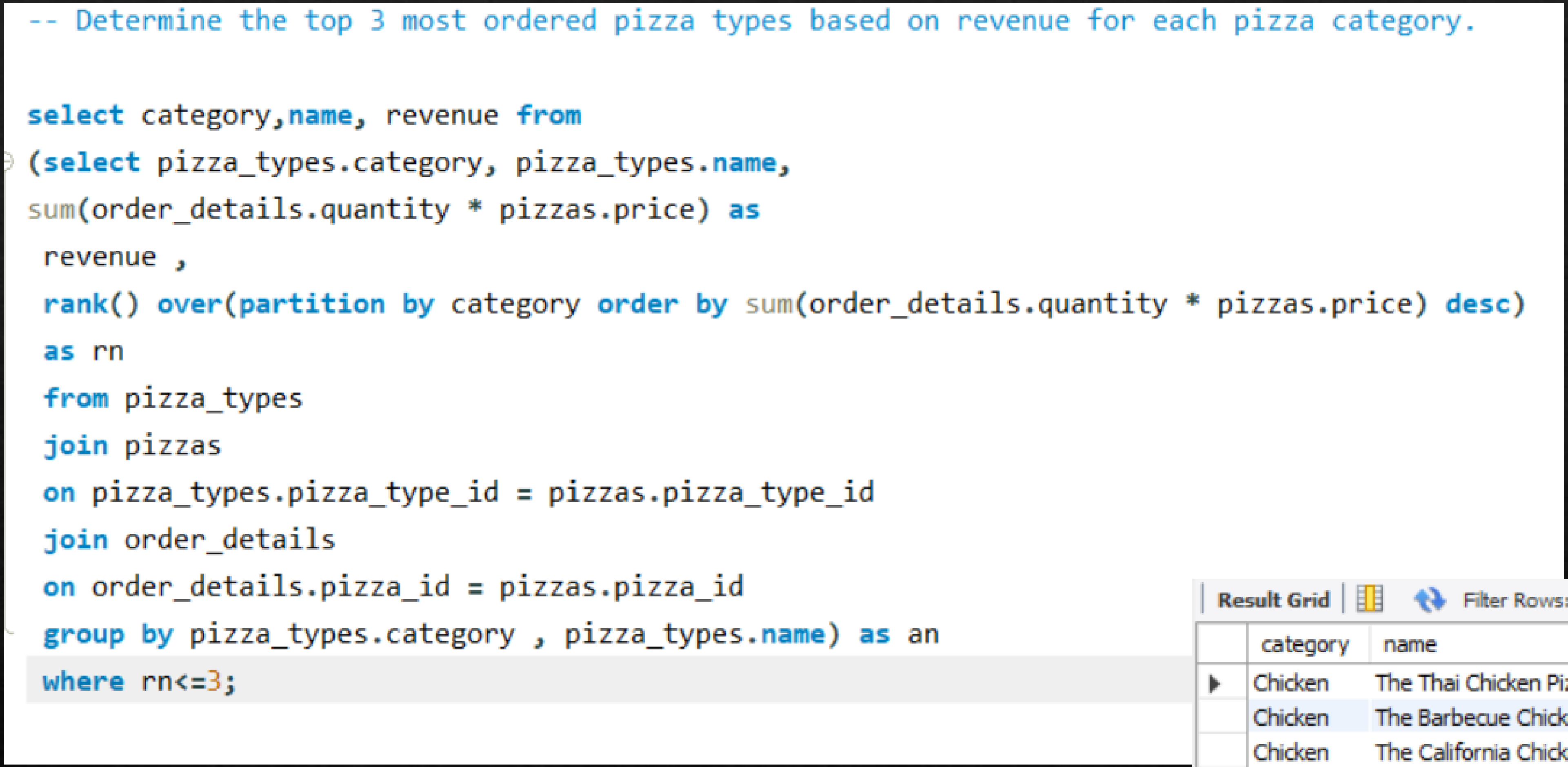


```sql
-- Analyze the cumulative revenue generated over time.

select order_date,
sum(revenue) over
(order by order_date) as cumulative_rev
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 pizza_s
```

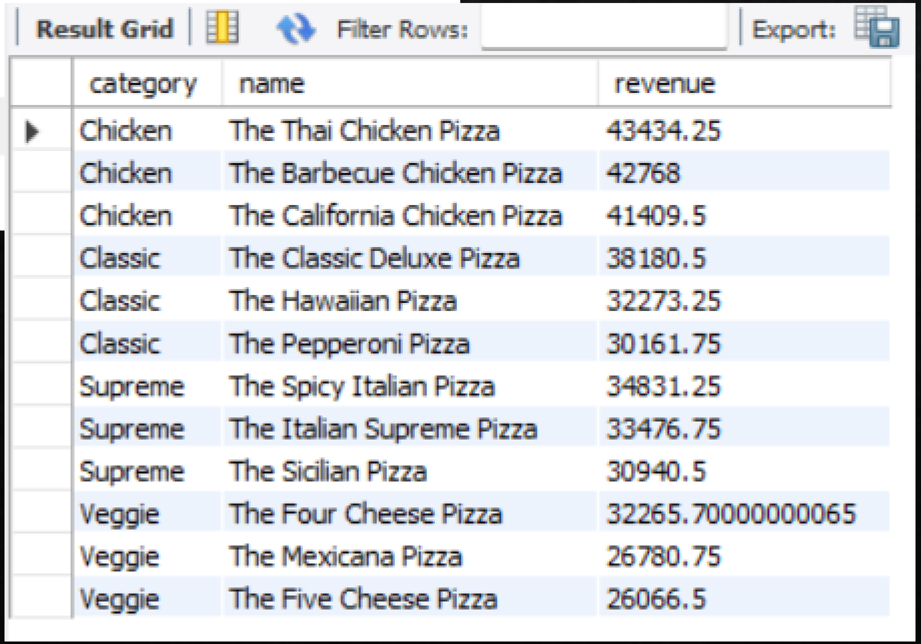


order_date	cumulative_rev
2015-01-01	2713.8500000000004
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55
2015-01-06	14358.5
2015-01-07	16560.7
2015-01-08	19399.05
2015-01-09	21526.4
2015-01-10	23990.350000000002
2015-01-11	25862.65
2015-01-12	27781.7
2015-01-13	29831.300000000003
2015-01-14	32358.700000000004
2015-01-15	34343.50000000001
2015-01-16	36937.65000000001
2015-01-17	39001.75000000001
2015-01-18	40978.600000000006
2015-01-19	43365.75000000001
2015-01-20	45763.65000000001
2015-01-21	47804.20000000001
2015-01-22	50300.90000000001
2015-01-23	52724.600000000006



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

select category,name, revenue from
(select pizza_types.category, pizza_types.name,
sum(order_details.quantity * pizzas.price) as
 revenue ,
 rank() over(partition by category order by sum(order_details.quantity * pizzas.price) desc)
 as rn
 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 an
 where rn<=3;
```



Result Grid | Filter Rows: | Export:

category	name	revenue
Chicken	The Thai Chicken Pizza	43434.25
Chicken	The Barbecue Chicken Pizza	42768
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	32265.70000000065
Veggie	The Mexicana Pizza	26780.75
Veggie	The Five Cheese Pizza	26066.5

Summary of Pizza Sales Analysis Using MySQL

In this project, I analyzed pizza sales data by answering a variety of business questions, ranging from basic to advanced, using MySQL.
The analysis was conducted on four CSV files imported into MySQL: **order_details**, **Orders**, **Pizzas**, and **pizza_types**, each containing critical sales data.

Key Insights:

1. **Total Orders**:
    1. I retrieved the total number of unique orders placed.
2. **Revenue Generation**:
    1. The total revenue generated from pizza sales was calculated by summing up the total quantity of pizzas ordered multiplied by their prices.
3. **Pizza Trends**:
    1. The highest-priced pizza and the most commonly ordered pizza size were identified, providing insight into customer preferences.
    2. The top 5 most ordered pizza types were highlighted, showing which pizzas were the most popular.
4. **Category and Time Analysis** (Intermediate Level):
    1. By joining the necessary tables, I determined the total quantity of each pizza category ordered and analyzed the distribution of orders by hour of the day, uncovering peak ordering times.
    2. The average number of pizzas ordered per day was calculated to track sales performance over time.
5. **Revenue and Category Analysis** (Advanced Level):
    1. I calculated the percentage contribution of each pizza type to the total revenue.
    2. A cumulative analysis of revenue was done over time, helping visualize how sales grew.
    3. The top 3 most ordered pizza types based on revenue were identified for each pizza category.

Conclusion:

This comprehensive MySQL analysis helped in uncovering vital business insights about pizza sales.
By analyzing trends in orders, revenue, and pizza preferences,
valuable data-driven recommendations can be made to optimize inventory, marketing strategies, and customer satisfaction.

THANK YOU!

Connect with Me on LinkedIn
LinkedIn Profile :ANAMIKA’S PROFILE