Borcelle

PIZZA RESTAURANT

SQL PROJECT

Sales Data Analysis

PIZZA HUT

CHECK IT NOW >>

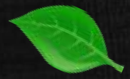
Introduction

Hello, my name is Laiba Hameed, and I am a BS IT student. I have learned SQL and MySQL for practice and to enhance my understanding of database management. In this project, I have utilized SQL queries to solve various questions related to PizzaHut sales, including analyzing order details, pizza categories, and revenue data. This project helped me apply my SQL skills in a real-world context and improve my ability to work with relational databases.

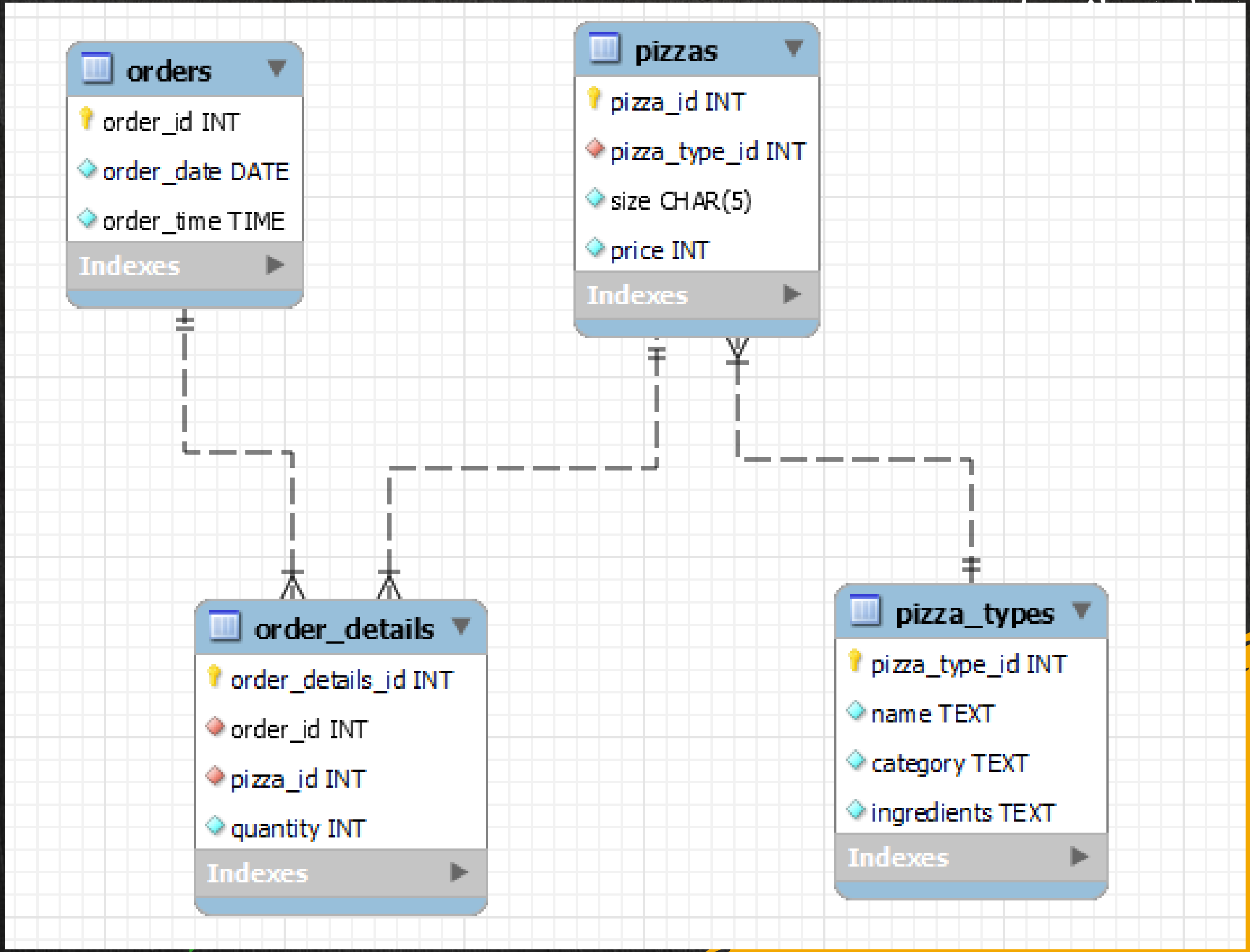
Overview of Pizza Hut Database

The PizzaHut database consists of four main tables to manage pizzas, pizza types, customer orders, and order details:

1. **pizzas**: Stores information about available pizzas.
2. **pizza_types**: Defines the categories and ingredients for each type of pizza.
3. **orders**: Manages customer orders and their details.
4. **order_details**: Stores the relationship between orders and pizzas, including quantities.



Entity-Relationship Diagram (ERD)

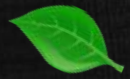


orders
order_id INT
order_date DATE
order_time TIME
Indexes

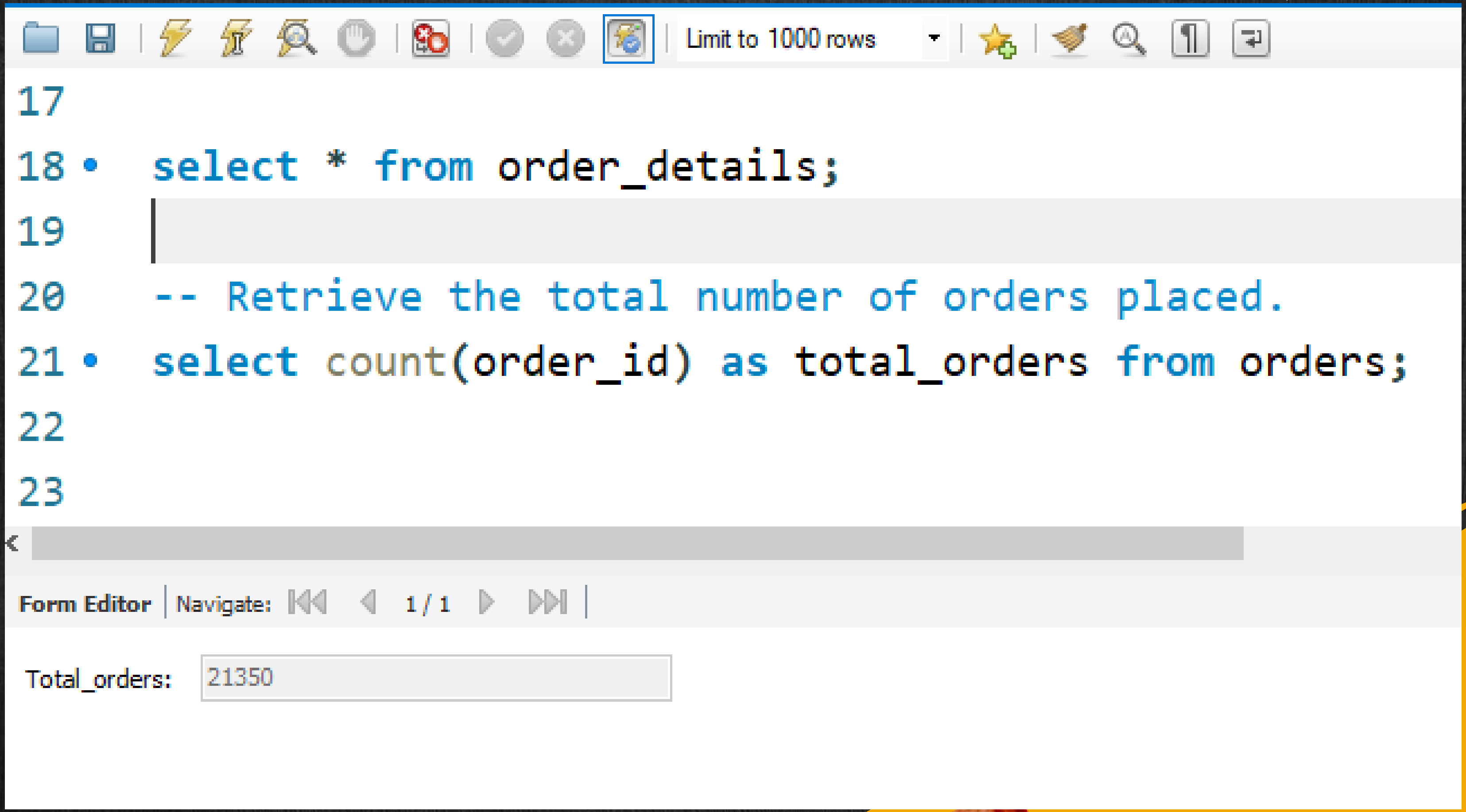
pizzas
pizza_id INT
pizza_type_id INT
size CHAR(5)
price INT
Indexes

order_details
order_details_id INT
order_id INT
pizza_id INT
quantity INT
Indexes

pizza_types
pizza_type_id INT
name TEXT
category TEXT
ingredients TEXT
Indexes



Q1: Retrieve the total number of orders placed.

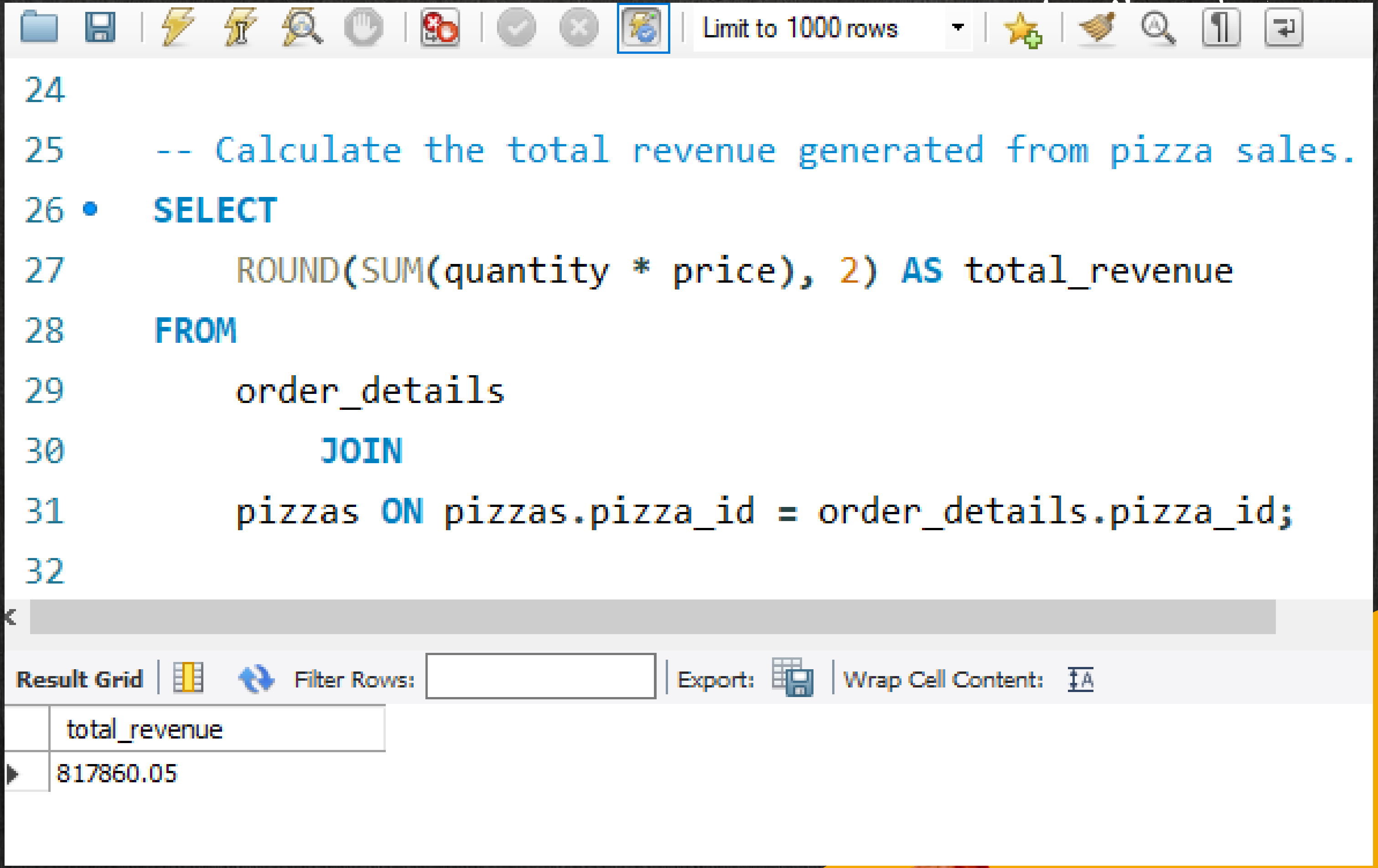


Limit to 1000 rows

```sql

select * from order_details;


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


```

Total_orders: 21350

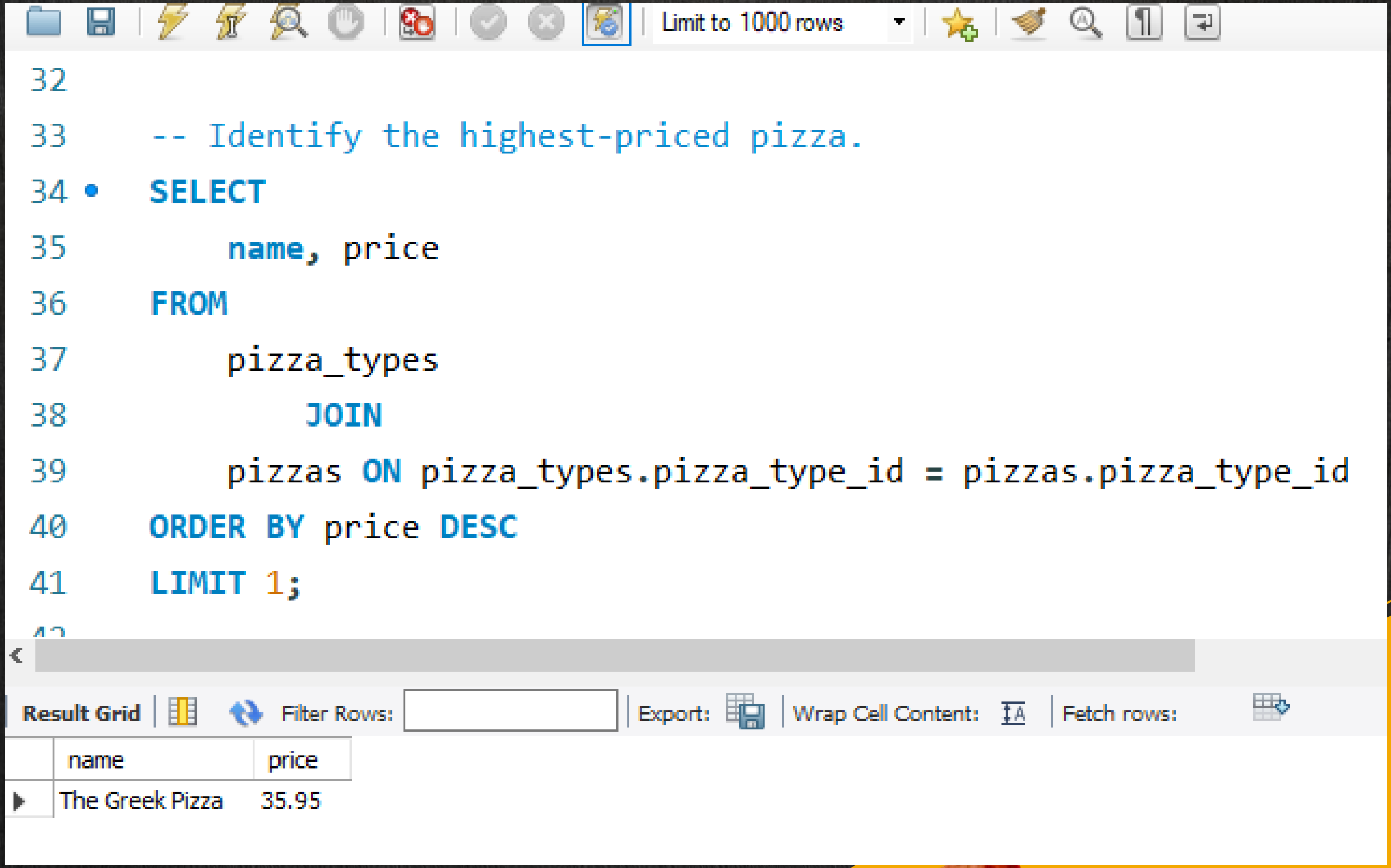
Q2: Calculate the total revenue generated from pizza sales.



```sql

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

```

```
total_revenue
817860.05
```

Q3: Identify the highest-priced pizza

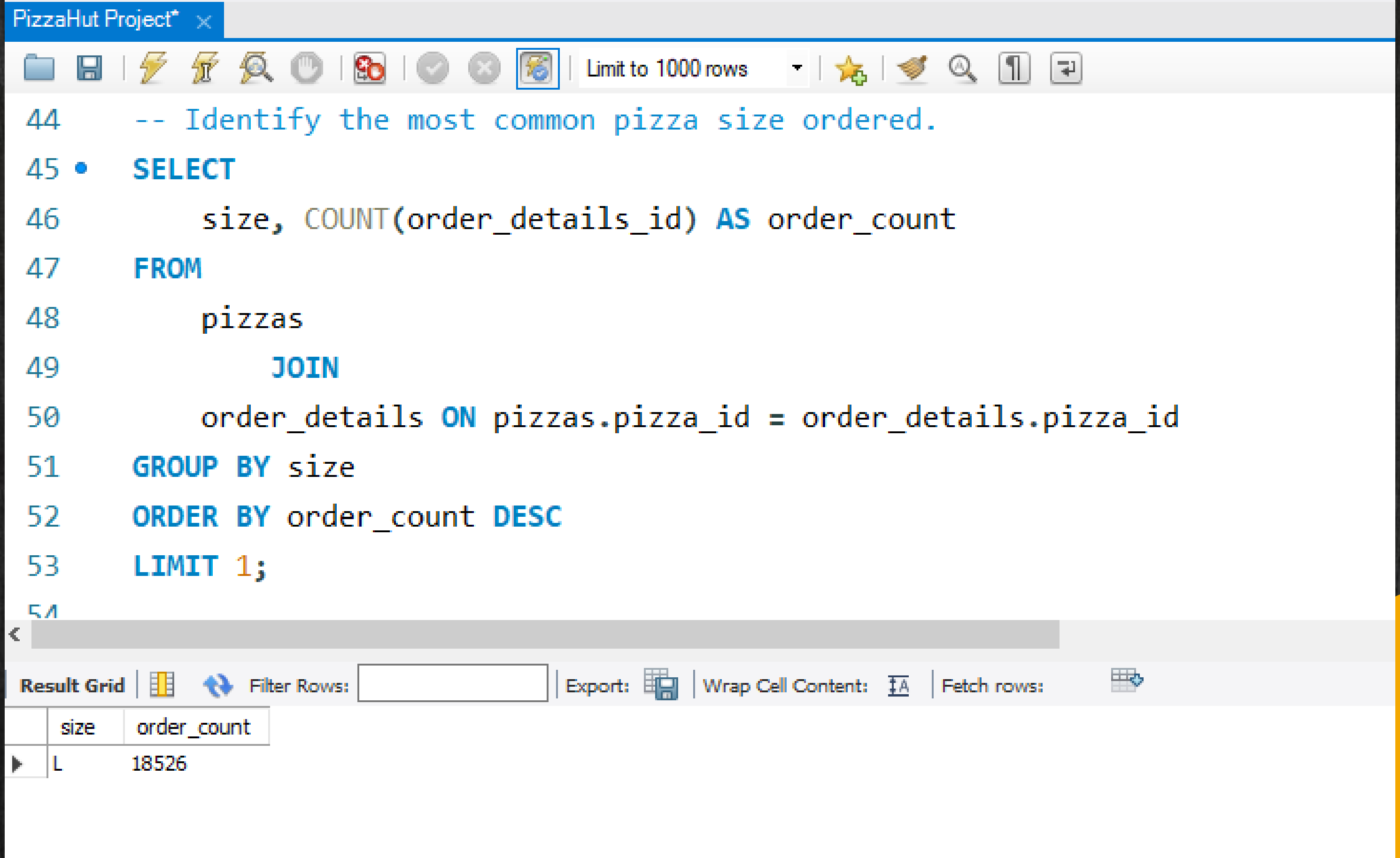


Limit to 1000 rows

```sql

-- Identify the highest-priced pizza.
SELECT
    name, price
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY price DESC
LIMIT 1;
```

name	price
The Greek Pizza	35.95

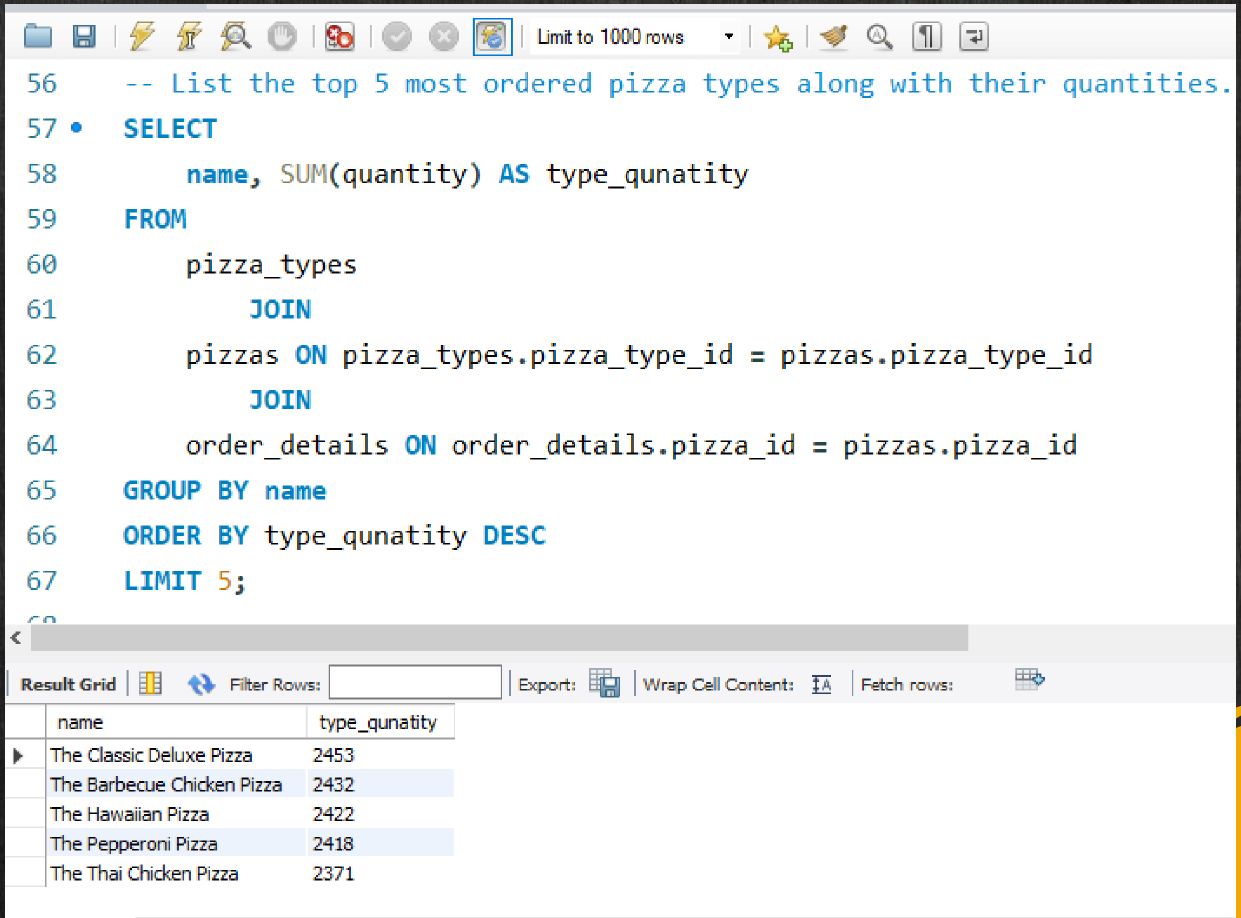
Q4: Identify the most common pizza size ordered.



```
-- Identify the most common pizza size ordered.
SELECT
    size, COUNT(order_details_id) AS order_count
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY size
ORDER BY order_count DESC
LIMIT 1;
```

size	order_count
L	18526

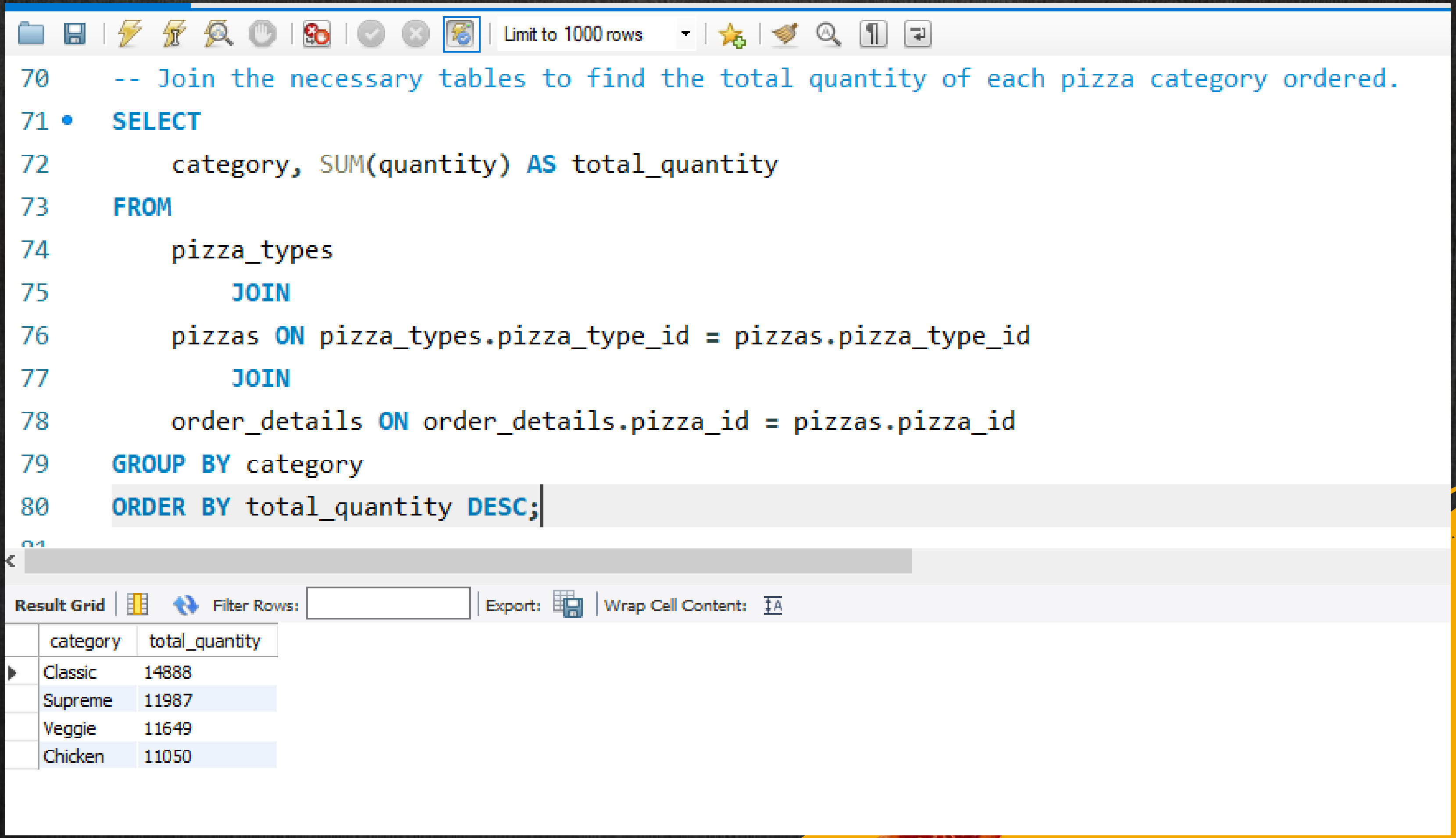
Q5: List the top 5 most ordered pizza types along with their quantities.



```sql
-- List the top 5 most ordered pizza types along with their quantities.
SELECT
    name, SUM(quantity) AS type_qunatity
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 name
ORDER BY type_qunatity DESC
LIMIT 5;
```

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

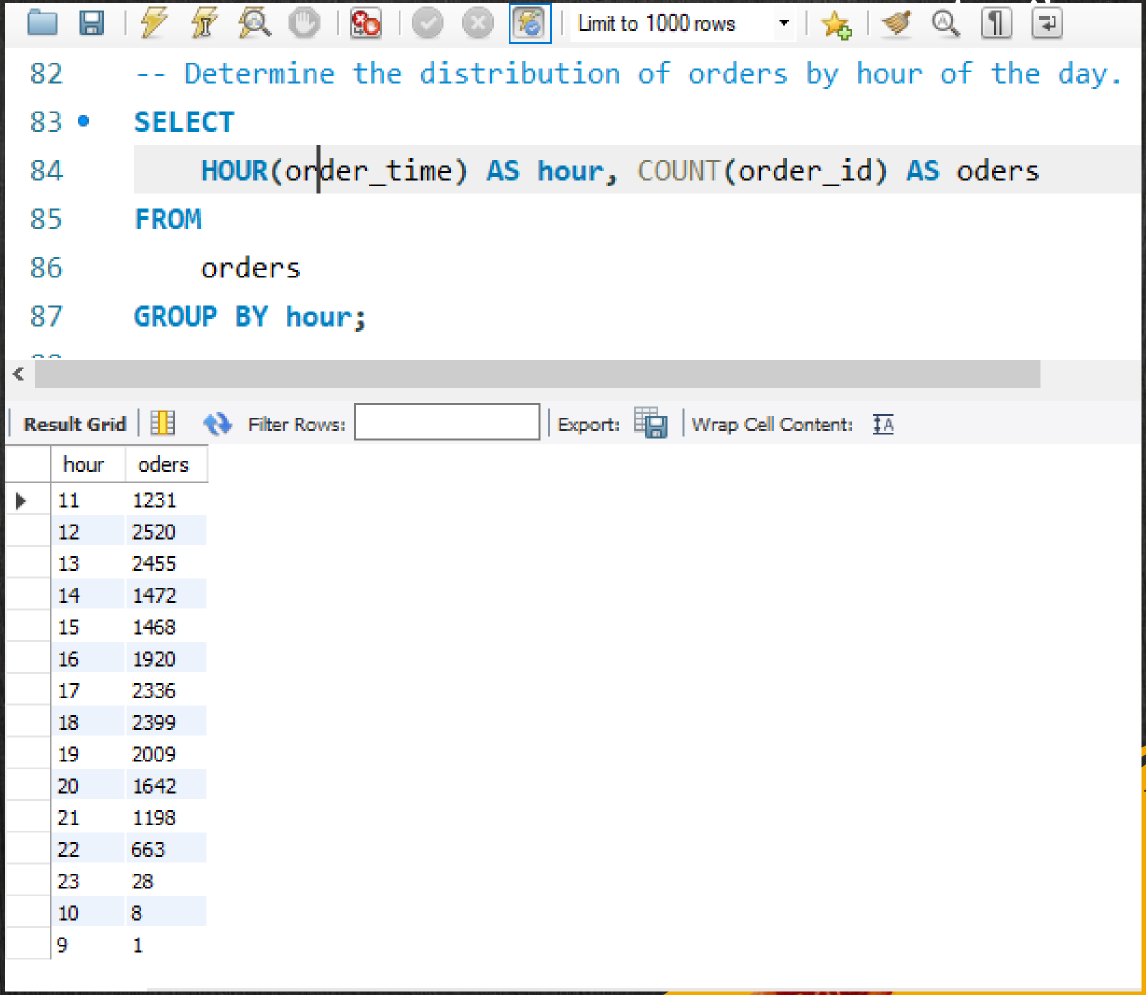
Q6: find the total quantity of each pizza category ordered.



```
-- Join the necessary tables to find the total quantity of each pizza category ordered.
SELECT
    category, SUM(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 category
ORDER BY total_quantity DESC;
```

category	total_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

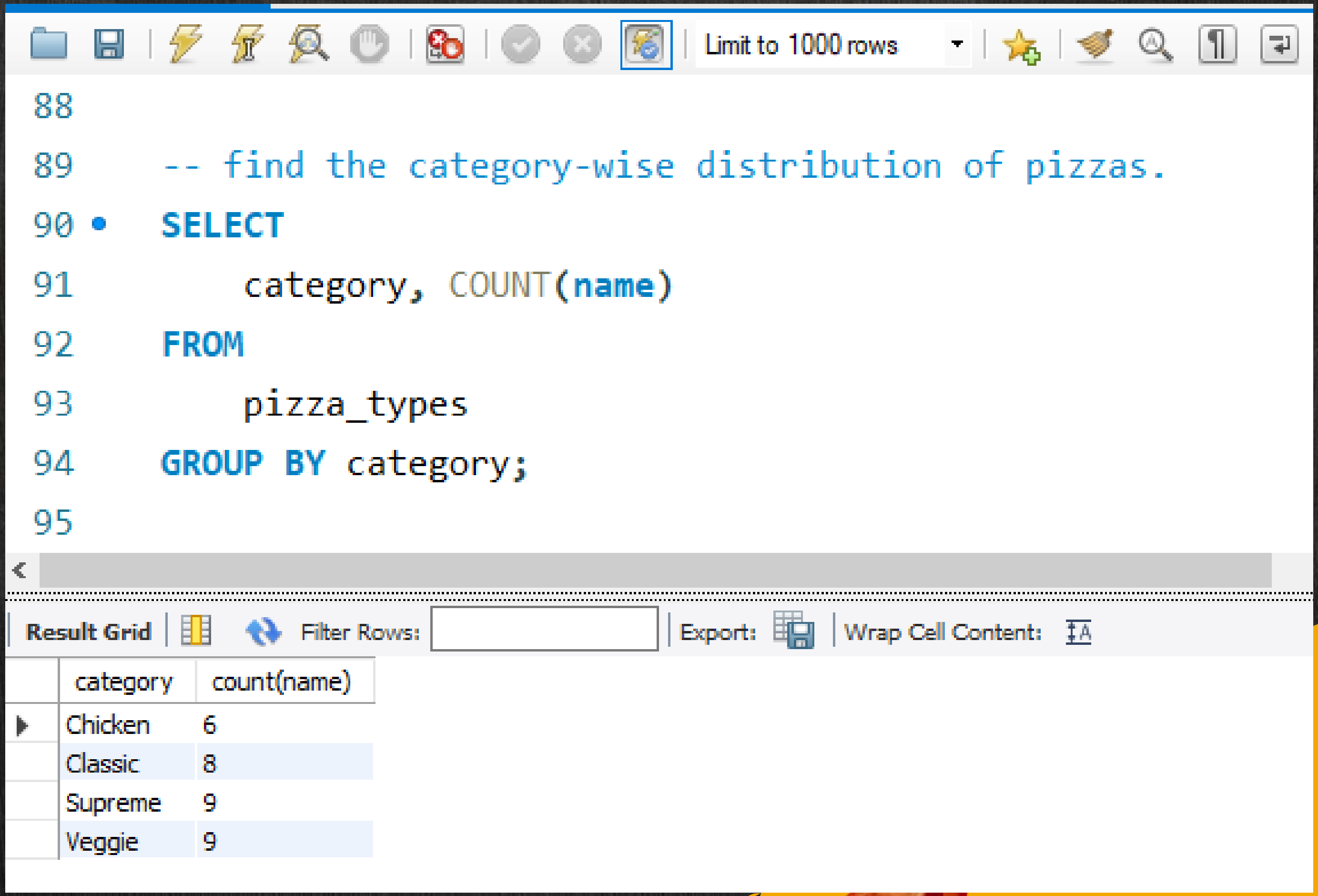
Q7: Determine the distribution of orders by hour of the day.



```sql
-- Determine the distribution of orders by hour of the day.
SELECT
    HOUR(order_time) AS hour, COUNT(order_id) AS oders
FROM
    orders
GROUP BY hour;
```

hour	oders
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
10	8
9	1

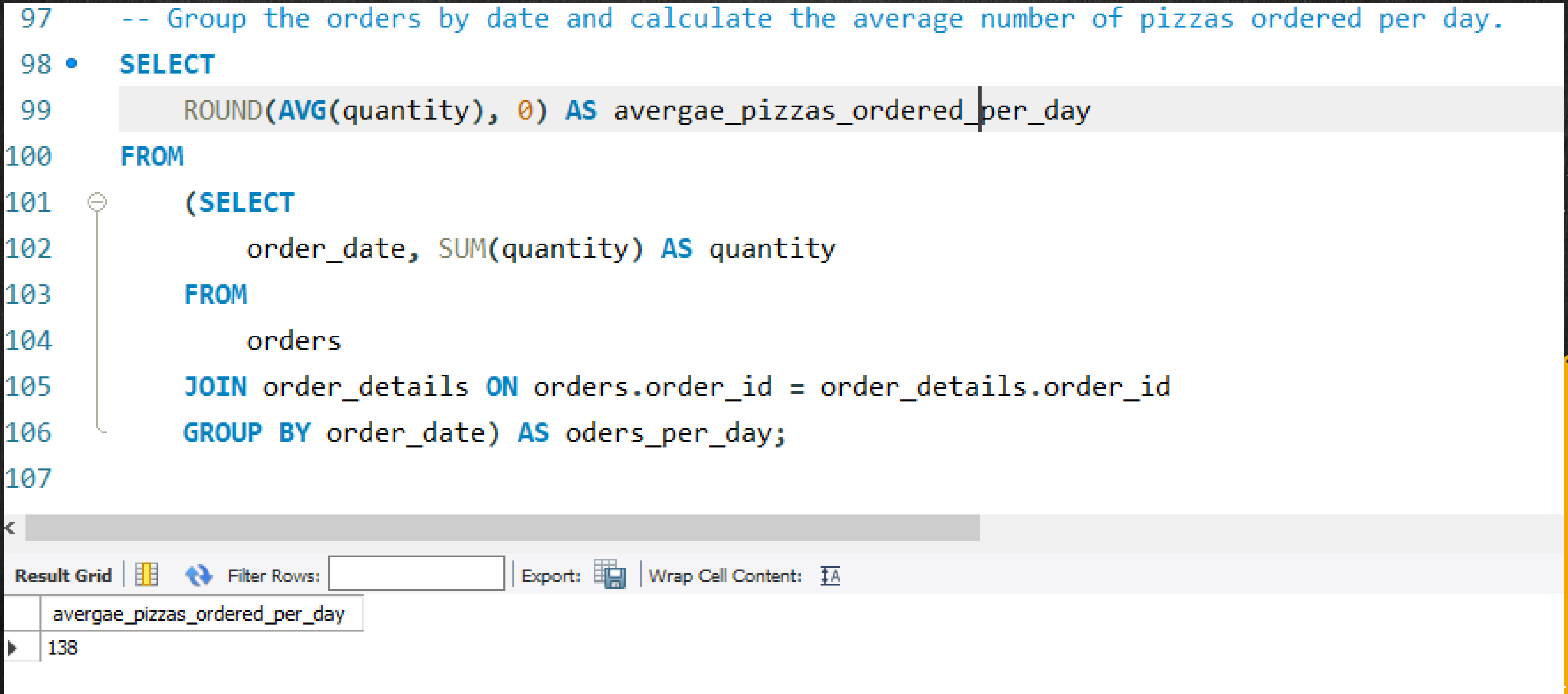
Q8: find the category-wise distribution of pizzas.



```sql
-- find the category-wise distribution of pizzas.
SELECT
    category, COUNT(name)
FROM
    pizza_types
GROUP BY category;
```

category	count(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

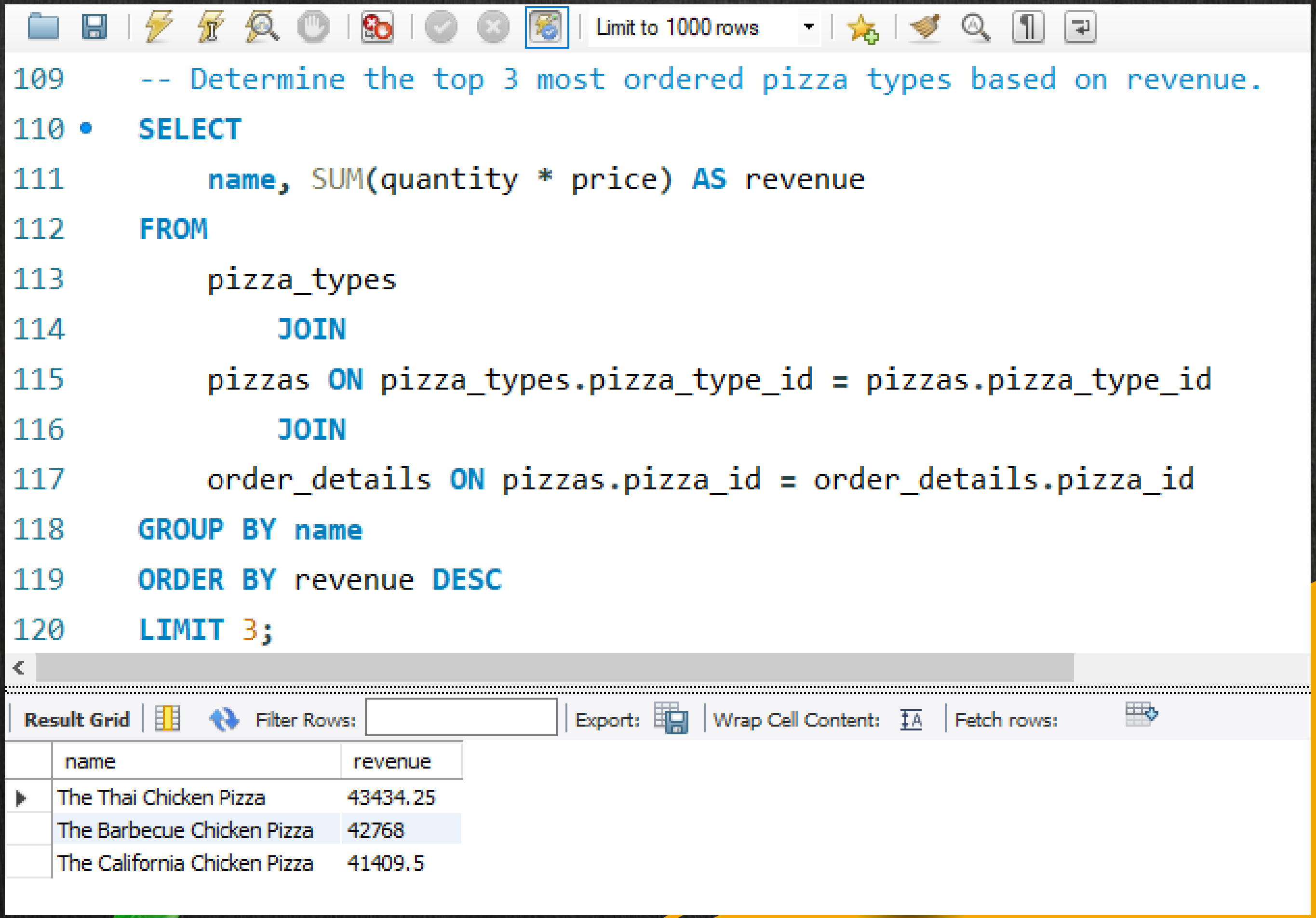
Q9: Group the orders by date and calculate the average number of pizzas ordered per day.



```sql
-- Group the orders by date and calculate the average number of pizzas ordered per day.
SELECT
    ROUND(AVG(quantity), 0) AS avergae_pizzas_ordered_per_day
FROM
    (SELECT
        order_date, SUM(quantity) AS quantity
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY order_date) AS oders_per_day;
```

```
avergae_pizzas_ordered_per_day
138
```

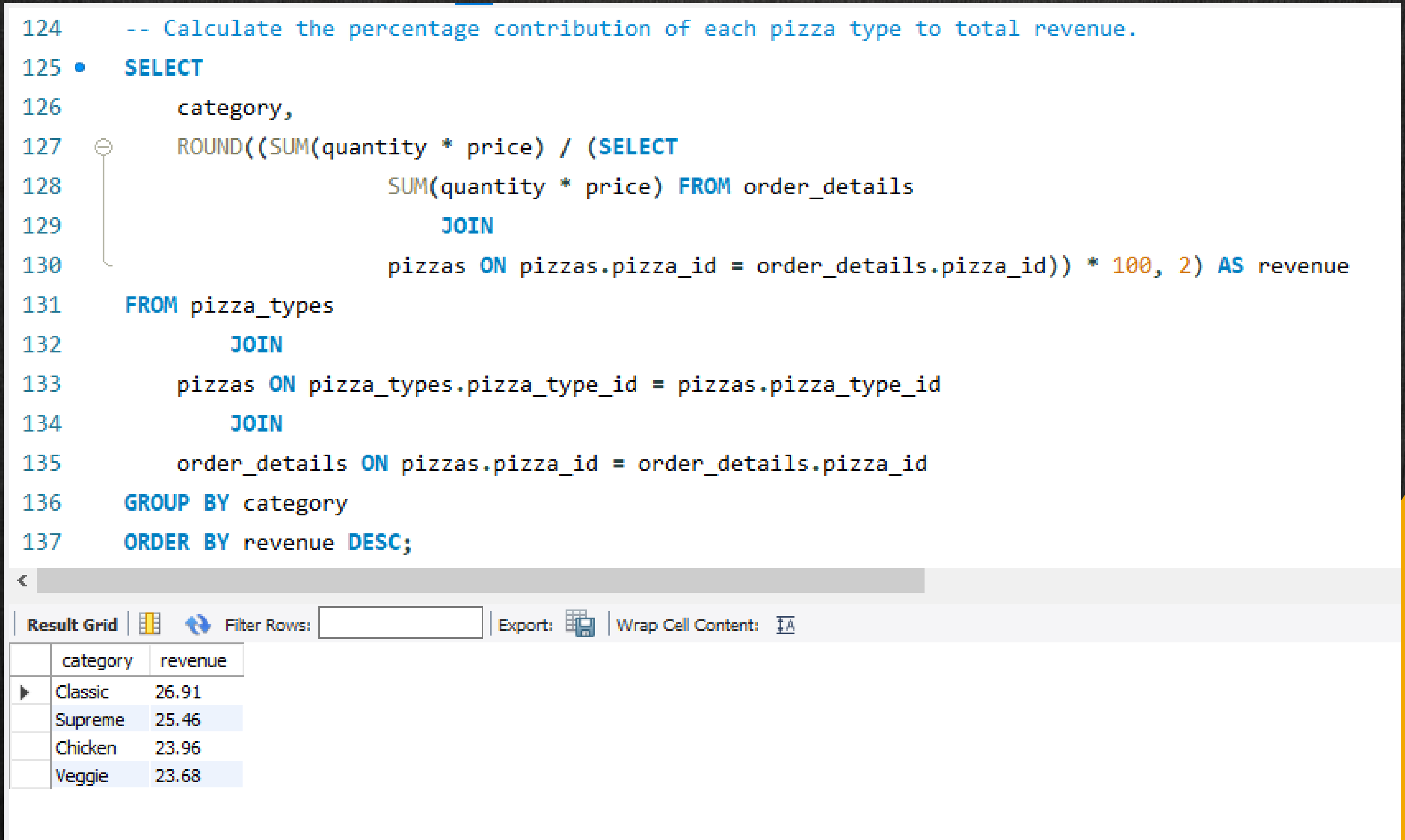
Q10: Determine the top 3 most ordered pizza types based on revenue.ieve the total number of orders placed.



```sql
-- Determine the top 3 most ordered pizza types based on revenue.
SELECT
    name, SUM(quantity * price) AS revenue
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 name
ORDER BY revenue DESC
LIMIT 3;
```

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

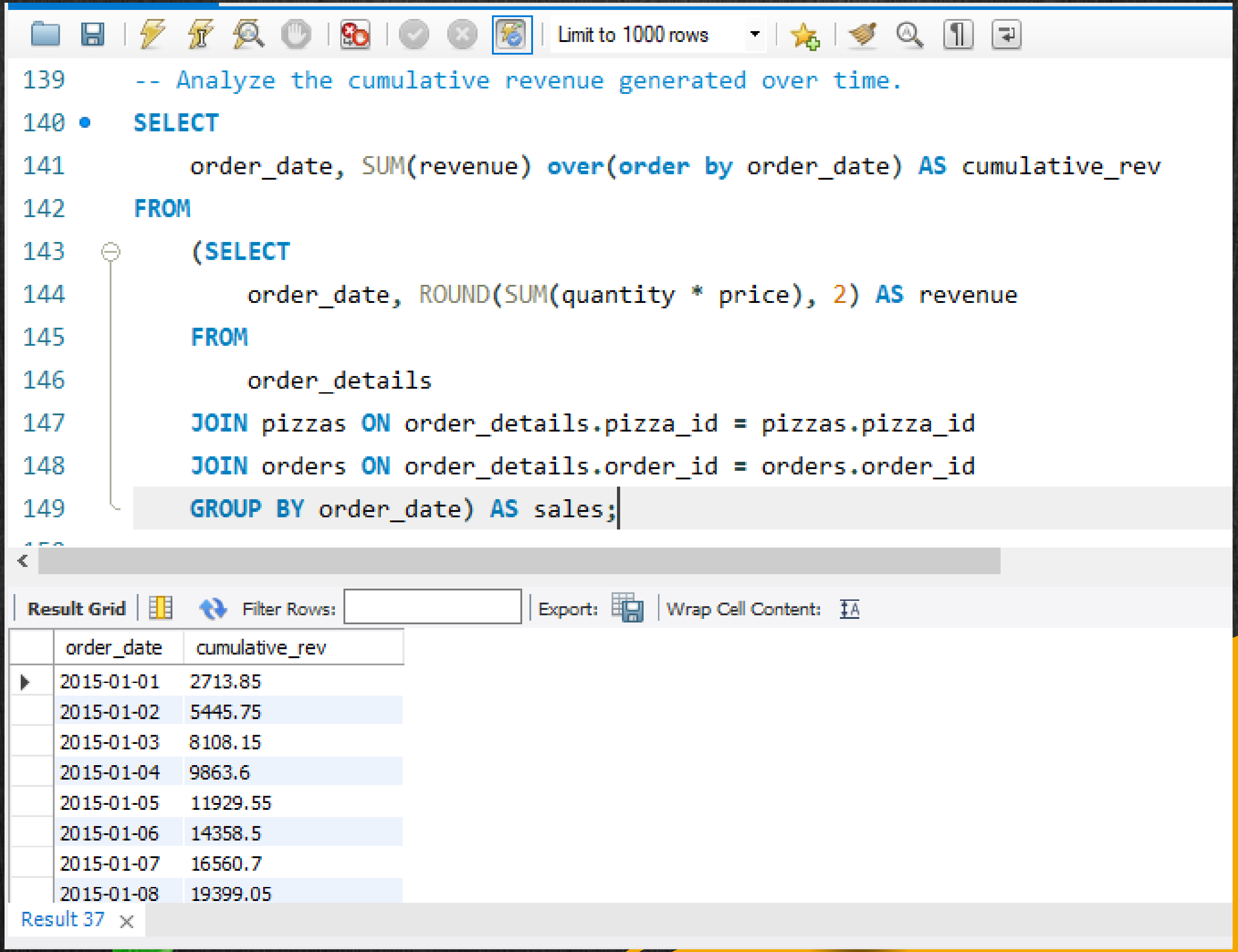
Q11: Calculate the percentage contribution of each pizza type to total revenue.



```sql
-- Calculate the percentage contribution of each pizza type to total revenue.
SELECT
    category,
    ROUND((SUM(quantity * price) / (SELECT
                    SUM(quantity * price) 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 pizzas.pizza_id = order_details.pizza_id
GROUP BY category
ORDER BY revenue DESC;
```

```
category   revenue
Classic    26.91
Supreme    25.46
Chicken    23.96
Veggie     23.68
```

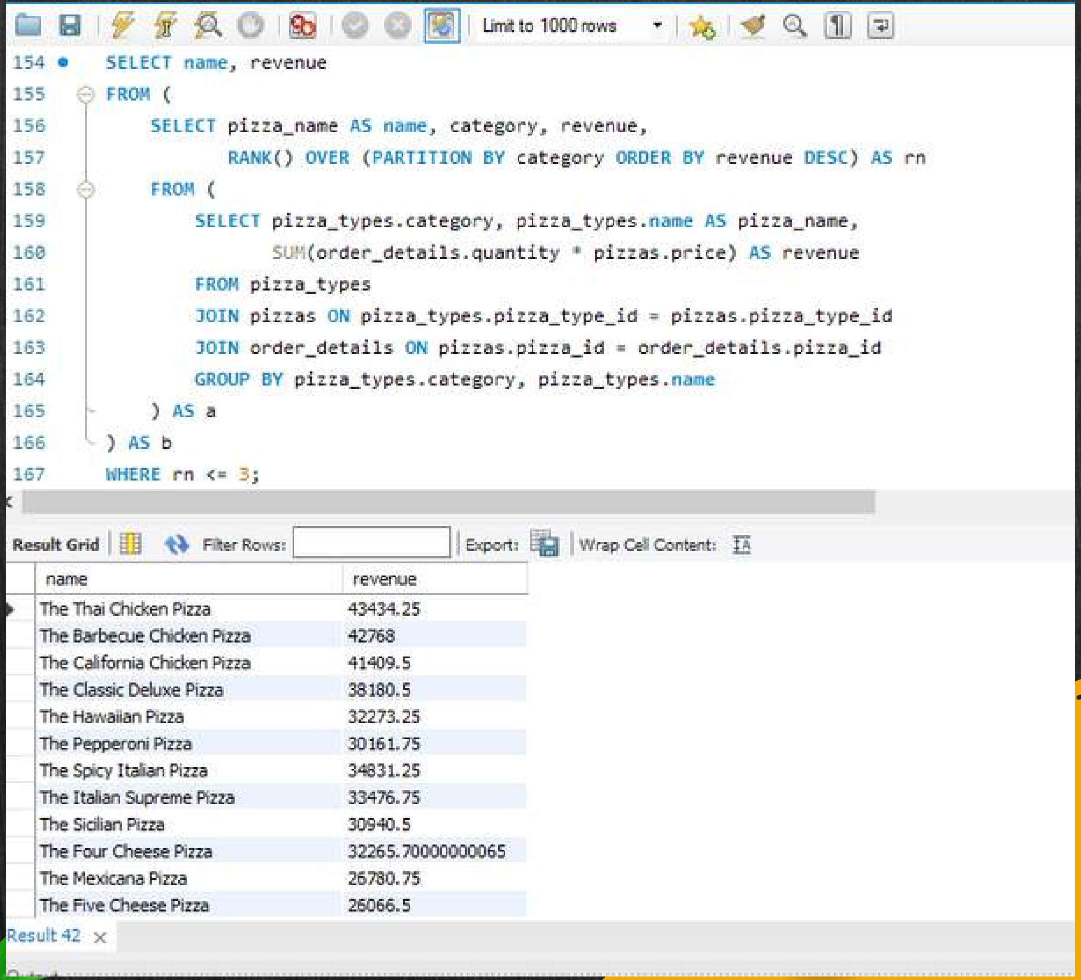
Q12: Analyze the cumulative revenue generated over time.



```sql
-- Analyze the cumulative revenue generated over time.
SELECT
    order_date, SUM(revenue) over(order by order_date) AS cumulative_rev
FROM
    (SELECT
        order_date, ROUND(SUM(quantity * price), 2) AS revenue
    FROM
        order_details
    JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
    JOIN orders ON order_details.order_id = orders.order_id
    GROUP BY order_date) AS sales;
```

order_date	cumulative_rev
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
2015-01-06	14358.5
2015-01-07	16560.7
2015-01-08	19399.05

Q13: Determine the top 3 most ordered pizza types based on revenue for each pizza category.



```sql
SELECT name, revenue
FROM (
    SELECT pizza_name AS name, category, revenue,
        RANK() OVER (PARTITION BY category ORDER BY revenue DESC) AS rn
    FROM (
        SELECT pizza_types.category, pizza_types.name AS pizza_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 pizzas.pizza_id = order_details.pizza_id
        GROUP BY pizza_types.category, pizza_types.name
    ) AS a
) AS b
WHERE rn <= 3;
```

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
The Classic Deluxe Pizza	38180.5
The Hawaiian Pizza	32273.25
The Pepperoni Pizza	30161.75
The Spicy Italian Pizza	34831.25
The Italian Supreme Pizza	33476.75
The Sicilian Pizza	30940.5
The Four Cheese Pizza	32265.70000000065
The Mexicana Pizza	26780.75
The Five Cheese Pizza	26066.5

THE END