

# *SQL Project Pizza Sales*



MYSQL®

# Who I Am

---

Hi, I'm Nithin, an aspiring data analyst with a strong interest in turning data into actionable insights. I have hands-on experience working with SQL, including a project where I analyzed pizza sales data to uncover trends, optimize performance, and support data-driven decision-making. I'm eager to continue building my analytical skills and contribute to real-world business problems.

mySQL®

I have worked on SQL questions across three levels: basic, intermediate, and advanced.

### **Basic:**

- Retrieve the total number of orders placed.
- Calculate the total revenue generated from pizza sales.
- Identify the highest-priced pizza.
- Identify the most common pizza size ordered.
- List the top 5 most ordered pizza types along with their quantities.

### **Intermediate:**

- Join the necessary tables to find the total quantity of each pizza category ordered.
- Determine the distribution of orders by hour of the day.
- Join relevant tables to find the category-wise distribution of pizzas.
- Group the orders by date and calculate the average number of pizzas ordered per day.
- Determine the top 3 most ordered pizza types based on revenue.

### **Advanced:**

- Calculate the percentage contribution of each pizza type to total revenue.
- Analyze the cumulative revenue generated over time.
- Determine the top 3 most ordered pizza types based on revenue for each pizza category.

Q1.Retrieve the total number of orders placed.

The screenshot shows a SQL query editor interface. The query is as follows:

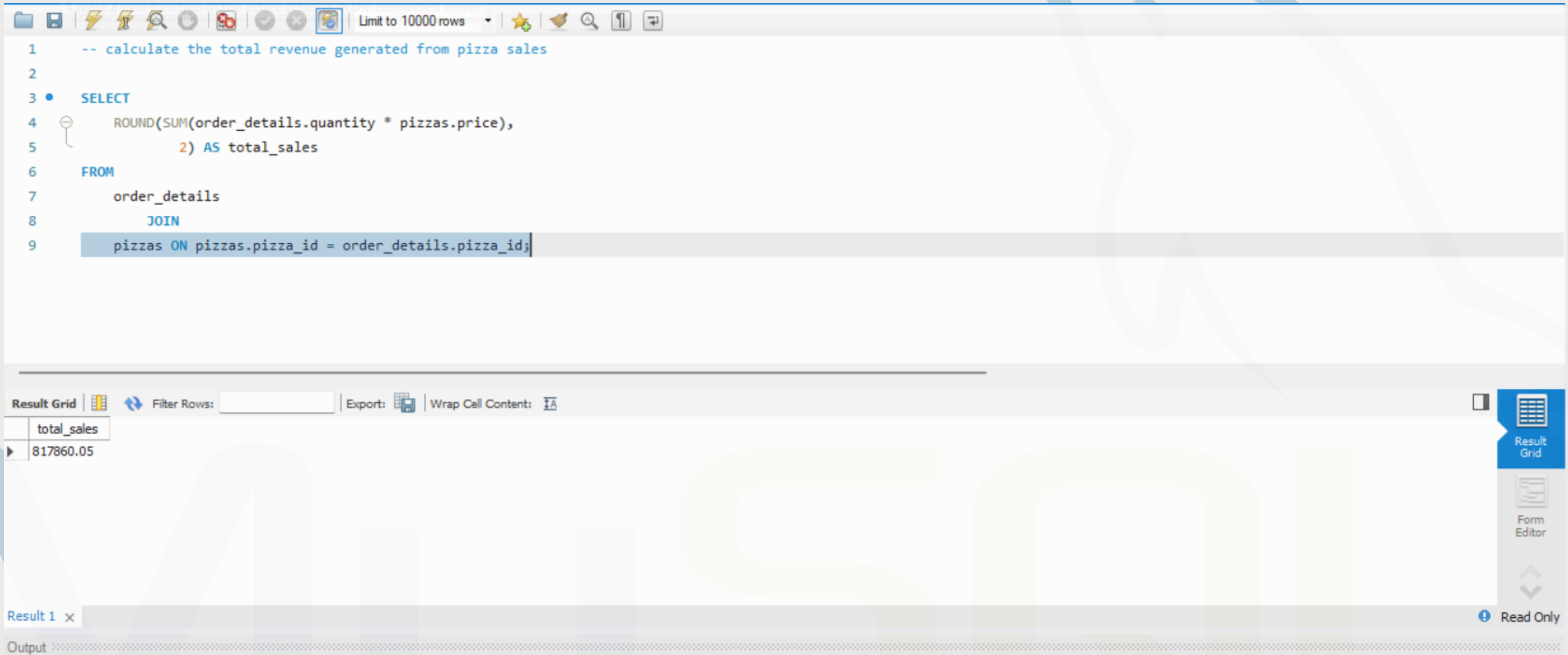
```
1  -- Retrieve the total number of orders placed.  
2  
3  •  Select count(order_id) as total_orders from orders;
```

The result grid shows the following data:

total_orders
21350

The interface includes a toolbar at the top with various icons, a status bar at the bottom indicating "Limit to 10000 rows", and a sidebar on the right with options for "Result Grid", "Form Editor", and "Read Only".

Q2. Calculate the total revenue generated from pizza sales?



The screenshot shows a SQL query editor interface. The query is as follows:

```
1  -- calculate the total revenue generated from pizza sales
2
3  SELECT
4  ROUND(SUM(order_details.quantity * pizzas.price),
5        2) AS total_sales
6  FROM
7  order_details
8  JOIN
9  pizzas ON pizzas.pizza_id = order_details.pizza_id;
```

Below the query editor, the 'Result Grid' is visible, showing the following data:

total_sales
817860.05

The interface also includes a toolbar at the top with various icons, a 'Limit to 10000 rows' dropdown, and a 'Filter Rows' input field. On the right side, there are buttons for 'Result Grid', 'Form Editor', and 'Read Only'.

### Q3. Identify the highest priced pizza?

The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

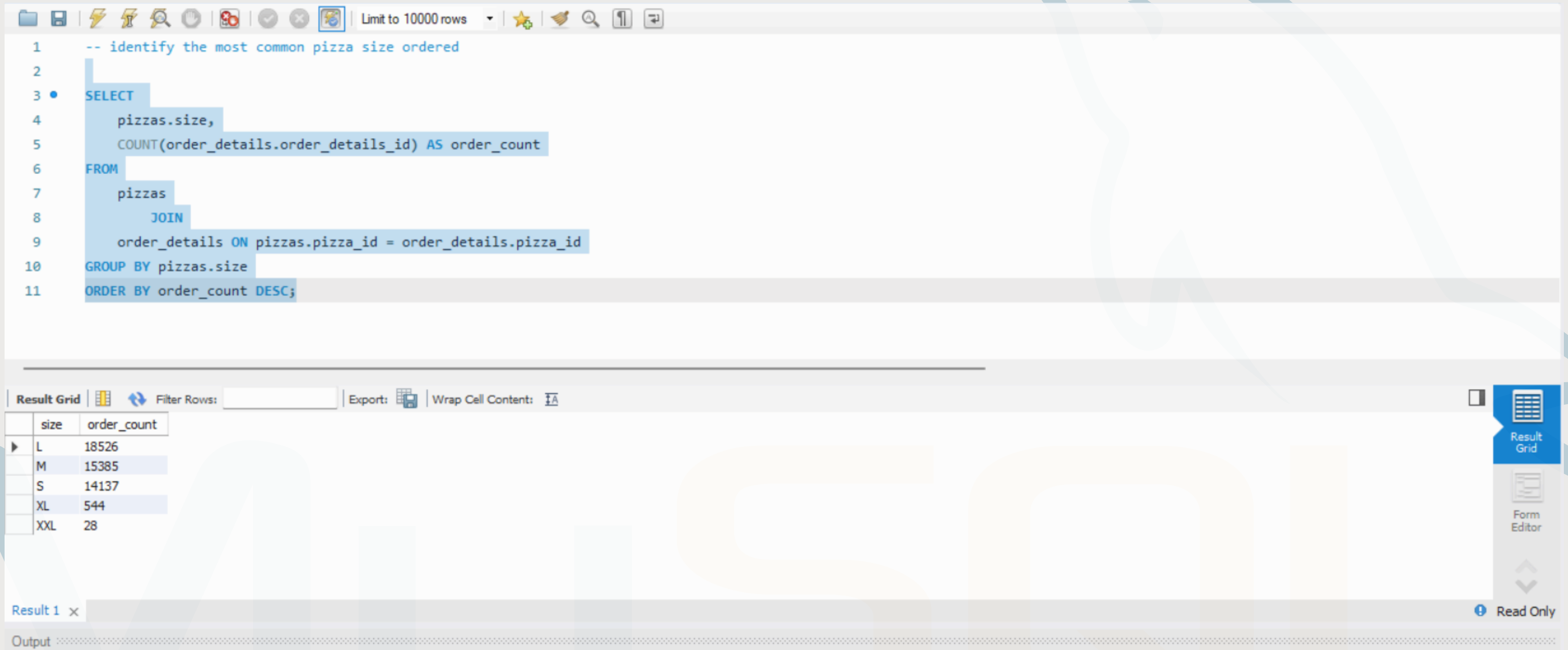
```
1  -- identify the highest - priced pizza.
2
3  • SELECT
4      pizza_types.name, pizzas.price
5  FROM
6      pizza_types
7      JOIN
8      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
9  ORDER BY pizzas.price DESC
10 LIMIT 1;
```

Below the query editor, the results are displayed in a table with columns 'name' and 'price'. The result shows 'The Greek Pizza' with a price of 35.95.

name	price
The Greek Pizza	35.95

The interface includes a 'Result Grid' tab, a 'Filter Rows' input, and an 'Export' button. A 'Form Editor' tab is also visible on the right. The bottom of the window shows a 'Result 1' tab and an 'Output' section.

## Q4. Identify the most common pizza ordered?



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 10000 rows' dropdown. The SQL editor contains the following query:

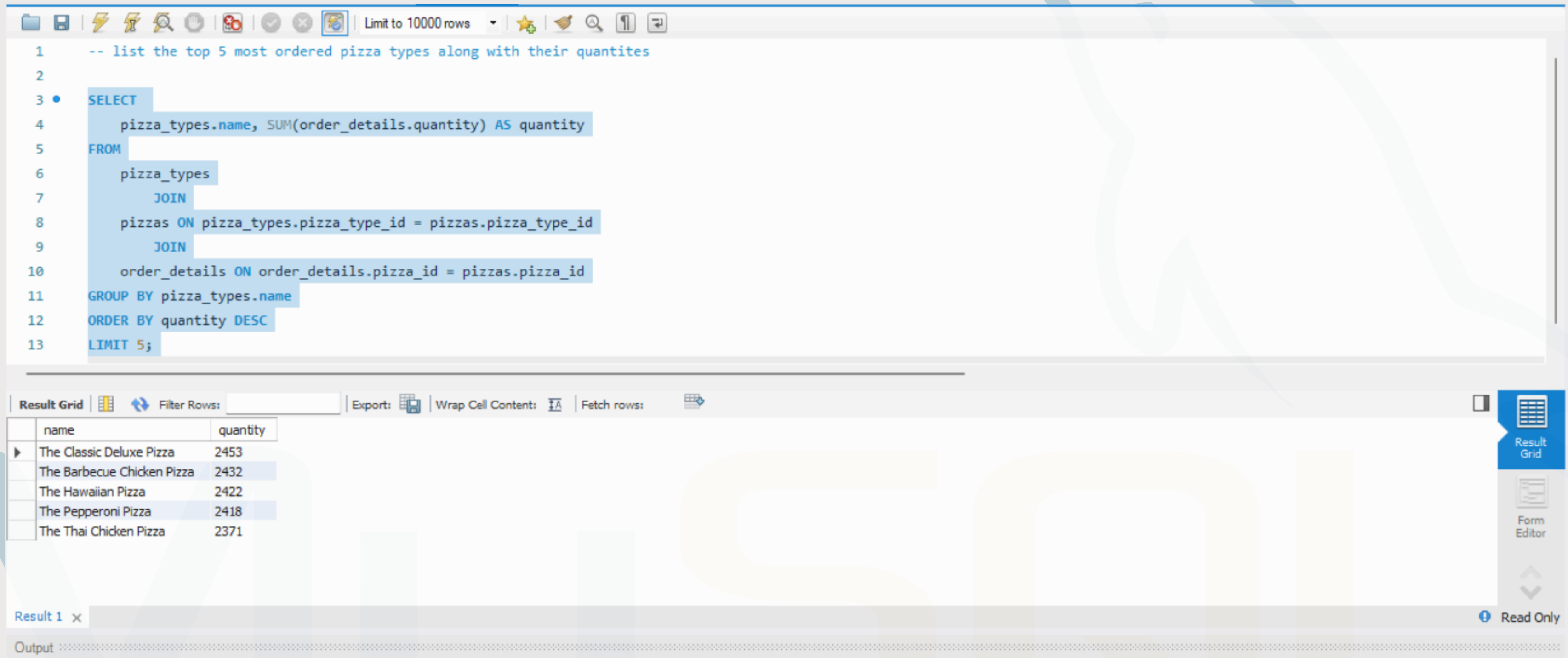
```
1  -- identify the most common pizza size ordered
2
3  SELECT
4      pizzas.size,
5      COUNT(order_details.order_details_id) AS order_count
6  FROM
7      pizzas
8      JOIN
9      order_details ON pizzas.pizza_id = order_details.pizza_id
10 GROUP BY pizzas.size
11 ORDER BY order_count DESC;
```

Below the editor is the 'Result Grid' section, which includes a 'Filter Rows' input, 'Export' and 'Wrap Cell Content' buttons, and a table of results. The table has two columns: 'size' and 'order\_count'. The results are as follows:

size	order_count
L	18526
M	15385
S	14137
XL	544
XXL	28

On the right side of the interface, there are buttons for 'Result Grid' and 'Form Editor'. At the bottom, there is a 'Read Only' indicator and an 'Output' section.

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



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 10000 rows' dropdown. The SQL editor contains the following query:

```
1  -- list the top 5 most ordered pizza types along with their quantities
2
3  • SELECT
4      pizza_types.name, SUM(order_details.quantity) AS quantity
5  FROM
6      pizza_types
7      JOIN
8      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
9      JOIN
10     order_details ON order_details.pizza_id = pizzas.pizza_id
11 GROUP BY pizza_types.name
12 ORDER BY quantity DESC
13 LIMIT 5;
```

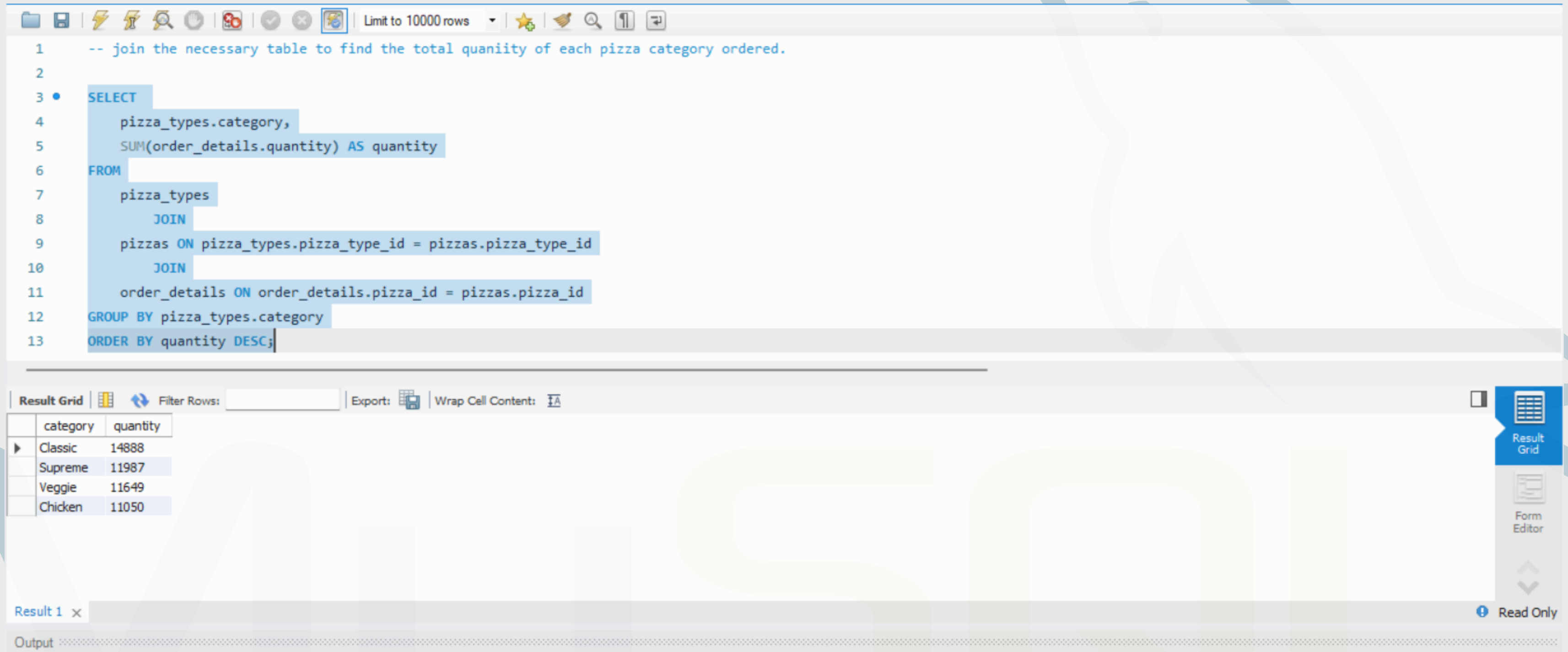
Below the editor is the 'Result Grid' tab, which displays the query results in a table:

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

On the right side of the interface, there are buttons for 'Result Grid', 'Form Editor', and a 'Read Only' status indicator. The bottom status bar shows 'Output' and 'Result 1 x'.



Q6. Join the necessary table to find the total quantity of each pizza category ordered.



```
1  -- join the necessary table to find the total quantity of each pizza category ordered.
2
3  SELECT
4      pizza_types.category,
5      SUM(order_details.quantity) AS quantity
6  FROM
7      pizza_types
8      JOIN
9      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
10     JOIN
11     order_details ON order_details.pizza_id = pizzas.pizza_id
12 GROUP BY pizza_types.category
13 ORDER BY quantity DESC;
```

Result Grid

	category	quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

Result 1 x

Output

Read Only

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

Limit to 10000 rows

```
1  -- determine the distribution of orders by hour of the day
2
3  • SELECT
4      HOUR(order_time) AS hour, COUNT(order_id) AS order_count
5  FROM
6      orders
7  GROUP BY HOUR(order_time);
```

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

	hour	order_count
▶	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

Result 1 x | Read Only

Output

Q8. Join relevant tables to find the category wise distribution of pizza.

The screenshot displays a SQL query editor interface. The query text is as follows:

```
1  -- join relevent tables to find the category wise distribution of pizza
2
3  • select category, count(name) from pizza_types
4  group by category;
```

Below the query editor, the 'Result Grid' tab is active, showing the following data:

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

The interface includes a toolbar at the top with icons for file operations, a 'Limit to 10000 rows' dropdown, and a sidebar on the right with options like 'Result Grid', 'Form Editor', 'Field Types', and 'Query Stats'. The bottom status bar indicates 'Result 1' and 'Read Only'.

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

The screenshot shows a database query editor with a SQL query and its results. The query is as follows:

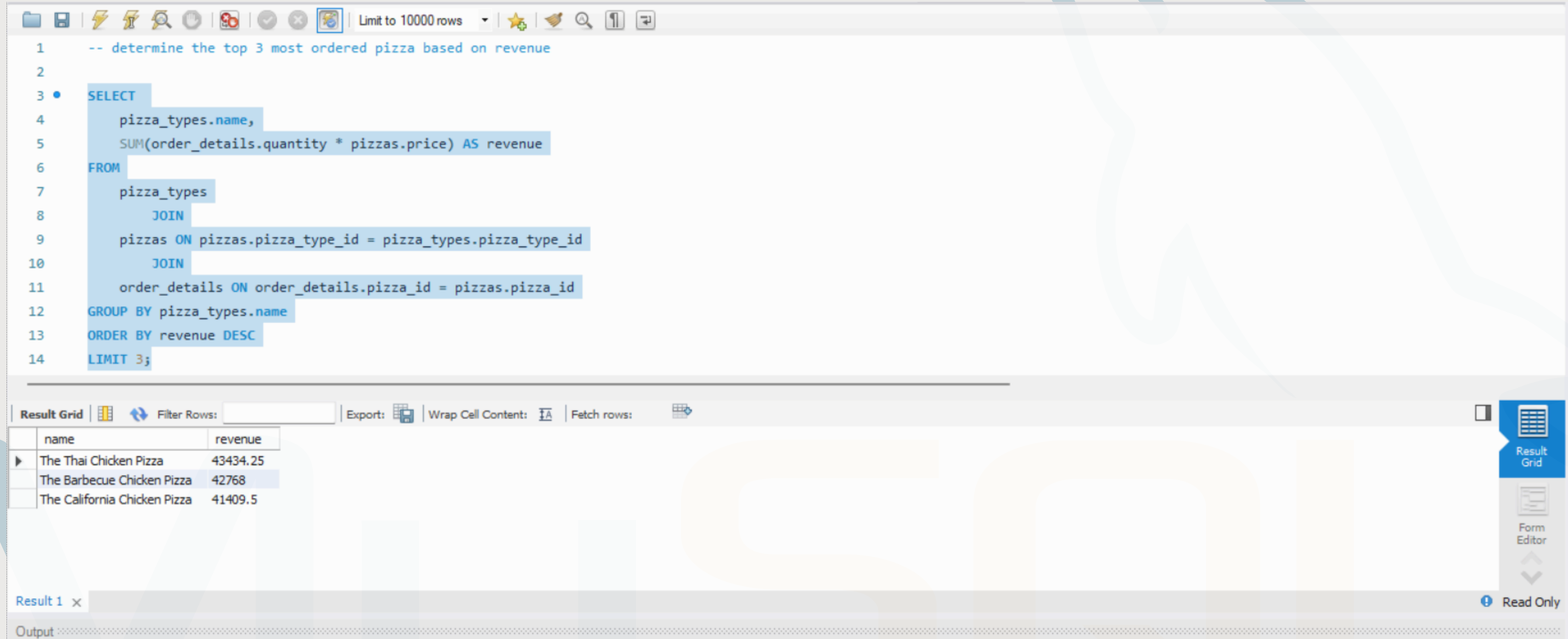
```
1  -- group the orders by date and calculate the average number of pizzas ordered per day
2
3  SELECT
4      ROUND(AVG(quantity), 0)
5  FROM
6      (SELECT
7          orders.order_date, SUM(order_details.quantity) AS quantity
8      FROM
9          orders
10         JOIN order_details ON orders.order_id = order_details.order_id
11        GROUP BY orders.order_date) AS order_quantity;
```

The results pane shows a single row with the value 138.

ROUND(AVG(quantity), 0)
138

The interface includes a toolbar at the top with icons for file operations, a 'Limit to 10000 rows' dropdown, and a 'Result Grid' button. The bottom of the window shows a 'Result 1' tab and an 'Output' pane.

Q10. Determine the top 3 most ordered pizza types based on revenue.



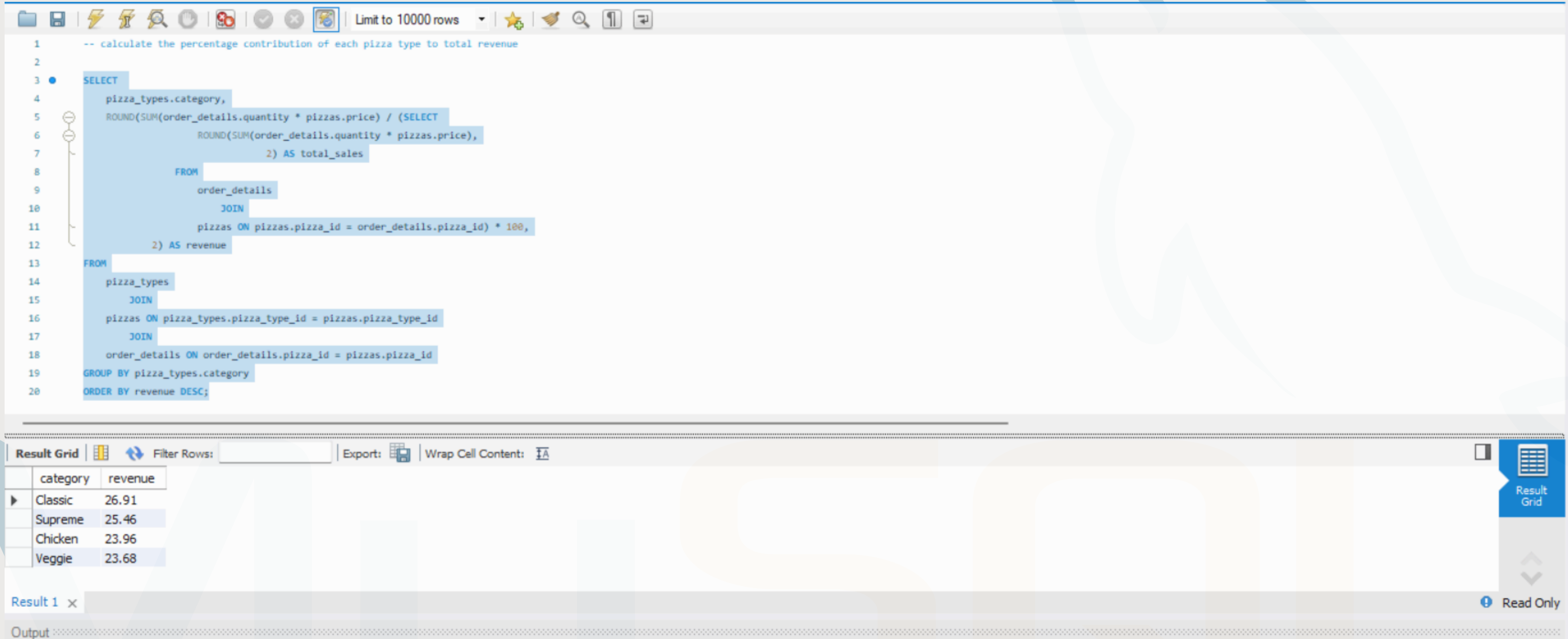
```
1  -- determine the top 3 most ordered pizza based on revenue
2
3  SELECT
4      pizza_types.name,
5      SUM(order_details.quantity * pizzas.price) AS revenue
6  FROM
7      pizza_types
8      JOIN
9      pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
10     JOIN
11     order_details ON order_details.pizza_id = pizzas.pizza_id
12 GROUP BY pizza_types.name
13 ORDER BY revenue DESC
14 LIMIT 3;
```

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

Result 1 x

Output

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



```
1  -- calculate the percentage contribution of each pizza type to total revenue
2
3  SELECT
4      pizza_types.category,
5      ROUND(SUM(order_details.quantity * pizzas.price) / (SELECT
6          ROUND(SUM(order_details.quantity * pizzas.price),
7              2) AS total_sales
8      FROM
9          order_details
10         JOIN
11             pizzas ON pizzas.pizza_id = order_details.pizza_id) * 100,
12          2) AS revenue
13 FROM
14     pizza_types
15     JOIN
16         pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
17     JOIN
18         order_details ON order_details.pizza_id = pizzas.pizza_id
19 GROUP BY pizza_types.category
20 ORDER BY revenue DESC;
```

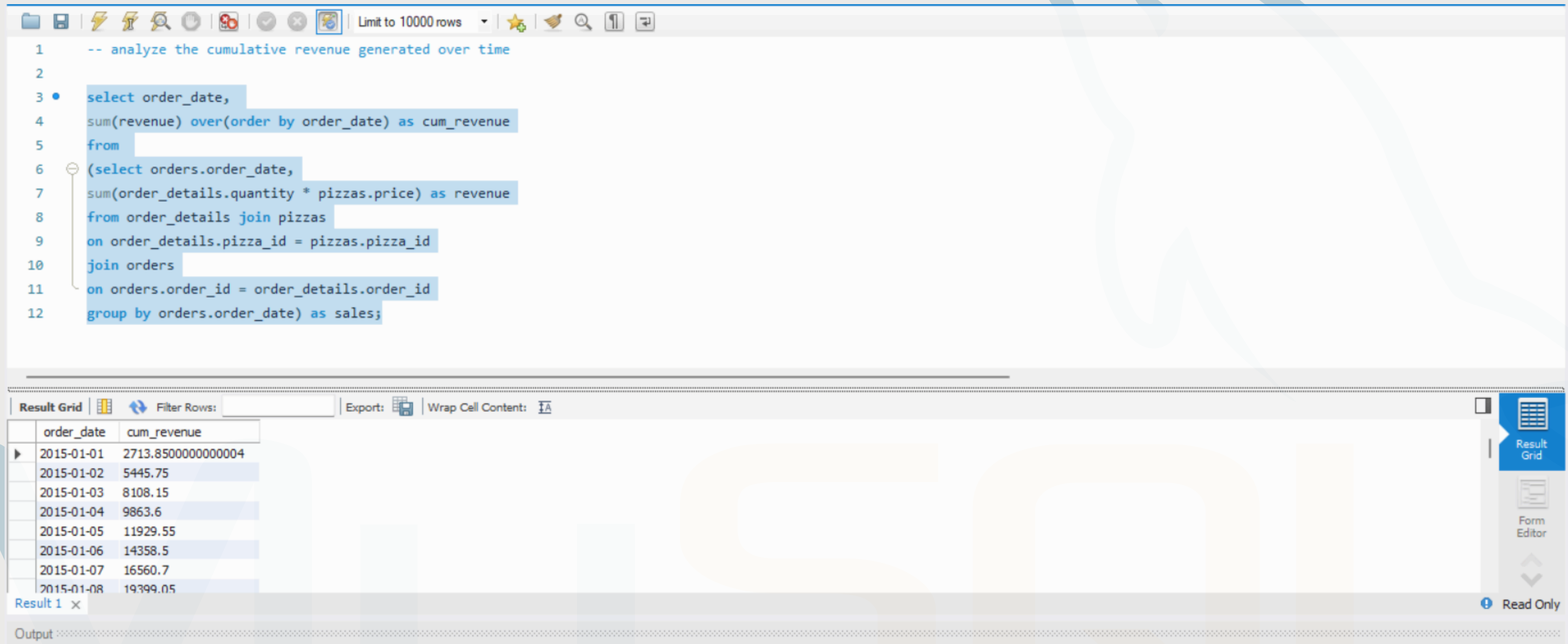
Result Grid

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

Result 1 x

Output

## Q12. Analyze the cumulative revenue generated over time.



The screenshot displays a SQL IDE interface. The top section shows a SQL query designed to calculate cumulative revenue over time. The query uses a subquery to calculate daily revenue from order details and pizzas, then uses a window function to calculate the cumulative revenue over time.

```
1  -- analyze the cumulative revenue generated over time
2
3  select order_date,
4  sum(revenue) over(order by order_date) as cum_revenue
5  from
6  (select orders.order_date,
7  sum(order_details.quantity * pizzas.price) as revenue
8  from order_details join pizzas
9  on order_details.pizza_id = pizzas.pizza_id
10 join orders
11 on orders.order_id = order_details.order_id
12 group by orders.order_date) as sales;
```

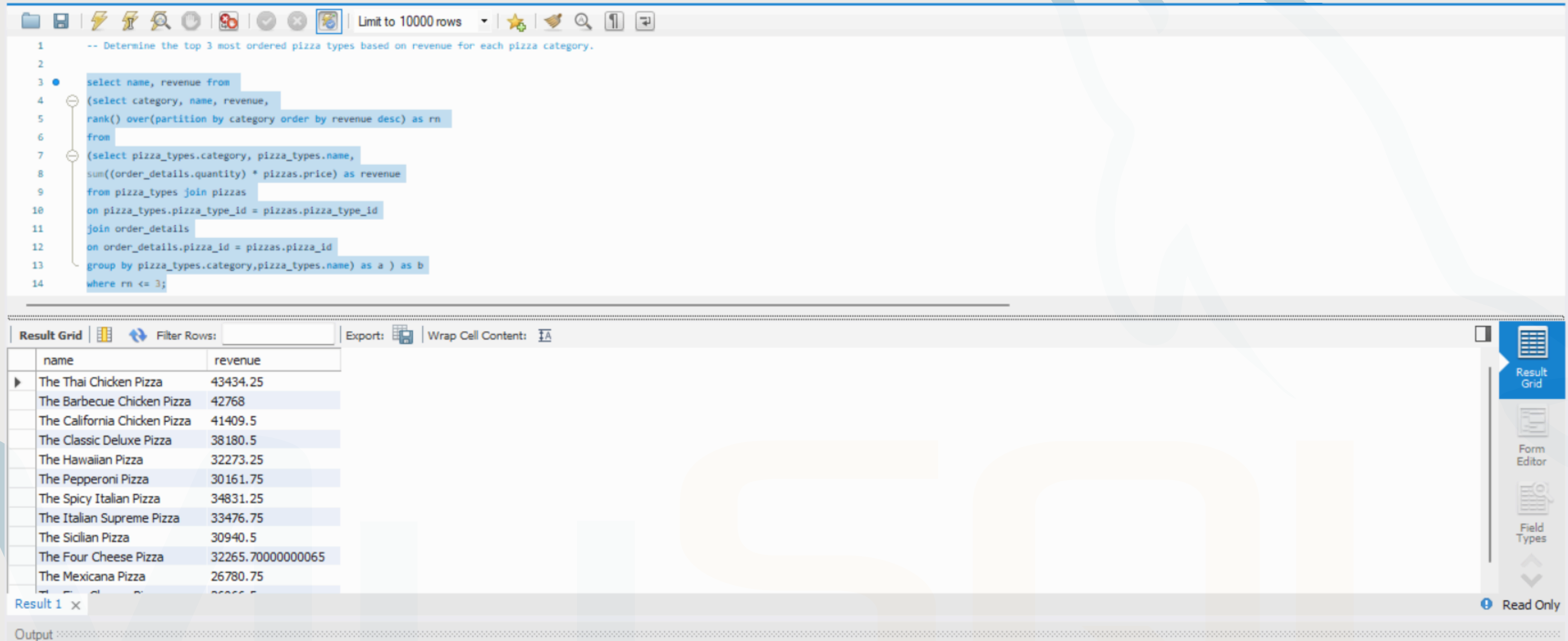
Below the query editor, the 'Result Grid' tab is active, showing the results of the query. The grid has two columns: 'order\_date' and 'cum\_revenue'. The results show a steady increase in cumulative revenue over the first eight days of January 2015.

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

The interface also includes a toolbar at the top with various icons for file operations, a 'Limit to 10000 rows' dropdown, and a 'Read Only' status indicator at the bottom right.



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



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

select name, revenue 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;
```

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

Result 1 x

Output



*The End*

*Thank You*

MySQL<sup>®</sup>

