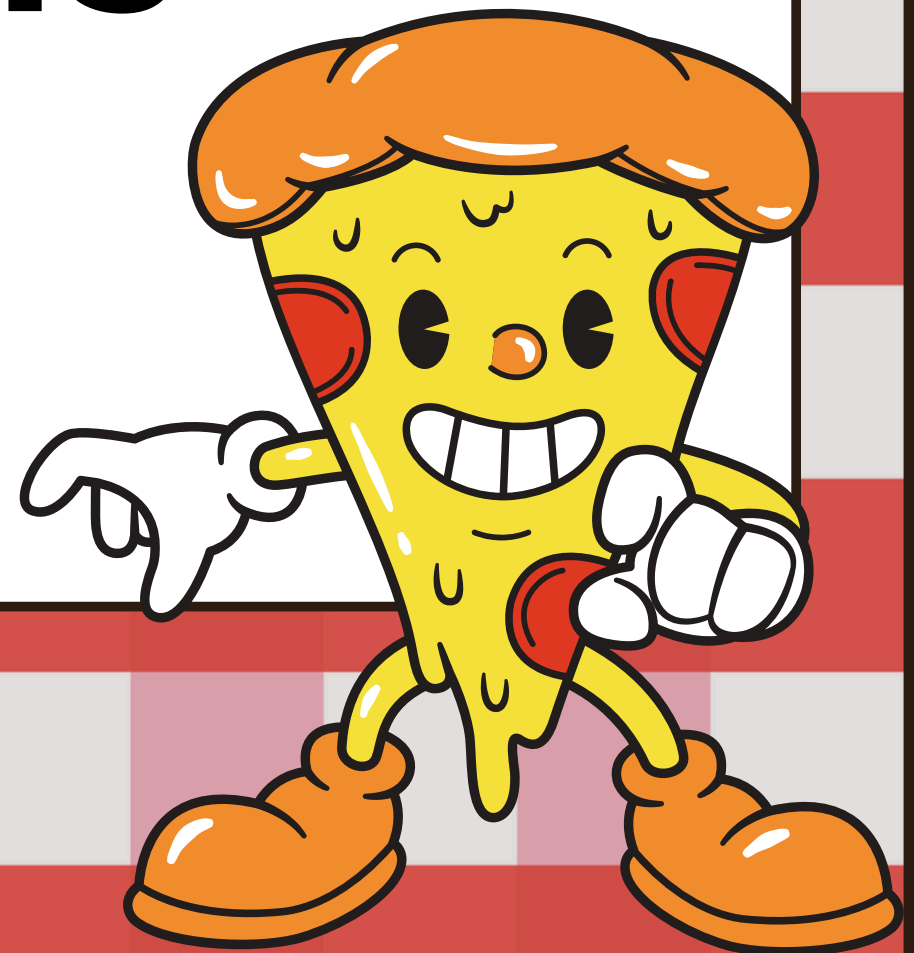
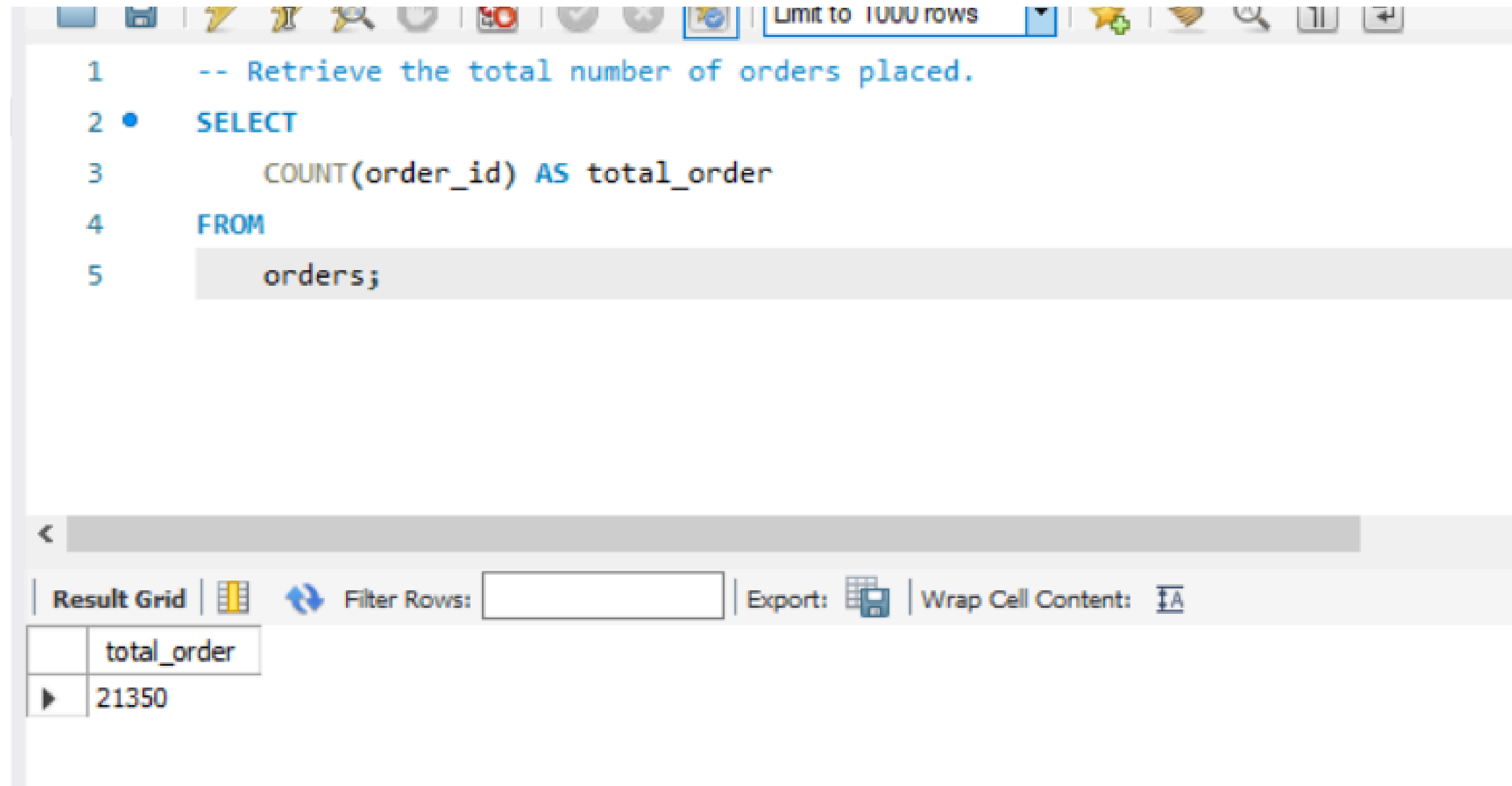


# Pizzahut SQL Analysis



# 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  • SELECT  
3      COUNT(order_id) AS total_order  
4  FROM  
5      orders;
```

Below the query editor, there is a horizontal scrollbar and a toolbar. The toolbar includes a "Result Grid" button, a "Filter Rows" input field, an "Export" button, and a "Wrap Cell Content" button.

The "Result Grid" is displayed below the toolbar, showing the following data:

	total_order
▶	21350

# Calculate the total revenue generated from pizza sales

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

```
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;
```

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

total\_sales


360.05


# Identify the highest-priced pizza.


```
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;
```


<


Result Grid



 Filter Rows:

Export: 

Wrap Cell Content: 





Fetch rows: 

	name	price
▶	The Greek Pizza	35.95

# Identify the most common pizza size ordered

```
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;
```

<

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

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

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





```
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;
```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
	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				

# Join the necessary tables to find the total quantity of each pizza category ordered.

```
1  -- Join the necessary tables to find the total quantity of each pizza category ordered.
2  • SELECT
3      pizza_types.category,
4      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.category
12  ORDER BY quantity DESC;
```

<

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

	category	quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

Result Grid

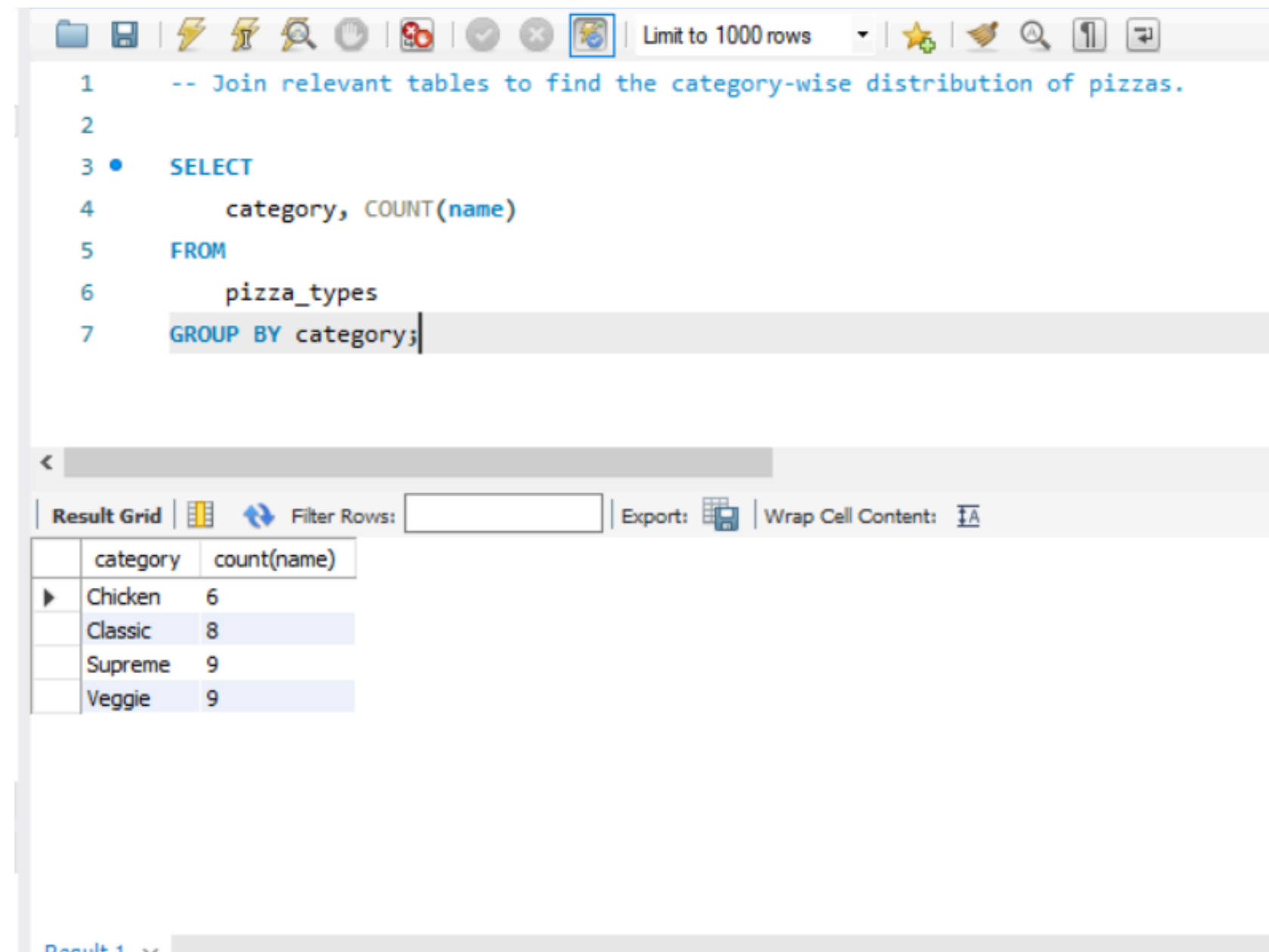
# Determine the distribution of orders by hour of the day.

```
2
3 • select hour(order_time) as hour ,count(order_id) as order_count
4   from orders
5   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			



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



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL editor contains the following query:

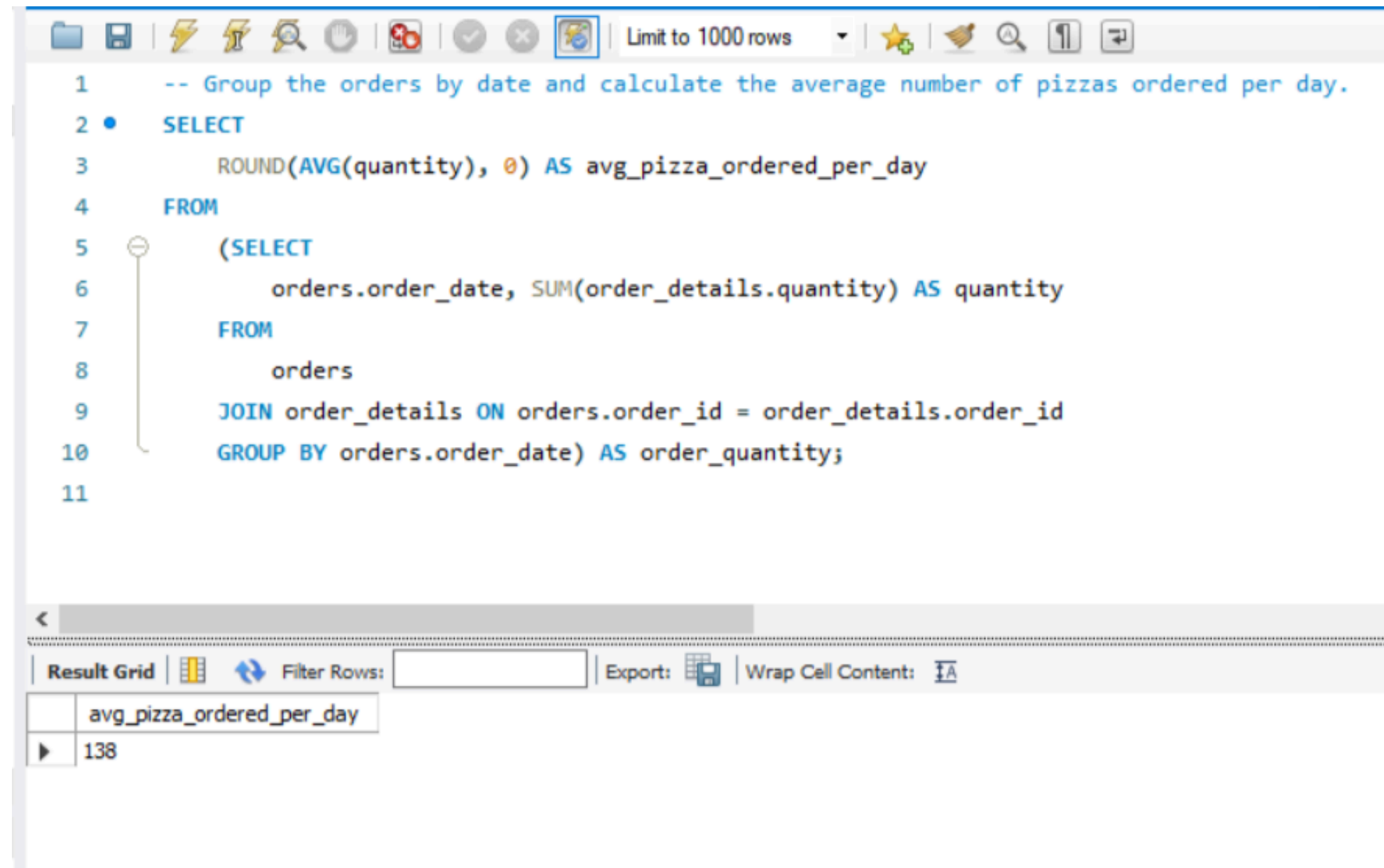
```
1  -- Join relevant tables to find the category-wise distribution of pizzas.
2
3  • SELECT
4      category, COUNT(name)
5  FROM
6      pizza_types
7  GROUP BY category;
```

Below the editor, the 'Result Grid' tab is active, displaying the query results in a table:

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

At the bottom left, 'Result 1' is indicated.

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



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL editor contains the following query:

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

Below the editor is a 'Result Grid' section with a horizontal scrollbar. It contains a single row with the column name 'avg\_pizza\_ordered\_per\_day' and the value '138'.

avg_pizza_ordered_per_day
138

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

```
1  -- Determine the top 3 most ordered pizza types 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 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.name
13 ORDER BY revenue DESC limit 3;
```

<

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Fetch rows:

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

# Thank you

