



# PIZZA SALES REPORT →

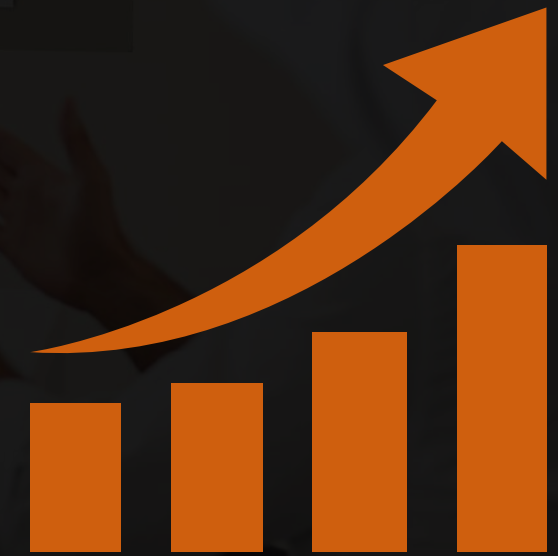
20  
24





# PROJECT OVERVIEW

This SQL project delves into the world of pizza sales data, utilizing SQL queries to extract valuable insights. By employing various SQL techniques, including basic to advanced queries, the project aims to uncover trends, patterns, and relationships within the dataset. The project is organized into 13 distinct queries, each designed to answer a specific question. The queries progressively increase in complexity, starting with basic data retrieval and moving towards more advanced analytical techniques.



# KEY OBJECTIVES

20  
24

- **Data Exploration:** Thoroughly explored the pizza sales dataset to understand its structure, content, and potential insights.
- **Query Development:** Craft a series of SQL queries to address specific questions related to sales, products, customers, and other relevant aspects.
- **Data Analysis:** Analyze the query results to identify trends, patterns, and correlations within the data.
- **Insight Generation:** Derive meaningful conclusions and insights based on the analysis, providing valuable information for business decision-making.



03





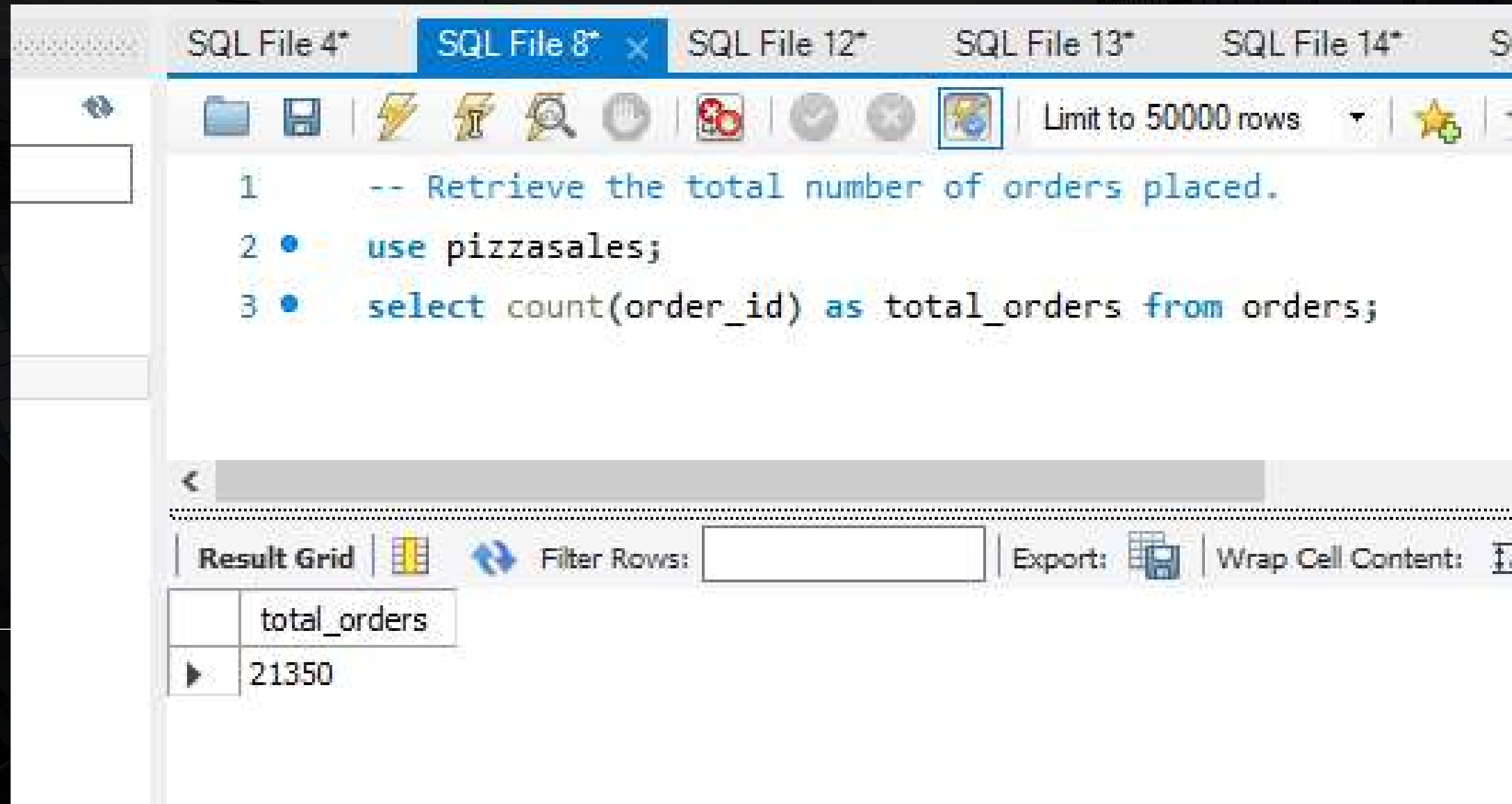
# KEY INSIGHTS

The key insights from the analysis are that the total revenue is \$ 817860.05, 'The Greek Pizza' is the highest priced, large size pizzas are the most ordered, classic pizzas have the highest sales, and the peak sales hour is between 11 AM and 4 PM.



# RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

20  
24



The screenshot shows a SQL IDE with multiple tabs. The active tab is 'SQL File 8\*'. The query editor contains the following SQL code:

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

Below the query editor, the 'Result Grid' is visible, showing the output of the query:

	total_orders
▶	21350

The IDE interface includes a toolbar with various icons for file operations, execution, and settings. A 'Limit to 50000 rows' dropdown is also visible in the toolbar.

05



# CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

20  
24

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

Result Grid



Filter Rows:

Export:



Wrap Cell Content:



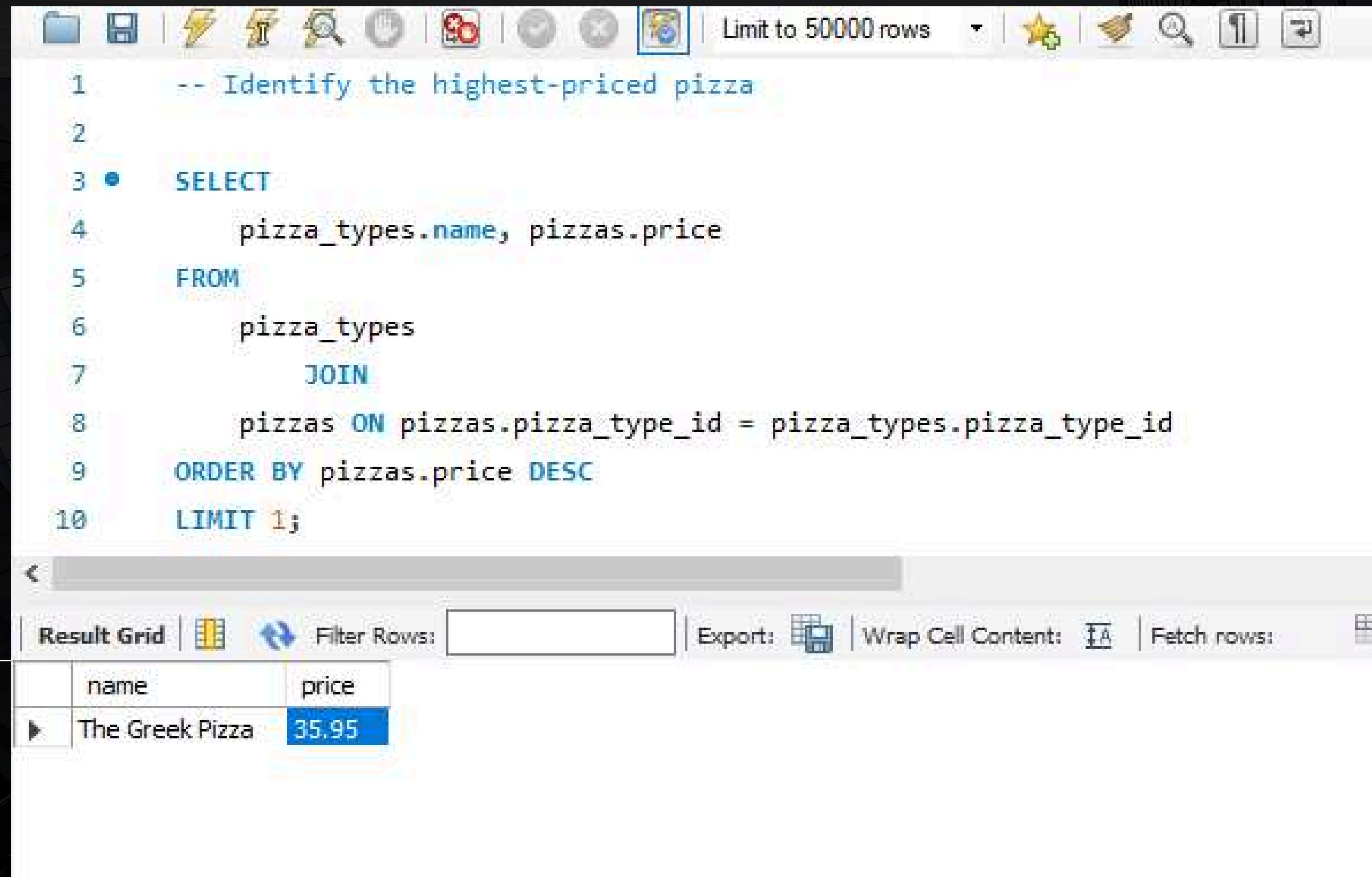
	revenue
▶	817860.05

05



# IDENTIFY THE HIGHEST-PRICED PIZZA

20  
24



The screenshot shows a SQL query editor with a toolbar at the top containing icons for file operations, search, and execution. The query text 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 pizzas.pizza_type_id = pizza_types.pizza_type_id
9  ORDER BY pizzas.price DESC
10 LIMIT 1;
```

Below the query editor is a result grid. The toolbar for the result grid includes a 'Result Grid' tab, a 'Filter Rows' input field, an 'Export' button, a 'Wrap Cell Content' toggle, and a 'Fetch rows' button. The result grid displays the following data:

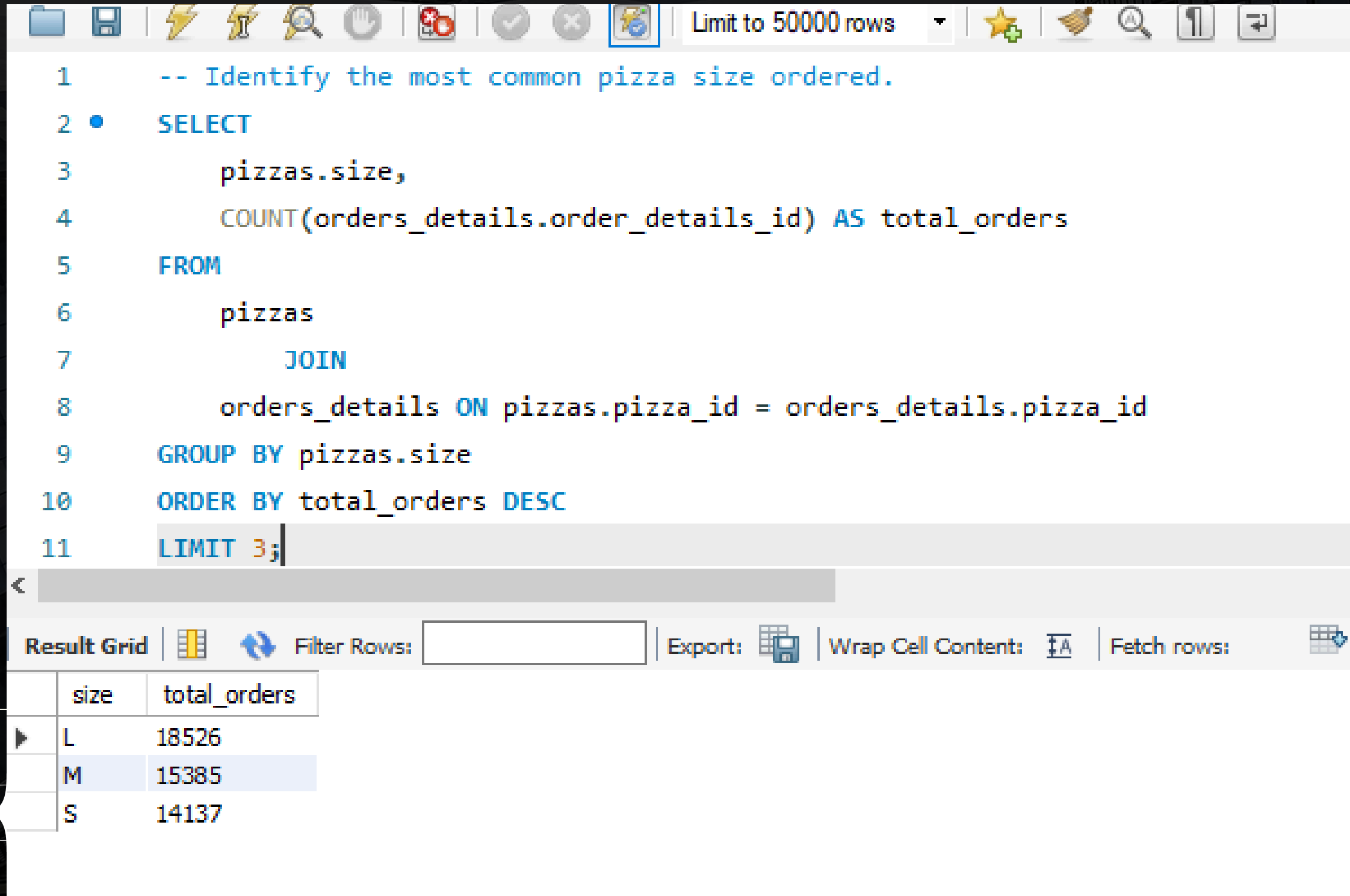
	name	price
▶	The Greek Pizza	35.95

05



# IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

20  
24



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

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

Below the query editor, there is a "Result Grid" section with a toolbar. The results are displayed in a table:

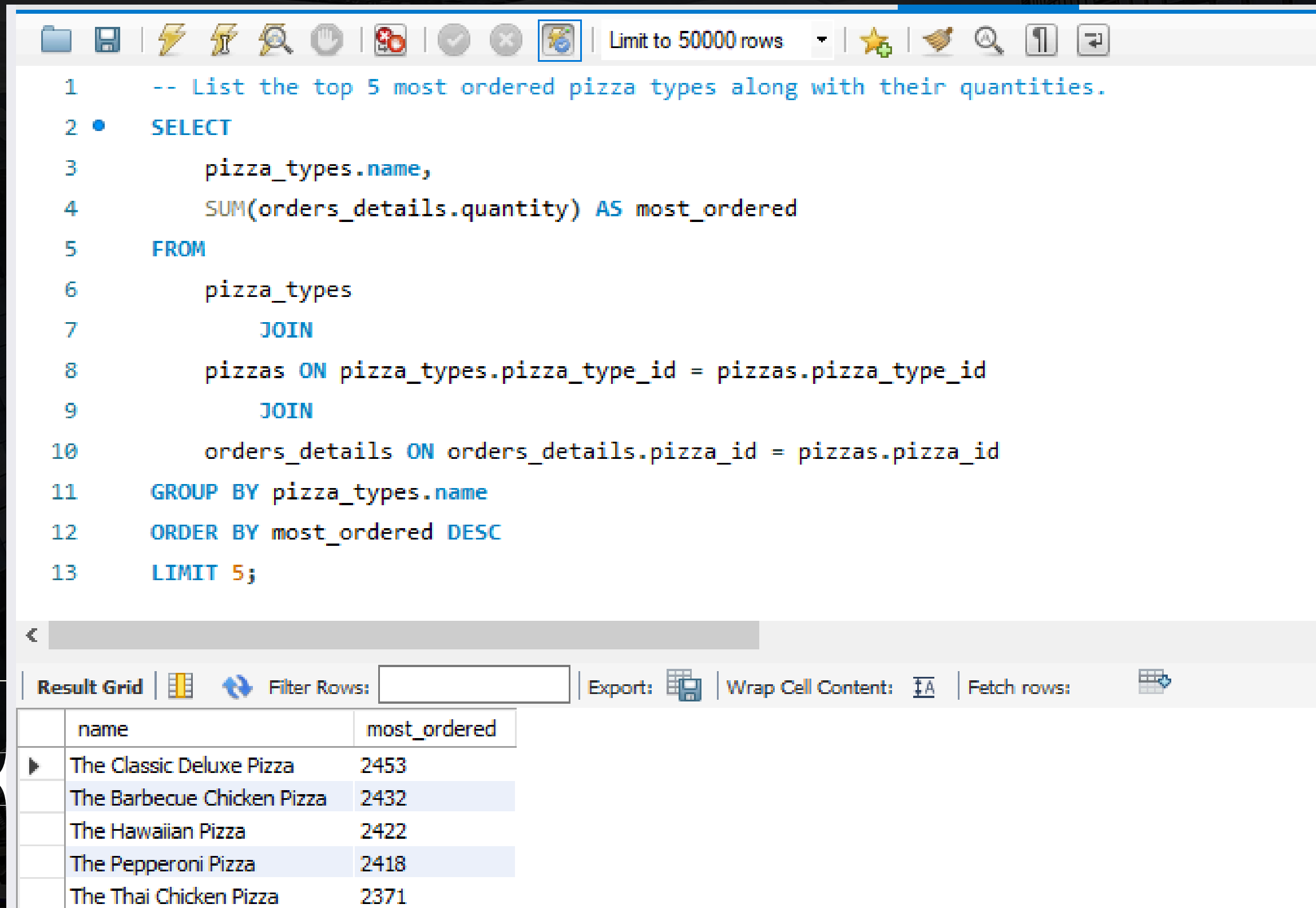
	size	total_orders
▶	L	18526
	M	15385
	S	14137

05



# LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.

20  
24



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and search, along with a 'Limit to 50000 rows' dropdown. The SQL editor contains a query to list the top 5 most ordered pizza types. Below the editor, the 'Result Grid' tab is active, displaying a table with two columns: 'name' and 'most\_ordered'. The results show five pizza types: 'The Classic Deluxe Pizza' (2453), 'The Barbecue Chicken Pizza' (2432), 'The Hawaiian Pizza' (2422), 'The Pepperoni Pizza' (2418), and 'The Thai Chicken Pizza' (2371).

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


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

05






# JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

20  
24



```
1  -- Join the necessary tables to find the total quantity of each pizza category ordered.
2  • SELECT
3      pizza_types.category,
4      SUM(orders_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     orders_details ON orders_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

05



# DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

20  
24

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

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

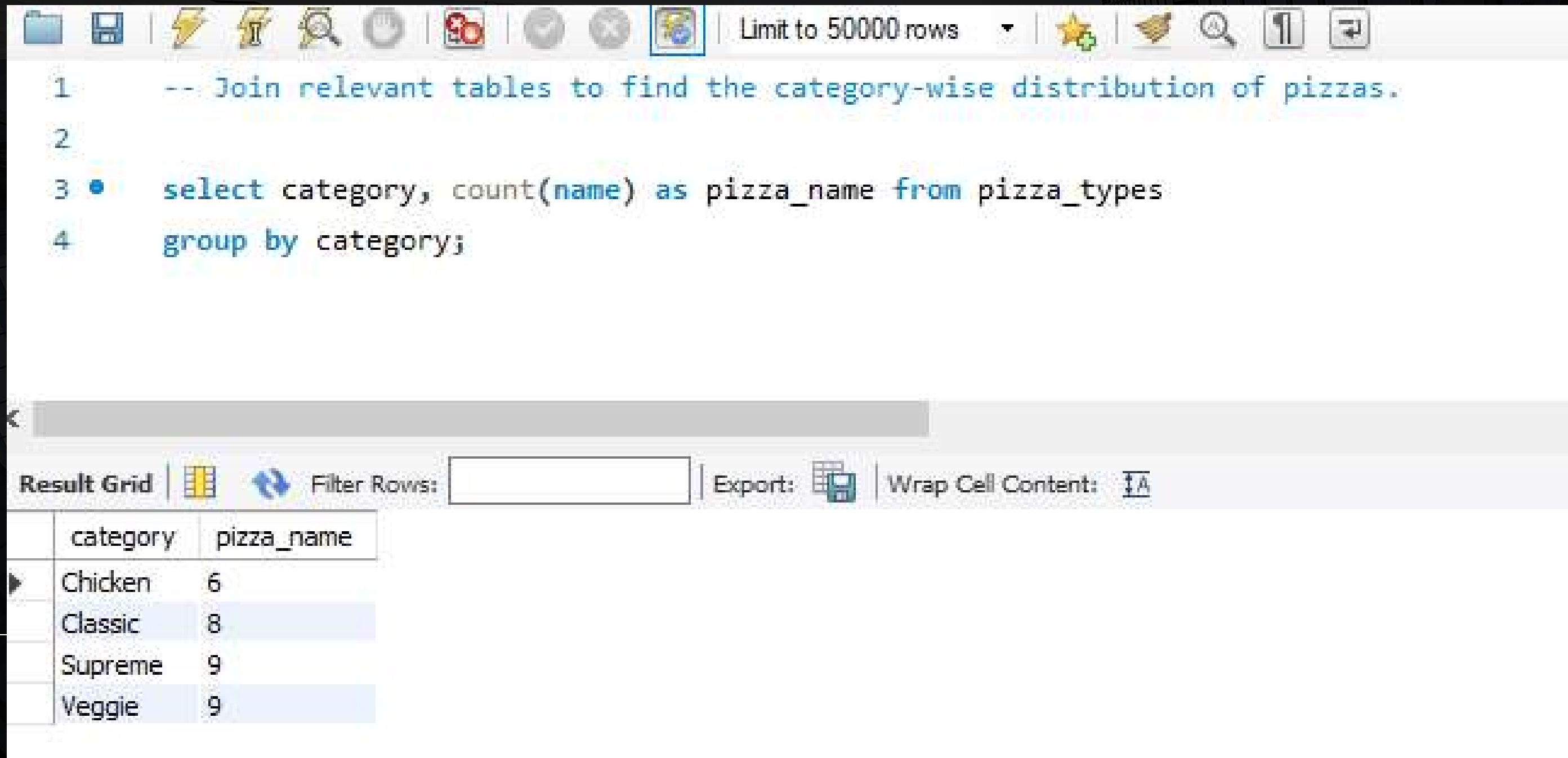
	hours	orders
▶	11	1231
	12	2520
	13	2455
	14	1472
	15	1468
	16	1920
	17	2336
	18	2399
	19	2009
	20	1642
	21	1198

05



# JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

20  
24



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 50000 rows' dropdown. The SQL editor contains the following code:

```
1  -- Join relevant tables to find the category-wise distribution of pizzas.  
2  
3  select category, count(name) as pizza_name from pizza_types  
4  group by category;
```

Below the editor is a 'Result Grid' section with a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are displayed in a table:

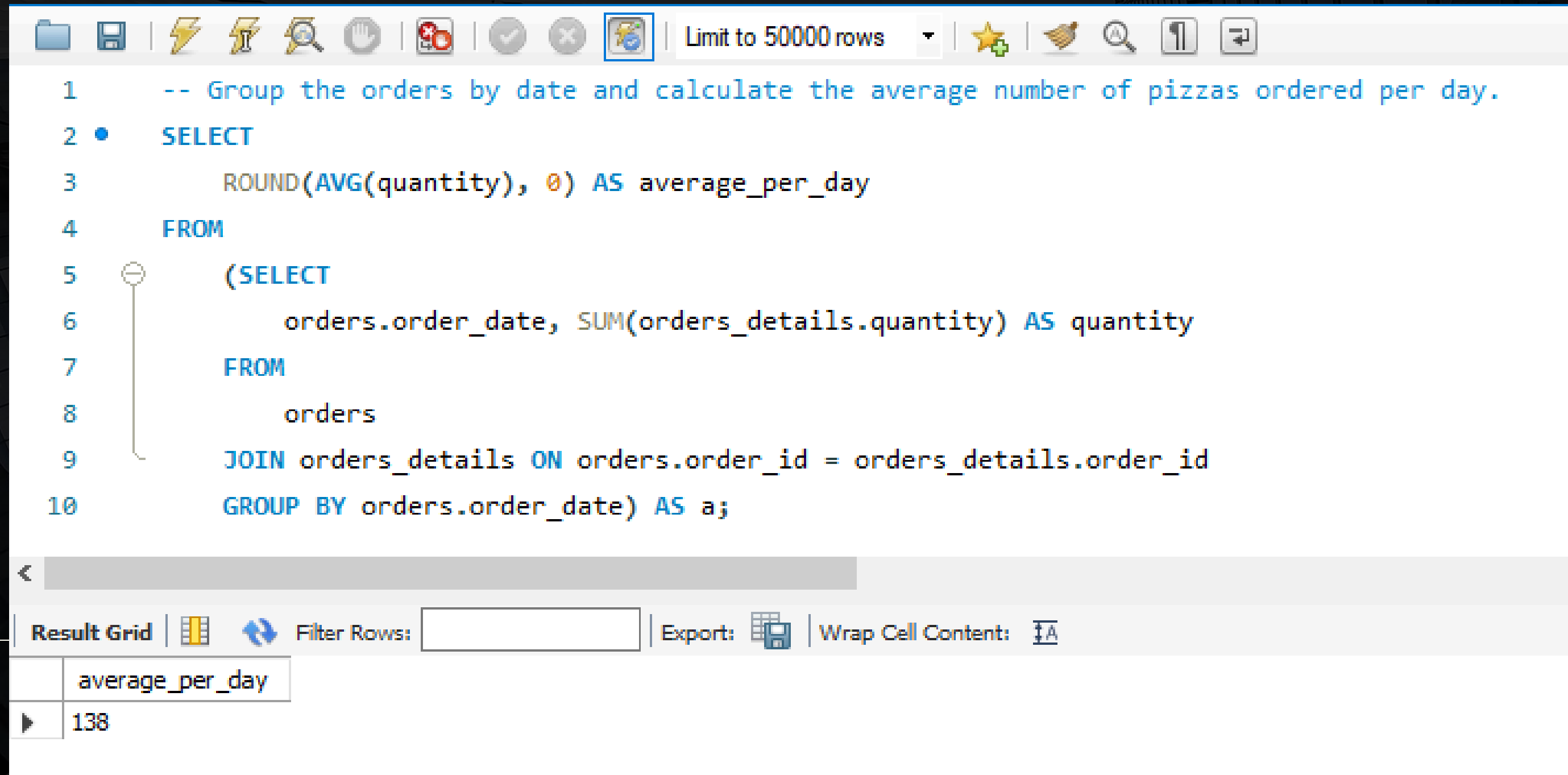
category	pizza_name
Chicken	6
Classic	8
Supreme	9
Veggie	9

05



# GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

20  
24



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 50000 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 average_per_day
4  FROM
5      (SELECT
6          orders.order_date, SUM(orders_details.quantity) AS quantity
7      FROM
8          orders
9      JOIN orders_details ON orders.order_id = orders_details.order_id
10     GROUP BY orders.order_date) AS a;
```

Below the editor is a 'Result Grid' section with a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The result grid shows one column, 'average\_per\_day', with a value of 138.

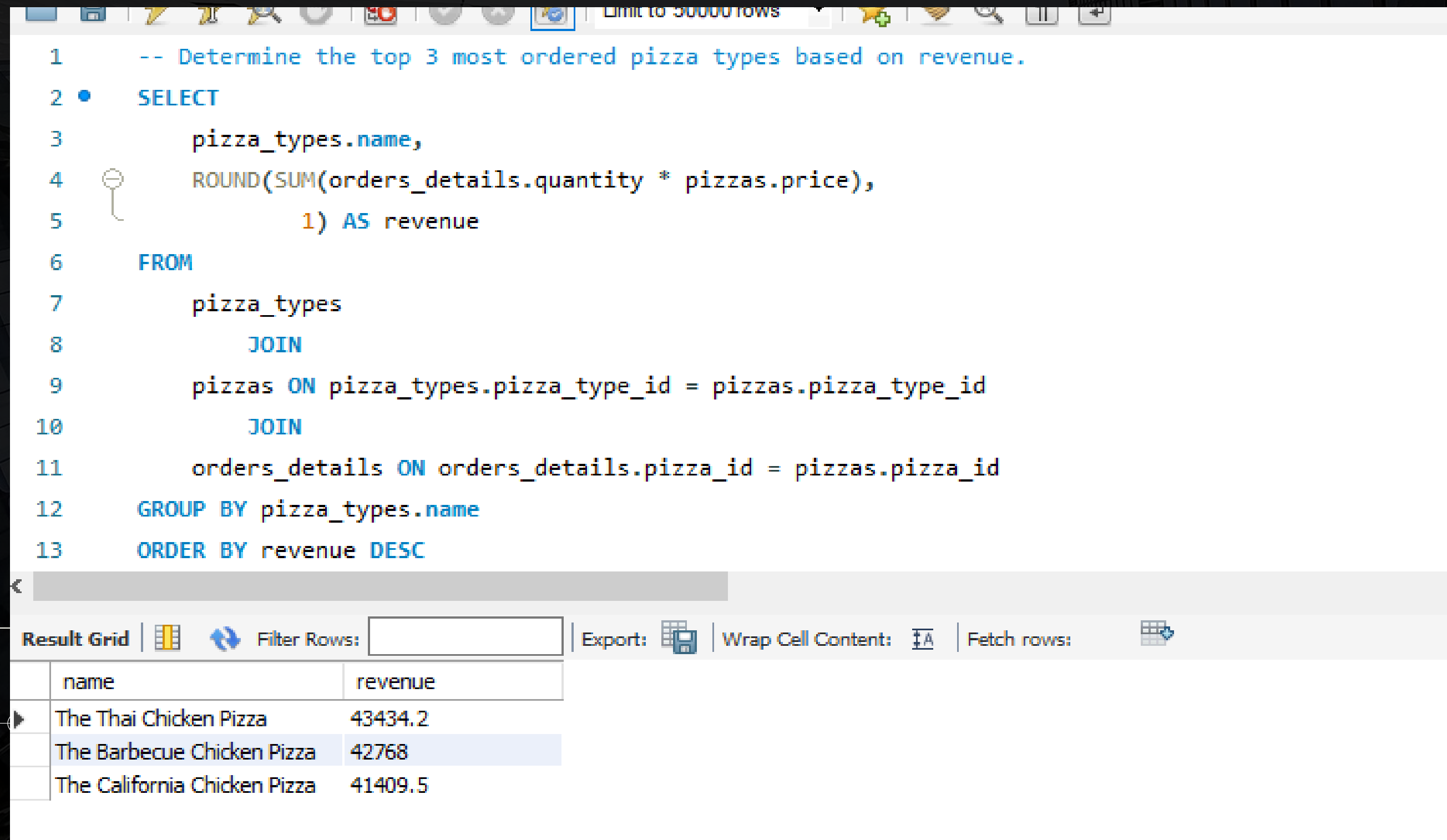
average_per_day
138

05



# DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

20  
24



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

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

	name	revenue
▶	The Thai Chicken Pizza	43434.2
	The Barbecue Chicken Pizza	42768
	The California Chicken Pizza	41409.5

05



# CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

20  
24

```
2 • SELECT
3     pizza_types.category,
4     round( (SUM(orders_details.quantity * pizzas.price) / (SELECT
5         ROUND(SUM(orders_details.quantity * pizzas.price),0)
6     FROM
7         orders_details
8         JOIN
9         pizzas ON pizzas.pizza_id = orders_details.pizza_id) ) * 100, 0) as revenue
10 FROM
11     pizza_types
12     JOIN
13     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
14     JOIN
15     orders_details ON orders_details.pizza_id = pizzas.pizza_id
16 GROUP BY pizza_types.category
17 ORDER BY revenue DESC;
```

	category	revenue
▶	Classic	27
	Supreme	25
	Veggie	24
	Chicken	24

05



**THANK YOU!**

