

PIZZA HUT SALES ANALYSIS

Deriving business insights through SQL

Presented by
Sahil Thakur



Understanding the Sales Data

To analyze pizza sales data and extract actionable insights using SQL techniques.



Tools Used: SQL
(PostgreSQL/MySQL/SQLite), Excel
(for viewing), GitHub

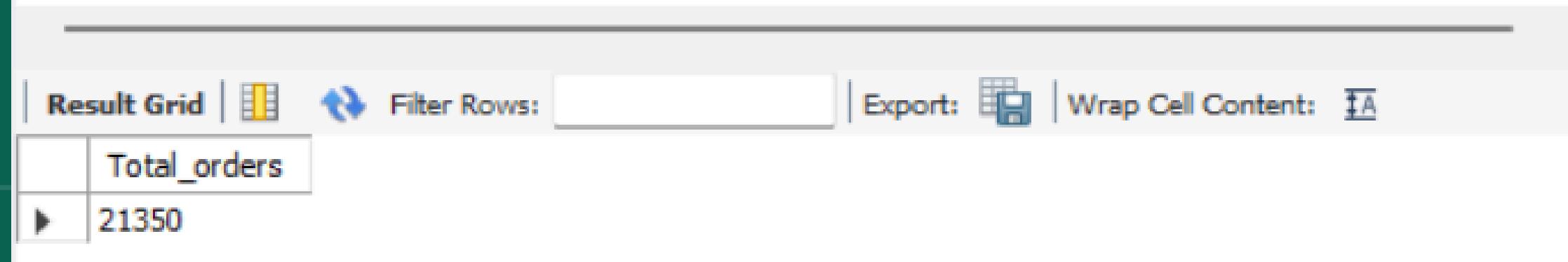
Dataset source : Public sample dataset used for learning purposes

Tables Involved:

- **orders**
- **order_details**
- **pizzas**
- **pizza_types**

Retrieve the total number of orders placed.

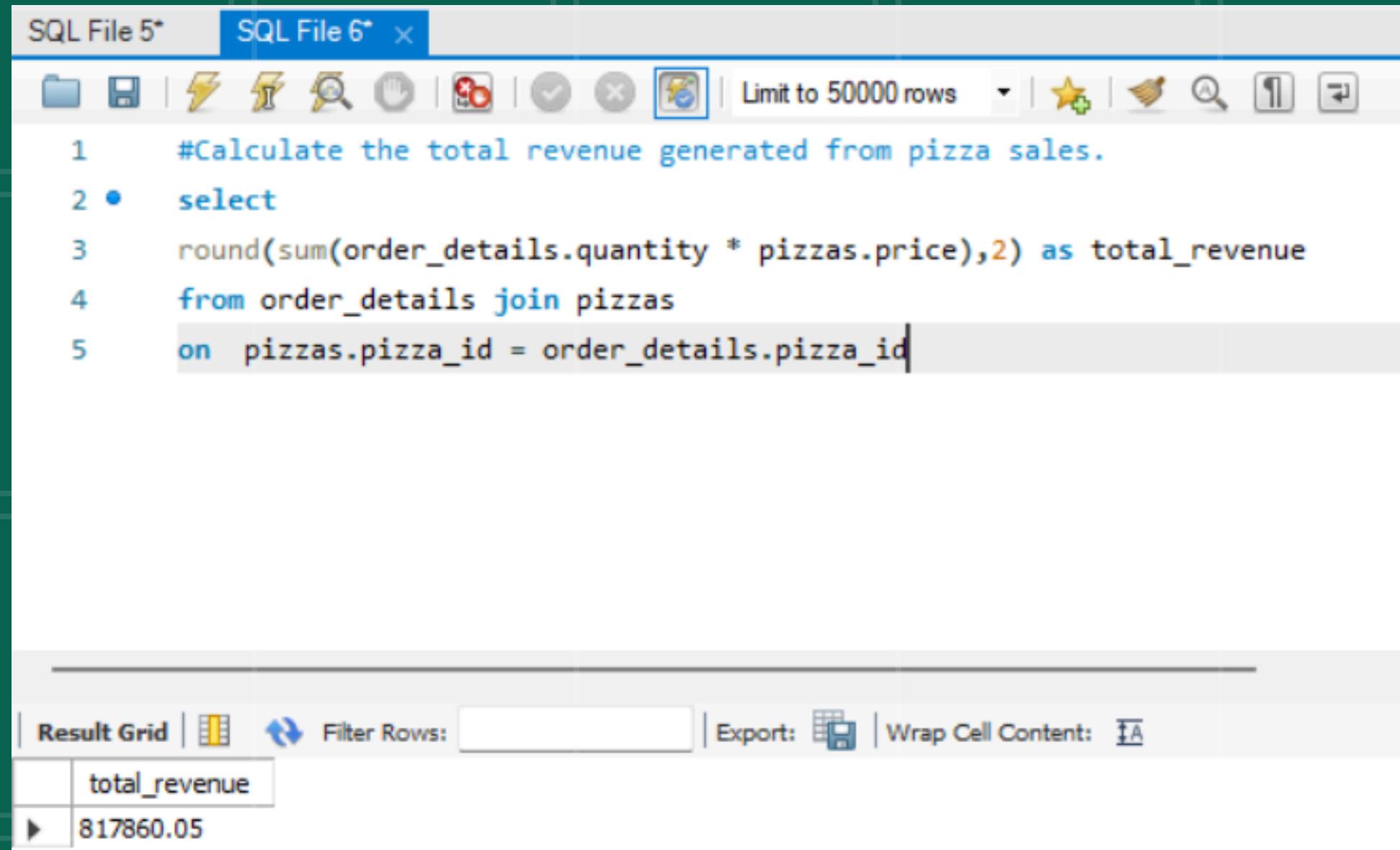
```
1  #Retrieve the total number of orders placed.  
2 • select count(order_id) as Total_orders from pizzahut.orders;
```



The screenshot shows the MySQL Workbench interface with a result grid. The grid has one row and two columns. The first column is labeled 'Total_orders' and contains the value '21350'. The interface includes standard database navigation buttons like 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'.

Total_orders	21350
--------------	-------

Calculate the total revenue generated from pizza sales.



The screenshot shows a MySQL Workbench interface with two tabs: "SQL File 5*" and "SQL File 6*". The "SQL File 6*" tab is active and contains the following SQL code:

```
1  #Calculate the total revenue generated from pizza sales.
2  • select
3      round(sum(order_details.quantity * pizzas.price),2) as total_revenue
4  from order_details join pizzas
5    on pizzas.pizza_id = order_details.pizza_id
```

The result grid below the code shows one row of data:

	total_revenue
▶	817860.05

Identify the highest-priced .

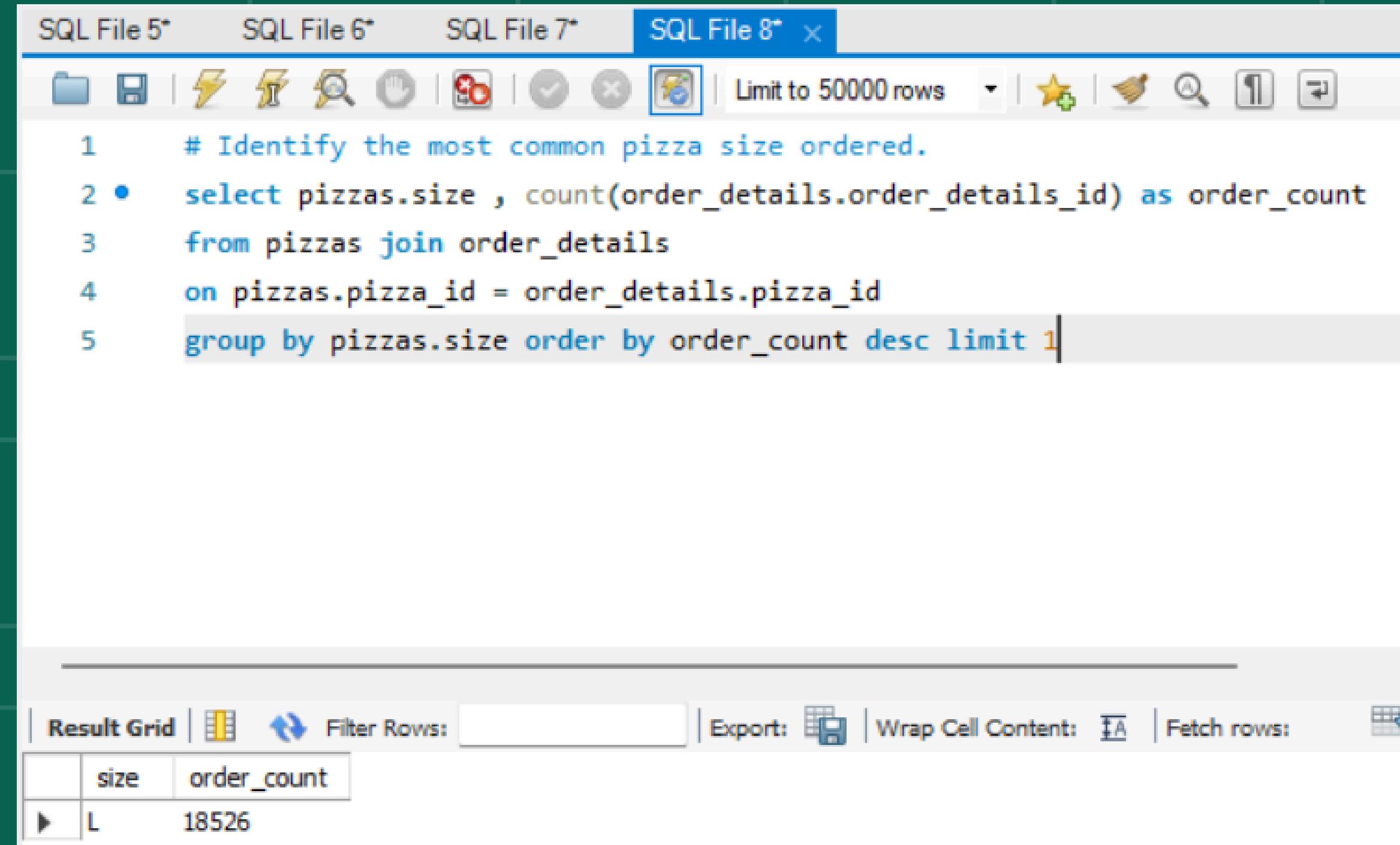
The screenshot shows a MySQL query editor window. At the top, there's a toolbar with various icons for file operations, search, and database management. A dropdown menu says "Limit to 50000 rows". Below the toolbar, the SQL query is displayed:

```
1 #Identify the highest-priced pizza.
2 • Select pizza_types.name , pizzas.price
3 from pizza_types join pizzas
4 on pizza_types.pizza_type_id = pizzas.pizza_type_id
5 order by price desc limit 1
```

At the bottom of the editor, there are buttons for "Result Grid" (selected), "Filter Rows:", "Export:", "Wrap Cell Content:", and "Fetch rows". The results grid shows one row of data:

	name	price
▶	The Greek Pizza	35.95

Identify the most common pizza size ordered.



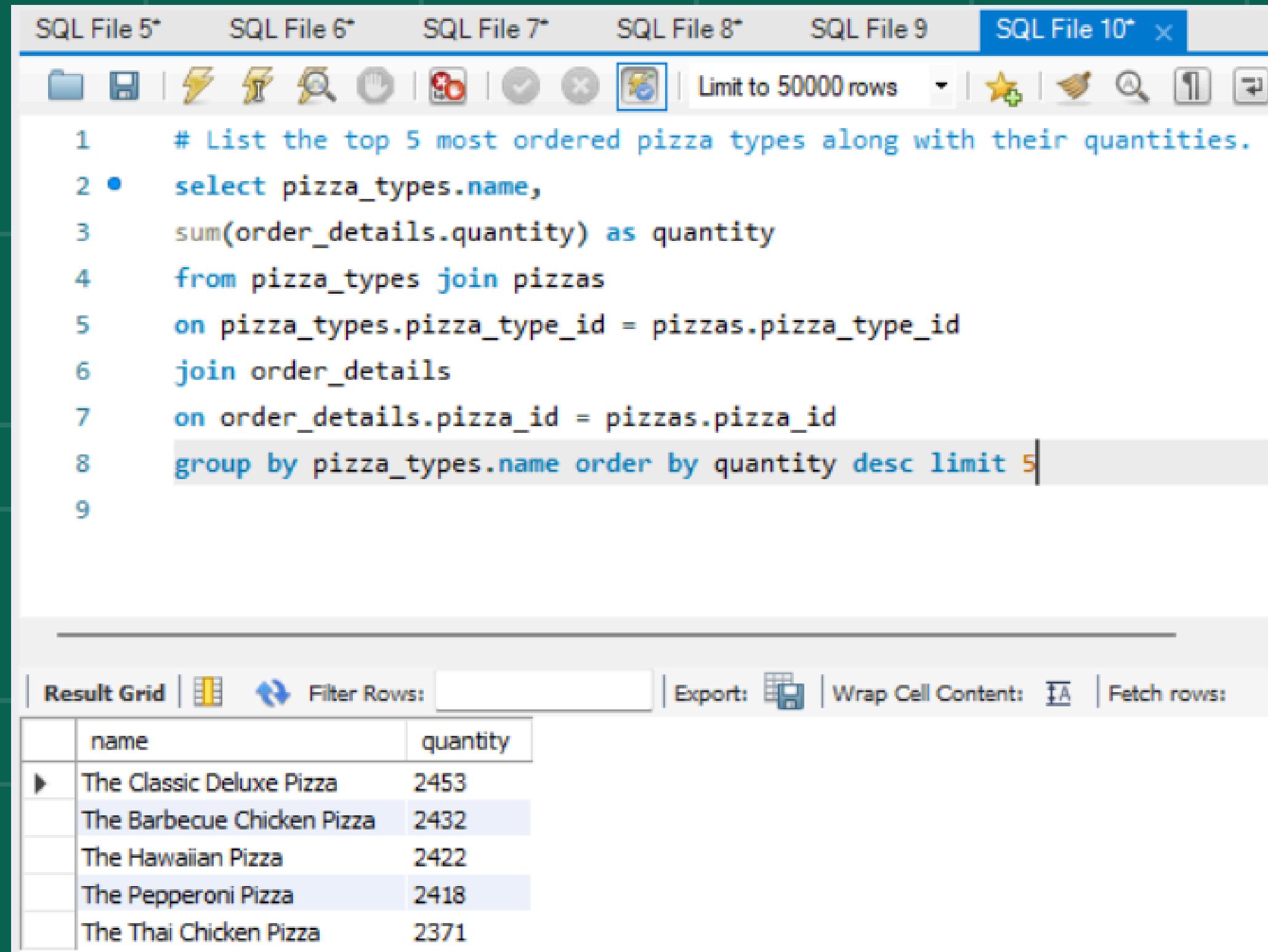
The screenshot shows a MySQL Workbench interface with a SQL editor window. The title bar of the window is labeled "SQL File 8*". The toolbar contains various icons for file operations, search, and database navigation. A dropdown menu "Limit to 50000 rows" is open. The SQL code in the editor is:

```
1  # Identify the most common pizza size ordered.
2 • select pizzas.size , count(order_details.order_details_id) as order_count
3   from pizzas join order_details
4     on pizzas.pizza_id = order_details.pizza_id
5   group by pizzas.size order by order_count desc limit 1
```

The result grid at the bottom shows one row of data:

	size	order_count
▶	L	18526

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



The screenshot shows a MySQL query editor interface with the following details:

- SQL File 10*** is the active tab.
- The toolbar includes icons for file operations (New, Save, Run, Undo, Redo, Find, Replace, Copy, Paste, etc.) and a "Limit to 50000 rows" dropdown.
- The SQL code is:

```
1  # List the top 5 most ordered pizza types along with their quantities.
2 •  select pizza_types.name,
3      sum(order_details.quantity) as quantity
4  from pizza_types join pizzas
5  on pizza_types.pizza_type_id = pizzas.pizza_type_id
6  join order_details
7  on order_details.pizza_id = pizzas.pizza_id
8  group by pizza_types.name order by quantity desc limit 5
9
```
- The result grid displays the following data:

	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.

The screenshot shows a SQL editor interface with multiple tabs at the top labeled "SQL File 5*", "SQL File 6*", "SQL File 7*", "SQL File 8*", "SQL File 9", "SQL File 10*", and "SQL File 11*". The "SQL File 11*" tab is active. Below the tabs is a toolbar with various icons for file operations, search, and database management. A status bar indicates "Limit to 50000 rows". The main area contains the following SQL code:

```
1  # Join the necessary tables to find the total quantity of each pizza category ordered.
2 • select pizza_types.category,
3   sum(order_details.quantity) as quantity
4   from pizza_types join pizzas
5   on pizza_types.pizza_type_id = pizzas.pizza_type_id
6   join order_details
7   on order_details.pizza_id = pizzas.pizza_id
8   group by pizza_types.category order by quantity desc
```

Below the code, there is a "Result Grid" section with a table showing the results of the query. The table has two columns: "category" and "quantity". The data is as follows:

	category	quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

Determine the distribution of orders by hour of the day.

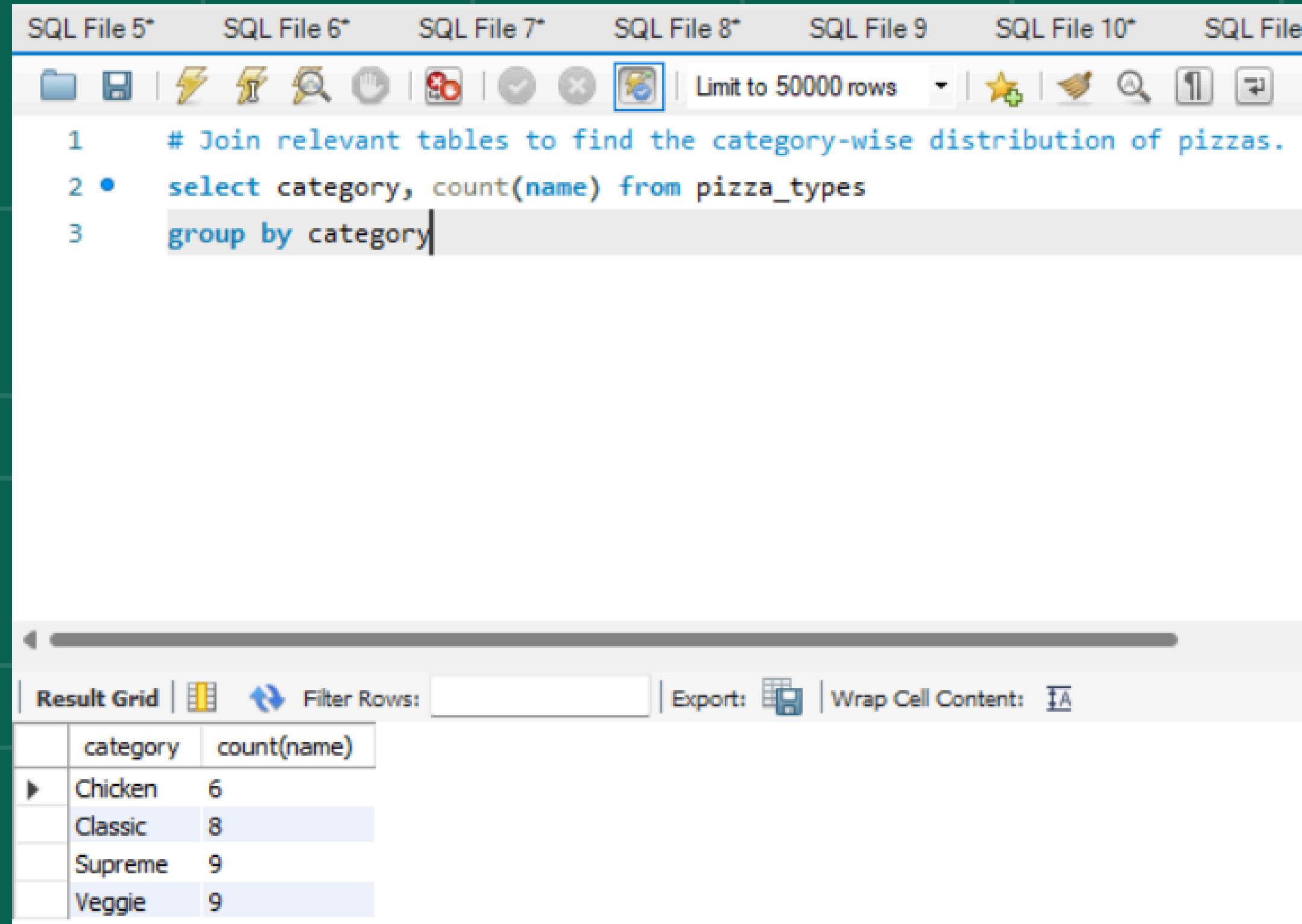
The screenshot shows a MySQL query editor interface. At the top, there are tabs for "SQL File 5*", "SQL File 6*", "SQL File 7*", "SQL File 8*", "SQL File 9", and "SQL File 10*". Below the tabs is a toolbar with various icons for file operations, search, and database management. A dropdown menu says "Limit to 50000 rows". The main area contains the following SQL code:

```
1  # Determine the distribution of orders by hour of the day.
2 •  SELECT hour(time), count(order_id) FROM pizzahut.orders
3   group by time
```

Below the code is a "Result Grid" table with two columns: "hour(time)" and "count(order_id)". The data is as follows:

	hour(time)	count(order_id)
▶	11	2
	11	1
	12	1
	12	3
	12	1
	12	1
	12	1
	12	1
	12	1
	13	2
	13	1
	13	2
	13	1
	13	2
	13	2
	13	3
	13	1

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



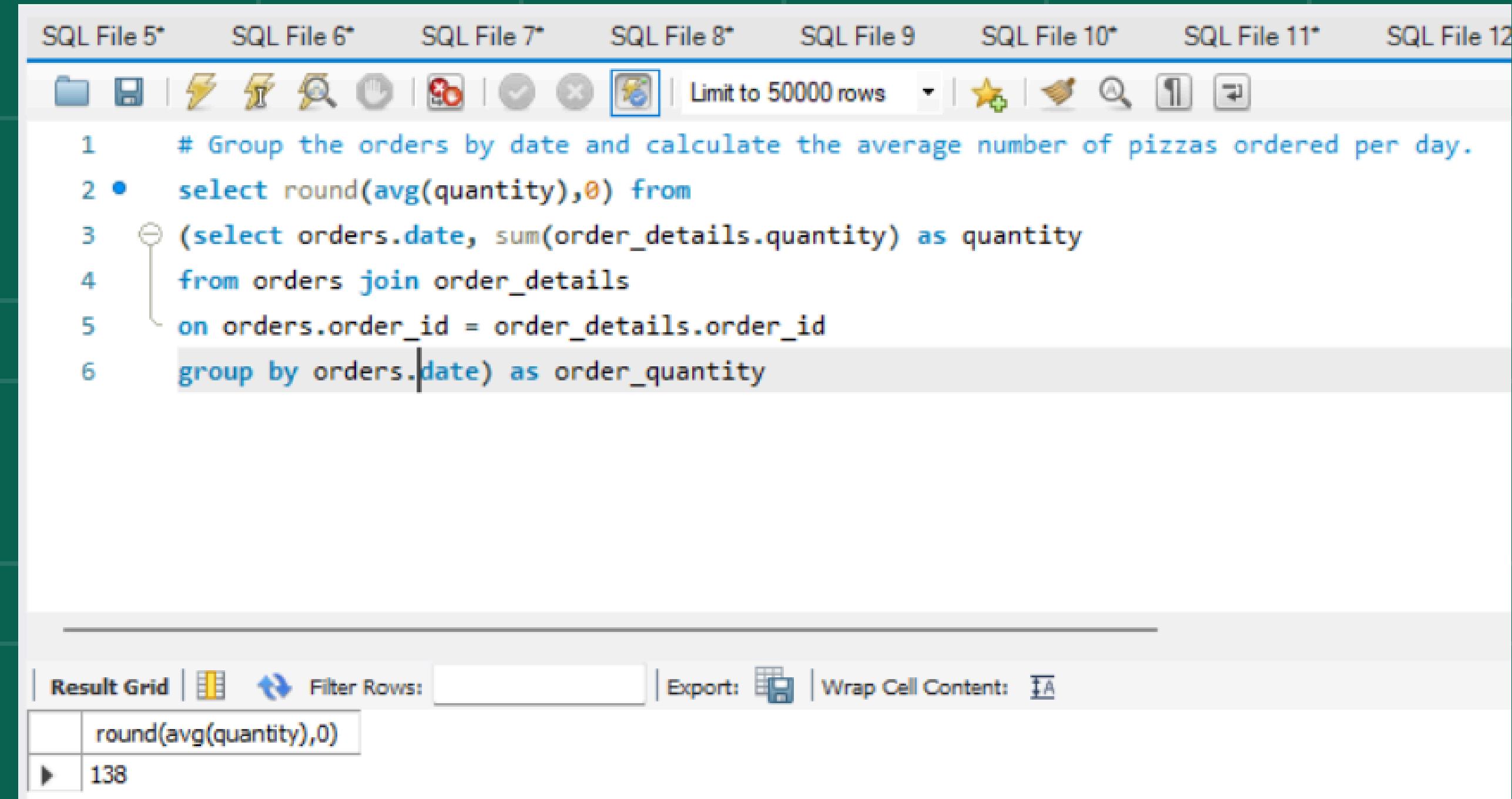
The screenshot shows a MySQL Workbench interface. The SQL editor tab has the following content:

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

The result grid below displays the following data:

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

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



The screenshot shows a MySQL Workbench interface with a SQL editor window. The window title is "SQL File 5*". The SQL code is as follows:

```
1  # Group the orders by date and calculate the average number of pizzas ordered per day.
2 • select round(avg(quantity),0) from
3   (select orders.date, sum(order_details.quantity) as quantity
4    from orders join order_details
5    on orders.order_id = order_details.order_id
6    group by orders.date) as order_quantity
```

The result grid at the bottom shows one row with the value "138".

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

The screenshot shows a MySQL Workbench interface with a SQL editor and a result grid.

SQL Editor:

```
SQL File 5* SQL File 6* SQL File 7* SQL File 8* SQL File 9 SQL File 10*
1 #Determine the top 3 most ordered pizza types based on revenue.
2 • s Execute the selected portion of the script or everything, if there is no selection
3 sum(order_details.quantity * pizzas.price) as revenue
4 from pizza_types join pizzas
5 on pizzas.pizza_type_id = pizza_types.pizza_type_id
6 join order_details
7 on order_details.pizza_id = pizzas.pizza_id
8 group by pizza_types.name order by revenue desc limit 3
```

Result Grid:

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

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

The screenshot shows a MySQL Workbench interface with a SQL editor and a result grid.

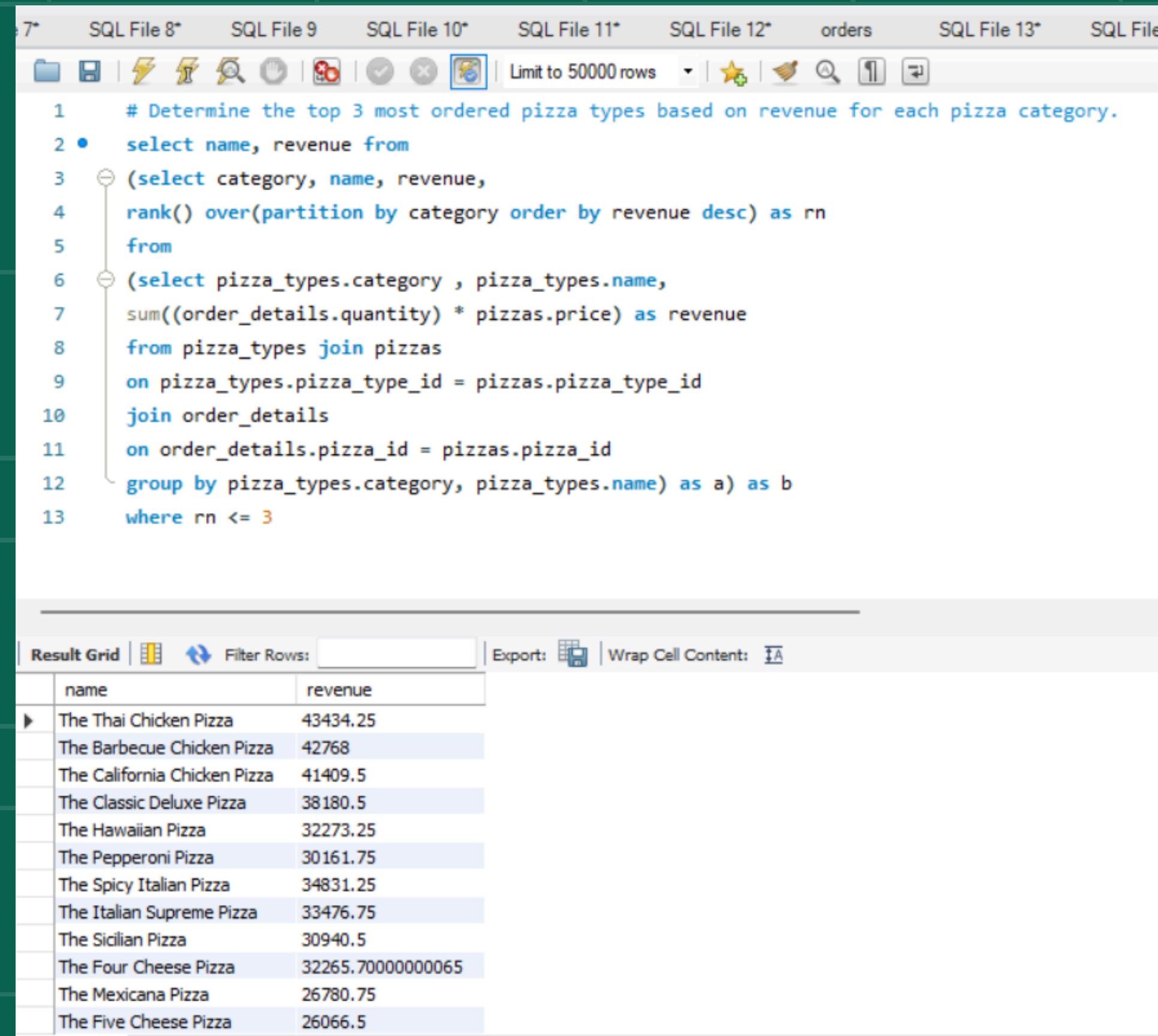
SQL Editor:

```
1  # Calculate the percentage contribution of each pizza type to total revenue.
2 • select pizza_types.category,
3   round(sum(order_details.quantity * pizzas.price)/ (select round(sum(order_details.quantity * pizzas.price),2) as total_sales
4   from order_details join pizzas
5   on pizzas.pizza_id = order_details.pizza_id)*100,2) as revenue
6   from pizza_types join pizzas
7   on pizza_types.pizza_type_id = pizzas.pizza_type_id
8   join order_details
9   on order_details.pizza_id = pizzas.pizza_id
10  group by pizza_types.category order by revenue desc
```

Result Grid:

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

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



The screenshot shows a SQL editor interface with a code editor at the top and a result grid at the bottom. The code editor contains a SQL query to determine the top 3 most ordered pizza types based on revenue for each pizza category. The result grid displays the top 15 pizza types with their names and revenues.

```
1  # Determine the top 3 most ordered pizza types based on revenue for each pizza category.
2  • select name, revenue from
3  (select category, name, revenue,
4  rank() over(partition by category order by revenue desc) as rn
5  from
6  (select pizza_types.category , pizza_types.name,
7  sum((order_details.quantity) * pizzas.price) as revenue
8  from pizza_types join pizzas
9  on pizza_types.pizza_type_id = pizzas.pizza_type_id
10 join order_details
11 on order_details.pizza_id = pizzas.pizza_id
12 group by pizza_types.category, pizza_types.name) as a) as b
13 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

Conclusion & Key Takeaways

- Learned how to retrieve aggregated data using COUNT(), SUM(), and MAX() to answer simple business queries like total orders, revenue, and highest-priced items.
- Understood how to identify most frequent values using GROUP BY and ORDER BY, helping in customer preference analysis.
- Gained experience in using joins across multiple tables to combine relevant data for deeper insights.
- Practiced using date/time functions (EXTRACT, HOUR, etc.) to analyze order patterns across hours and days.
- Learned how to group data effectively to understand category-wise distributions and average daily performance, enabling better sales trend identification.
- Understood the use of percentage calculations and subqueries to analyze revenue contribution per item.
- Practiced cumulative analysis using window functions or running totals to see business growth over time.
- Mastered the logic to apply multi-level grouping and filtering (e.g., top 3 pizzas by category revenue), helping simulate real-world business reporting.

A photograph of a man with a dark beard and short hair, wearing a blue blazer over a light-colored shirt. He is looking down at a black smartphone he is holding in his hands. The background is slightly blurred, showing what appears to be an office or a modern interior space.

Thank You

sahilt018@gmail.com