

PIZZA SALE ANALYSIS

DATA ANALYSIS

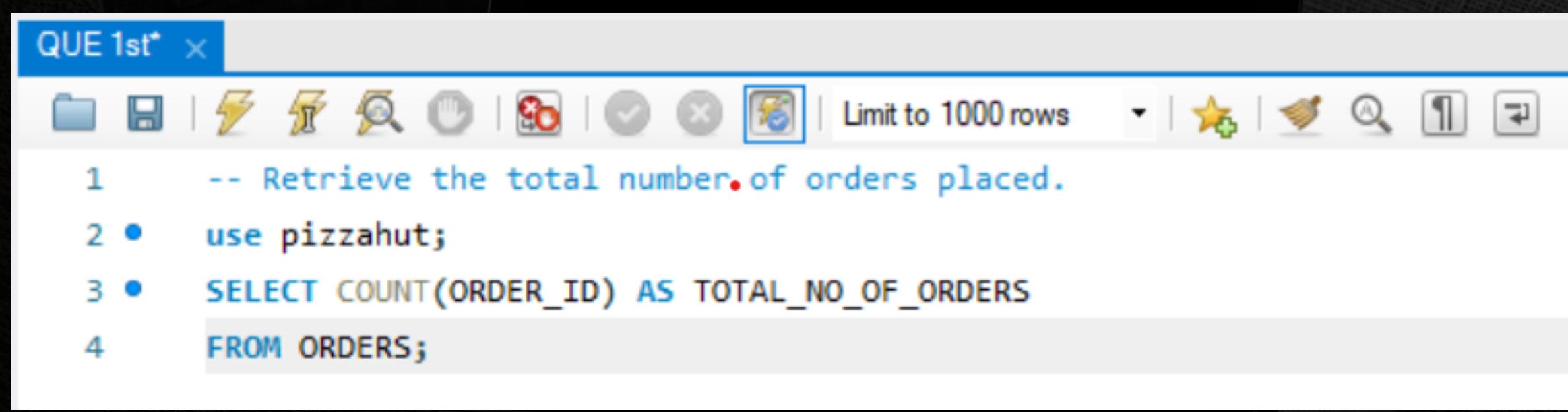


RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

FINANCIAL OUTLOOK

QUE 1st* X

1 -- Retrieve the total number of orders placed.
2 • use pizzahut;
3 • SELECT COUNT(ORDER_ID) AS TOTAL_NO_OF_ORDERS
4 FROM ORDERS;



Result Grid |  Filter Rows: _____

	TOTAL_NO_OF_ORDERS
▶	21350

-- CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES

The screenshot shows a MySQL query editor interface. The top bar has tabs labeled "QUE 1st*" and "QUE 2nd X". Below the tabs is a toolbar with various icons for file operations, search, and navigation. The main area contains a SQL query:

```
1 -- Calculate the total revenue generated from pizza sales
2 • SELECT
3     SUM((orders_details.QUANTITY * pizzas.price)) AS TOTAL_REVENUE
4     FROM orders_details JOIN PIZZAS
5     ON orders_details.PIZZA_ID = pizzas.pizza_id;
```

Below the query results, there is a "Result Grid" section. It includes a "Result Grid" button, a refresh icon, and a "Filter Rows:" input field. The result grid displays one row of data:

	TOTAL_REVENUE
▶	817860.049999993

-- IDENTIFY THE HIGHEST-PRICED PIZZA.

The screenshot shows a MySQL Workbench interface. The top window is a query editor with the following SQL code:

```
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 PIZZAS.PRICE DESC LIMIT 1;
```

The bottom window is a "Result Grid" displaying the query results:

	NAME	PRICE
▶	The Greek Pizza	35.95

-- IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

The screenshot shows a MySQL query editor interface. At the top, there are tabs labeled "QUE 1st*", "QUE 2nd", and "QUE 4TH X". Below the tabs is a toolbar with various icons for file operations and database management. A dropdown menu is open, showing the option "Limit to 1000 rows". The main area contains a SQL query:

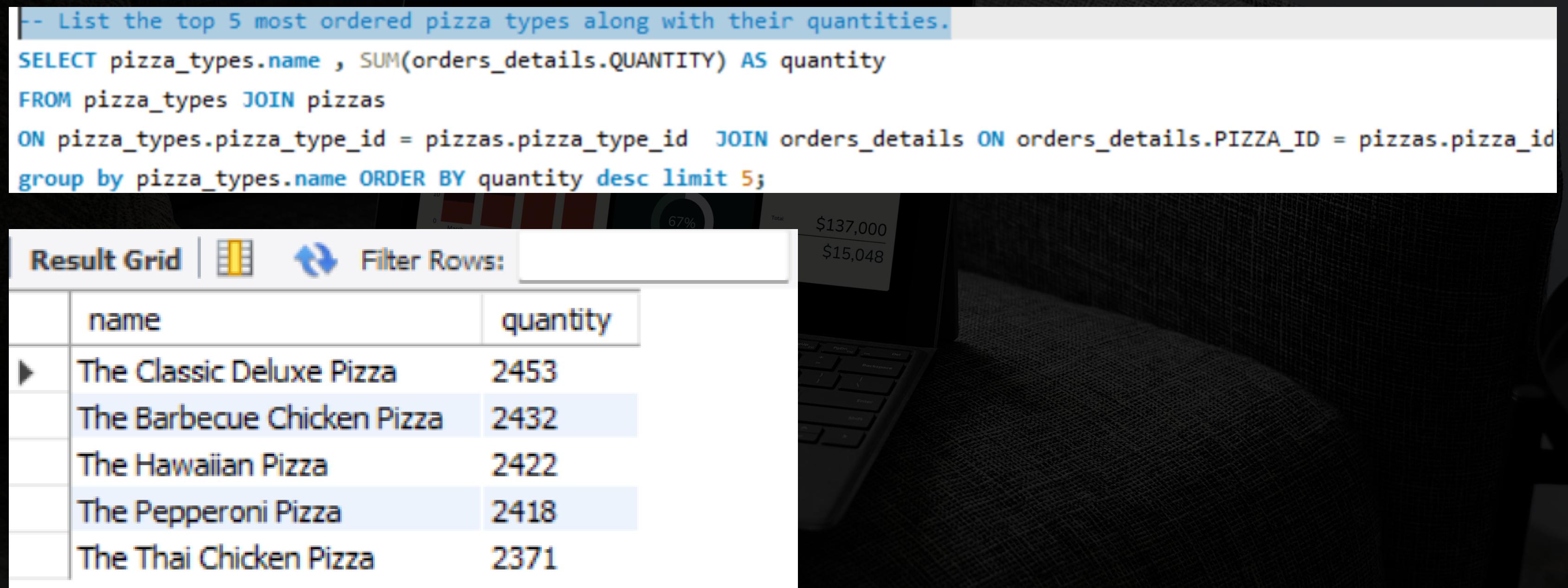
```
1 -- Identify the most common pizza size ordered.  
2 • USE PIZZAHUT;  
3 • SELECT PIZZAS.SIZE , COUNT(ORDERS_DETAILS.ORDERS_DETAILS_ID) AS ORDERS_COUNT  
4 FROM PIZZAS JOIN ORDERS_DETAILS  
5 ON PIZZAS.PIZZA_ID = ORDERS_DETAILS.PIZZA_ID  
6 group by PIZZAS.SIZE  
7 ORDER BY ORDERS_COUNT DESC LIMIT 1;
```

Below the query, the "Result Grid" tab is selected, showing the results of the query:

	SIZE	ORDERS_COUNT
▶	L	18526

There is also a "Filter Rows:" input field at the bottom of the result grid.

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



The image shows a laptop screen with a MySQL database interface. The query window contains the following SQL code:

```
-- List the top 5 most ordered pizza types along with their quantities.  
SELECT pizza_types.name , SUM(orders_details.QUANTITY) AS quantity  
FROM pizza_types JOIN pizzas  
ON pizza_types.pizza_type_id = pizzas.pizza_type_id  JOIN orders_details ON orders_details.PIZZA_ID = pizzas.pizza_id  
group by pizza_types.name ORDER BY quantity desc limit 5;
```

The results window displays a table titled "Result Grid" showing the top 5 most ordered pizza types:

	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

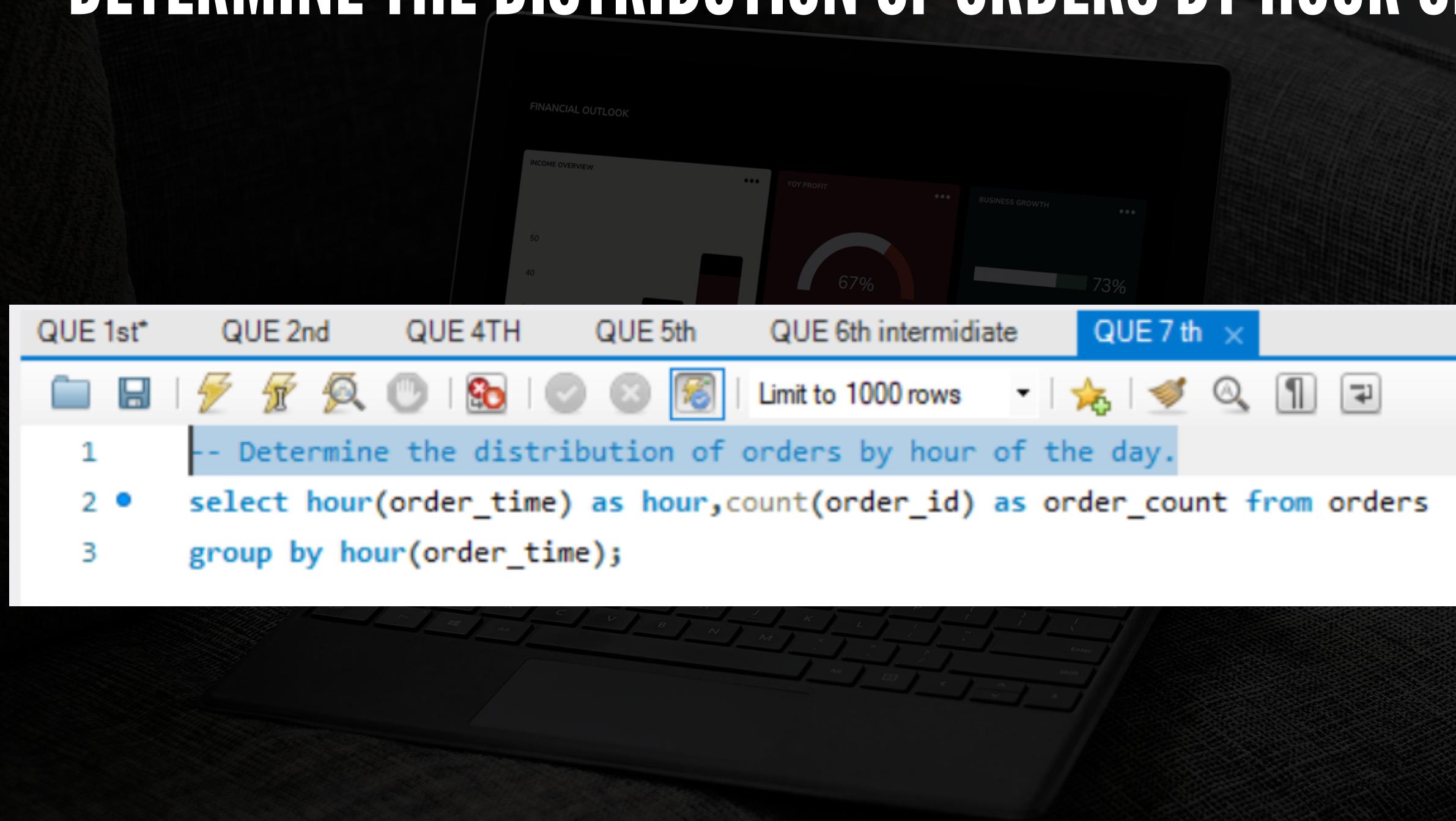
-- JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

```
select pizza_types.category , sum(orders_details.QUANTITY) as no_of_quantity
from pizza_types join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join orders_details
on orders_details.PIZZA_ID = pizzas.pizza_id
group by pizza_types.category ;
```

The screenshot shows a MySQL Workbench interface with a results grid. The grid has two columns: 'category' and 'no_of_quantity'. The data is as follows:

	category	no_of_quantity
▶	Classic	14888
	Veggie	11649
	Supreme	11987
	Chicken	11050

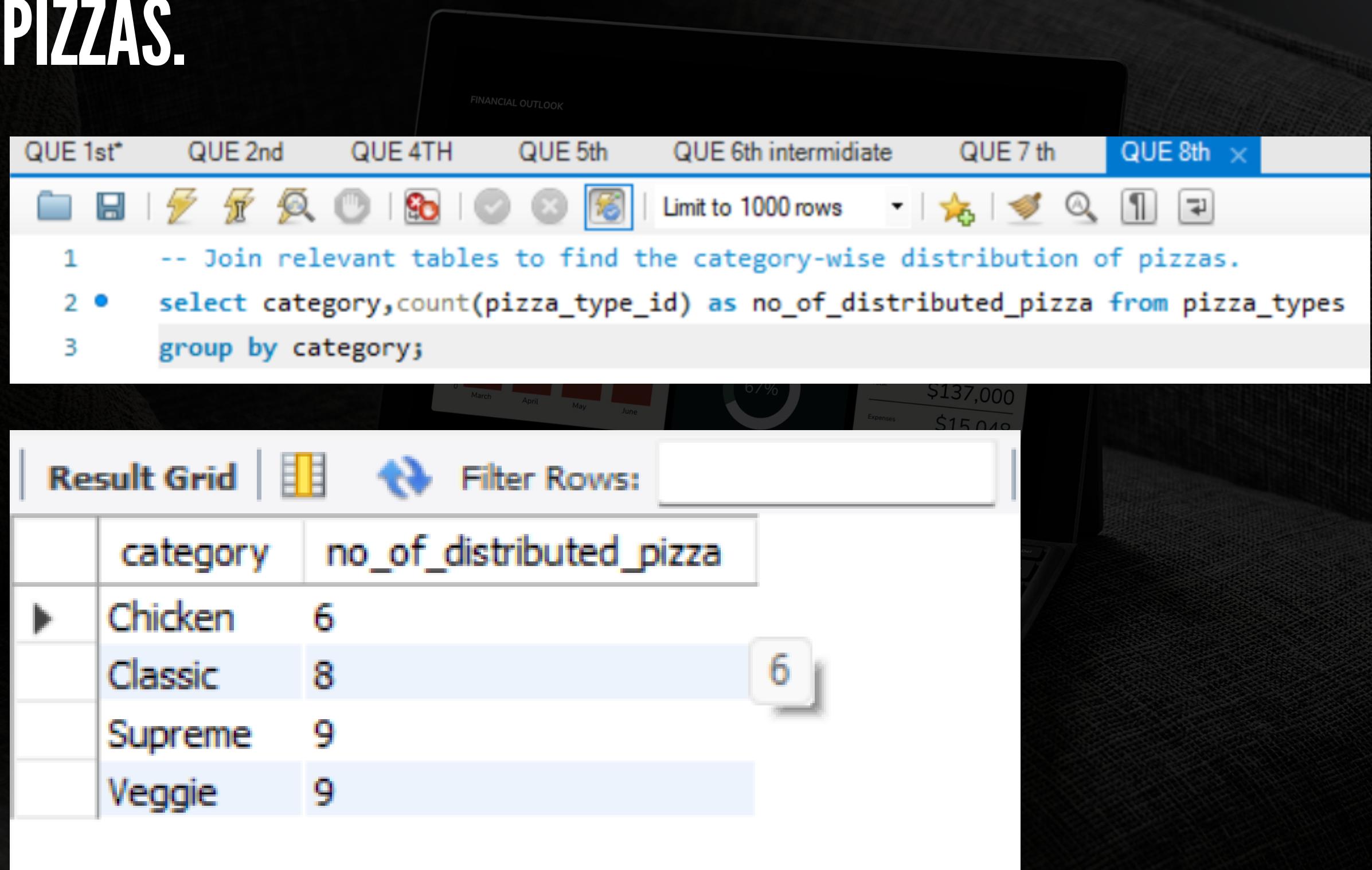
-- DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.



```
FINANCIAL OUTLOOK
INCOME OVERVIEW
YOY PROFIT
BUSINESS GROWTH
67% 73%
QUE 1st* QUE 2nd QUE 4TH QUE 5th QUE 6th intermediate QUE 7 th x
[File] [Save] | [New] [Edit] [Find] [Search] [Run] [Cancel] [Help] | Limit to 1000 rows | [Star] [Comment] [Copy] [Print]
1   -- Determine the distribution of orders by hour of the day.
2 •   select hour(order_time) as hour, count(order_id) as order_count from orders
3   group by hour(order_time);
```

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

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



The screenshot shows a database interface with a query editor and a result grid. The query editor contains the following SQL code:

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

The result grid displays the following data:

	category	no_of_distributed_pizza
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

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

```
-- Group the orders by date and calculate the average number of pizzas ordered per day.  
select avg(quantity) from  
(SELECT ORDERS.ORDER_DATE , SUM(ORDERS_DETAILS.QUANTITY) as quantity  
FROM ORDERS JOIN ORDERS_DETAILS  
ON ORDERS.ORDER_ID = ORDERS_DETAILS.ORDER_ID  
GROUP BY ORDERS.ORDER_DATE) as order_quantity;
```

The screenshot shows a user interface for querying a database. At the top, there are three buttons: "Result Grid" with a grid icon, "Filter Rows" with a filter icon, and another "Filter Rows" button. Below these buttons is a table with two rows of data. The first row contains the column header "avg(quantity)". The second row contains the value "138.4749".

	avg(quantity)
▶	138.4749

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

```
-- Determine the top 3 most ordered pizza types based on revenue.  
-- revenue = quantity * price of pizza  
  
select pizza_types.name , sum(orders_details.QUANTITY * pizzas.price) as revenue  
from pizza_types join pizzas  
on pizza_types.pizza_type_id = pizzas.pizza_type_id  
join orders_details  
on orders_details.pizza_id = pizzas.pizza_id  
group by pizza_types.name  
order by revenue desc limit 3;
```

Result Grid | Filter Rows:

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

CONCLUSION

Through this project, I gained hands-on experience in analyzing real-world sales data using SQL. By writing and executing structured queries, I was able to:

- Retrieve key metrics such as total orders and revenue.
- Identify top-performing pizzas by quantity and revenue.
- Analyze customer preferences based on size and category.
- Understand order trends by time and date for operational planning.

This analysis highlights how powerful SQL can be in turning raw transactional data into meaningful insights. It also strengthens my ability to work with relational databases, perform joins, use aggregate functions, and apply logical thinking to solve business problems.

GROUND

THE INDUSTRY'S HISTORY

THANK YOU....!

