

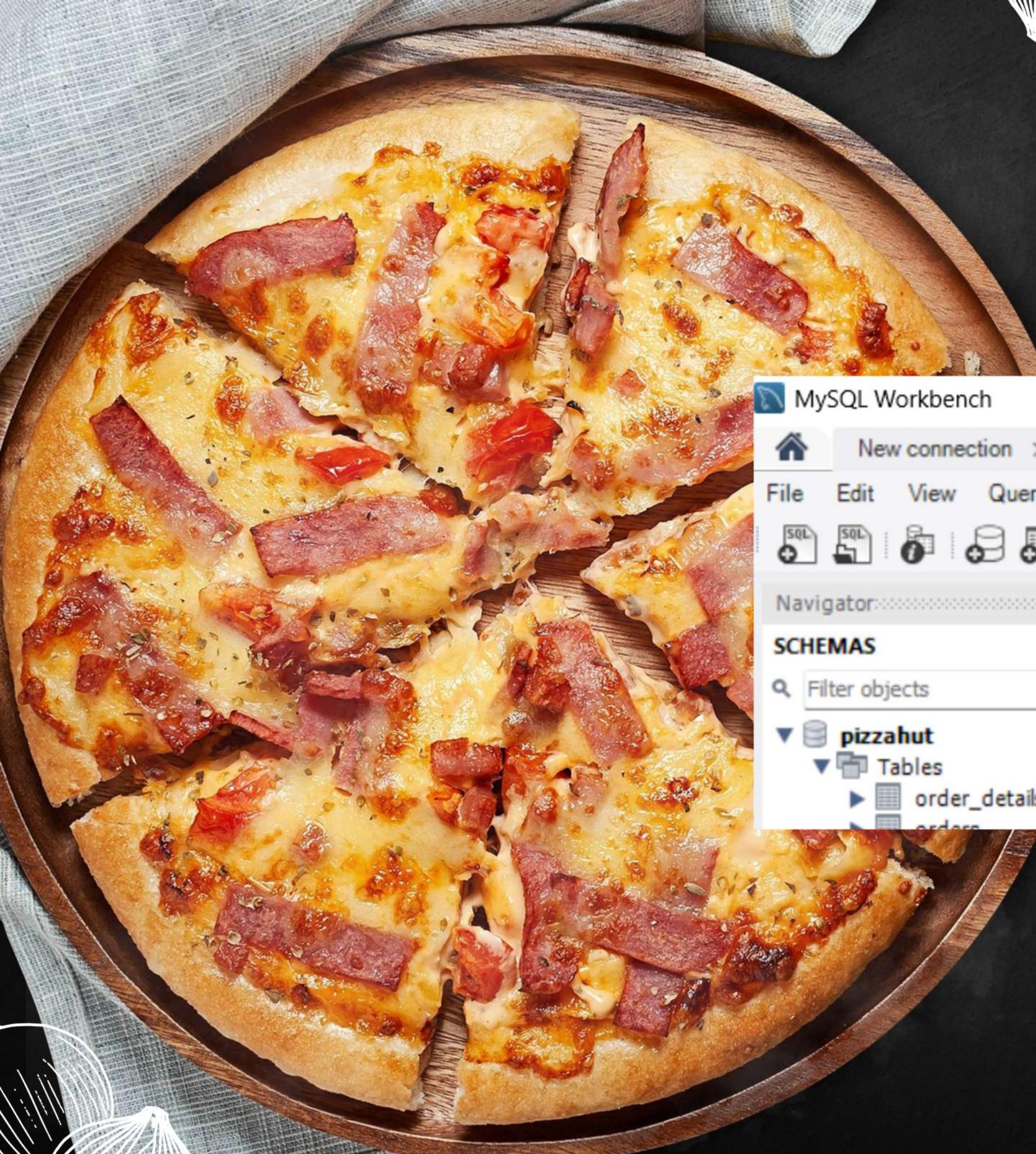
PIZZA SALES DATA ANALYSIS USING SQL



INTRODUCTION

- This project focuses on analyzing the Pizza sales dataset using SQL to understand business performance.
- It explores sales, orders, pizza categories, size and revenue contribution.
- Interconnected tables such as Orders, Order_Details, Pizza and Pizza_Types simulate real-world sales workflows.
- SQL queries are used to extract, transform and analyze sales pattern and performance metrics.
- The project strengthens concepts of database designs, joins, normalization and analytical SQL.
- Through analysis insights are gained and revenue distribution, peak order times, and best-selling pizzas.
- The overall goal is to support business decision using data-driven insights.





1.RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

MySQL Workbench

New connection X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

pizzahut

Tables

order_details

orders

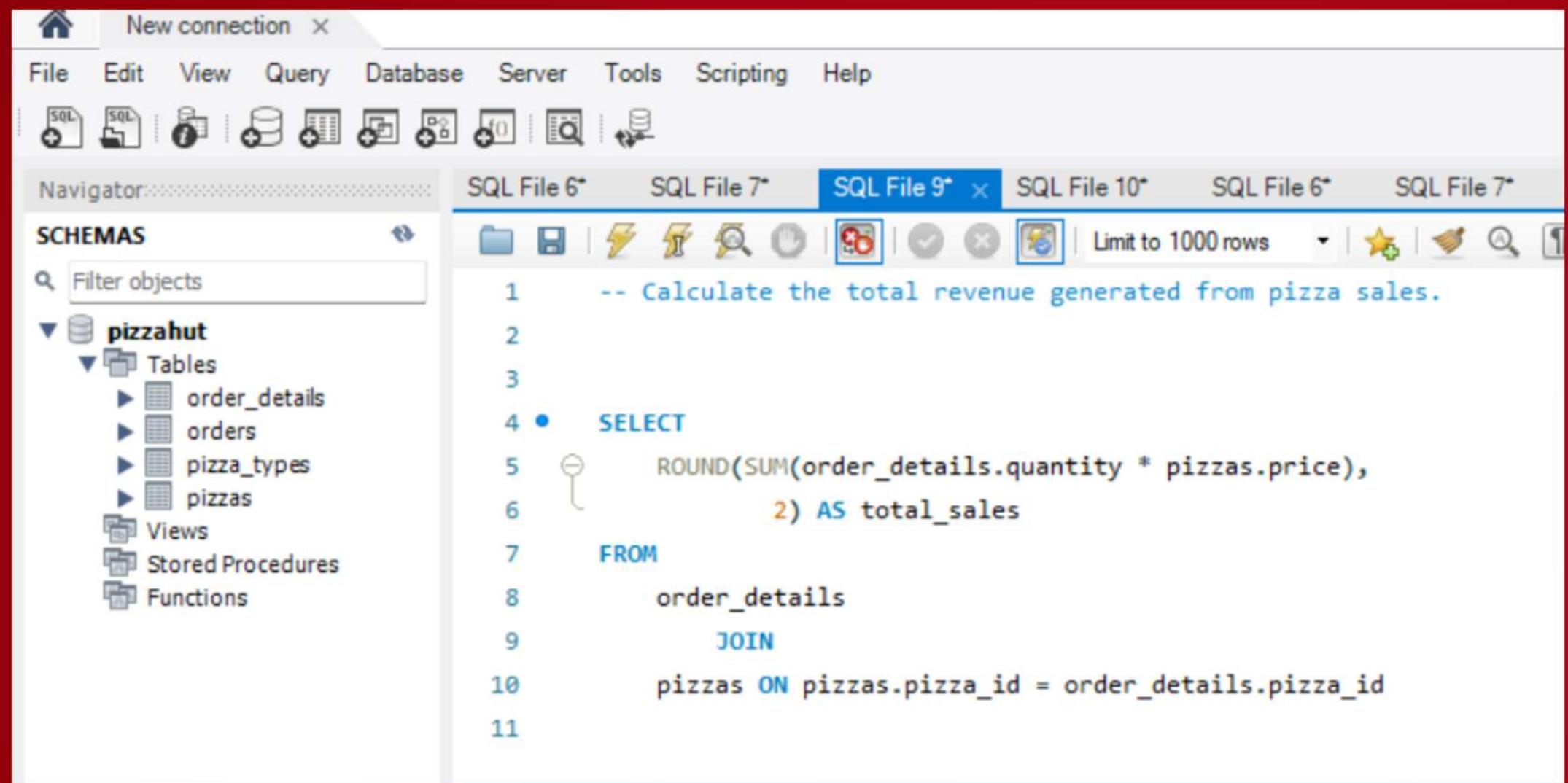
SQL File 6* SQL File 7* SQL File 9* SQL File 10* SQL File 6* SQL File 10*

1 -- Retrieve the total number of orders placed.
2 • SELECT COUNT(order_id) as total_orders FROM orders;

Result Grid | Filter Rows

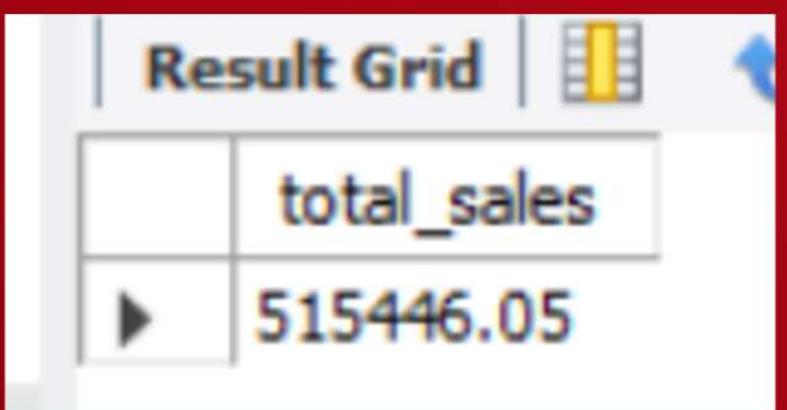
total_orders
21350

2.CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.



The screenshot shows a SQL query editor interface. On the left, the 'SCHEMAS' pane displays a database named 'pizzahut' with tables like 'order_details', 'orders', 'pizza_types', and 'pizzas'. The main area contains the following SQL code:

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



The screenshot shows the 'Result Grid' with one row containing the calculated total sales value.

total_sales
515446.05



3. IDENTIFY THE HIGHEST-PRICED PIZZA.

The screenshot shows a SQL query editor interface with a red header featuring a tomato slice graphic. The menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The toolbar contains various icons for database management. The main window has tabs for SQL File 6*, SQL File 7*, SQL File 9*, SQL File 10* (selected), SQL File 6*, SQL File 7*, and SQL File 8*. On the left, a Navigator pane shows the SCHEMAS section with a pizzahut schema expanded, revealing Tables (order_details, orders, pizza_types, pizzas), Views, Stored Procedures, and Functions. The central pane displays the following SQL code:

```
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;
```

The screenshot shows a Result Grid window with a red header featuring a tomato slice graphic. The grid has two columns: NAME and price. A single row is displayed, showing "The Greek Pizza" in the NAME column and "35.95" in the price column.

	NAME	price
▶	The Greek Pizza	35.95

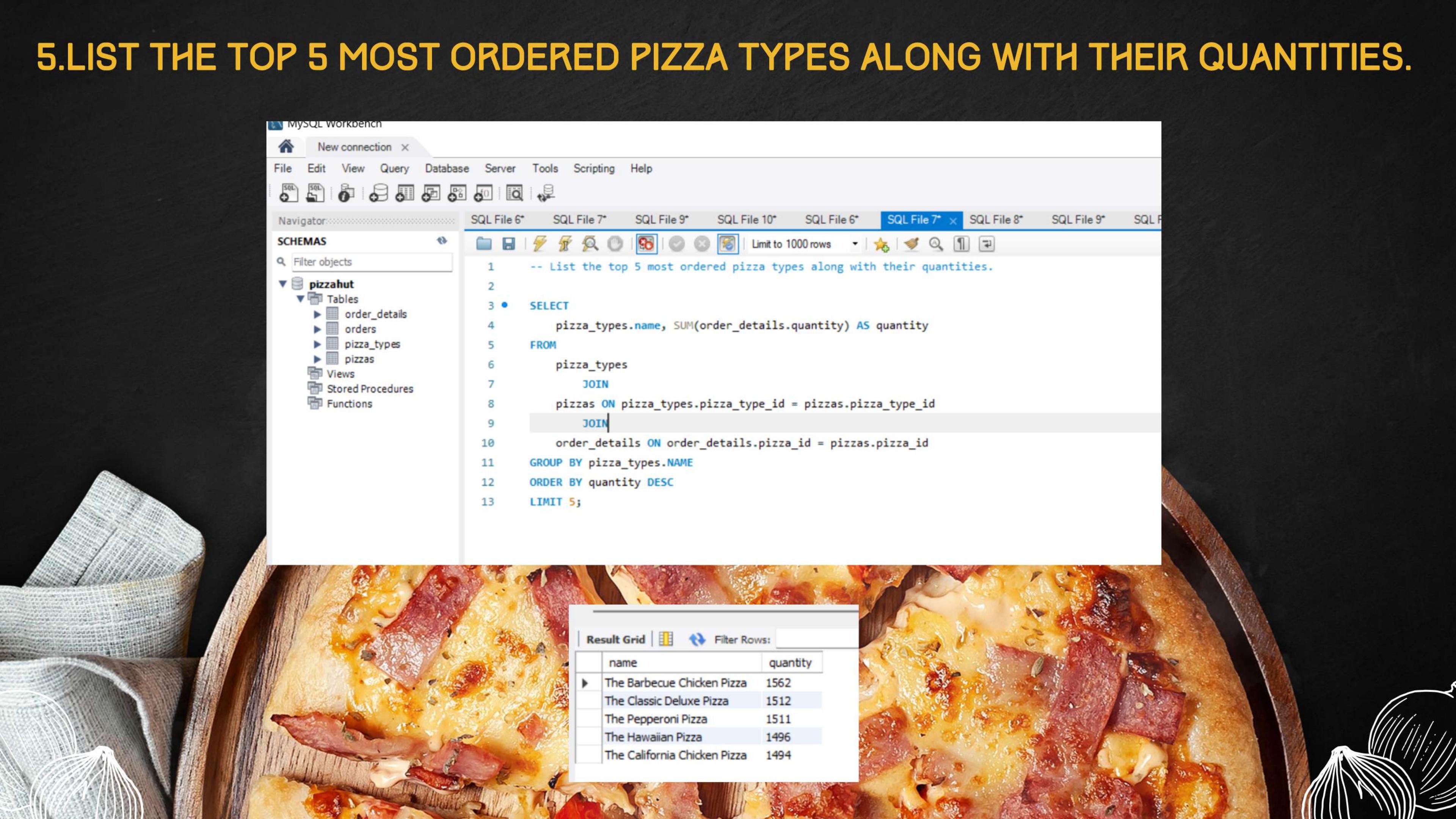
4. IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

```
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  
10 ON  
11     pizzas.pizza_id = order_details.pizza_id  
12 GROUP BY  
13     pizzas.size  
14 ORDER BY  
15     order_count DESC  
16 LIMIT  
17     0, 1000;
```

Result Grid	
size	order_count
L	11696
M	9707
S	8837
XL	357
XXL	18



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



MySQL Workbench

New connection X

File Edit View Query Database Server Tools Scripting Help

Navigator SQL File 6* SQL File 7* SQL File 9* SQL File 10* SQL File 6* SQL File 7* SQL File 8* SQL File 9* SQL File 10*

SCHEMAS Filter objects

pizzahut

Tables: order_details, orders, pizza_types, pizzas

Views

Stored Procedures

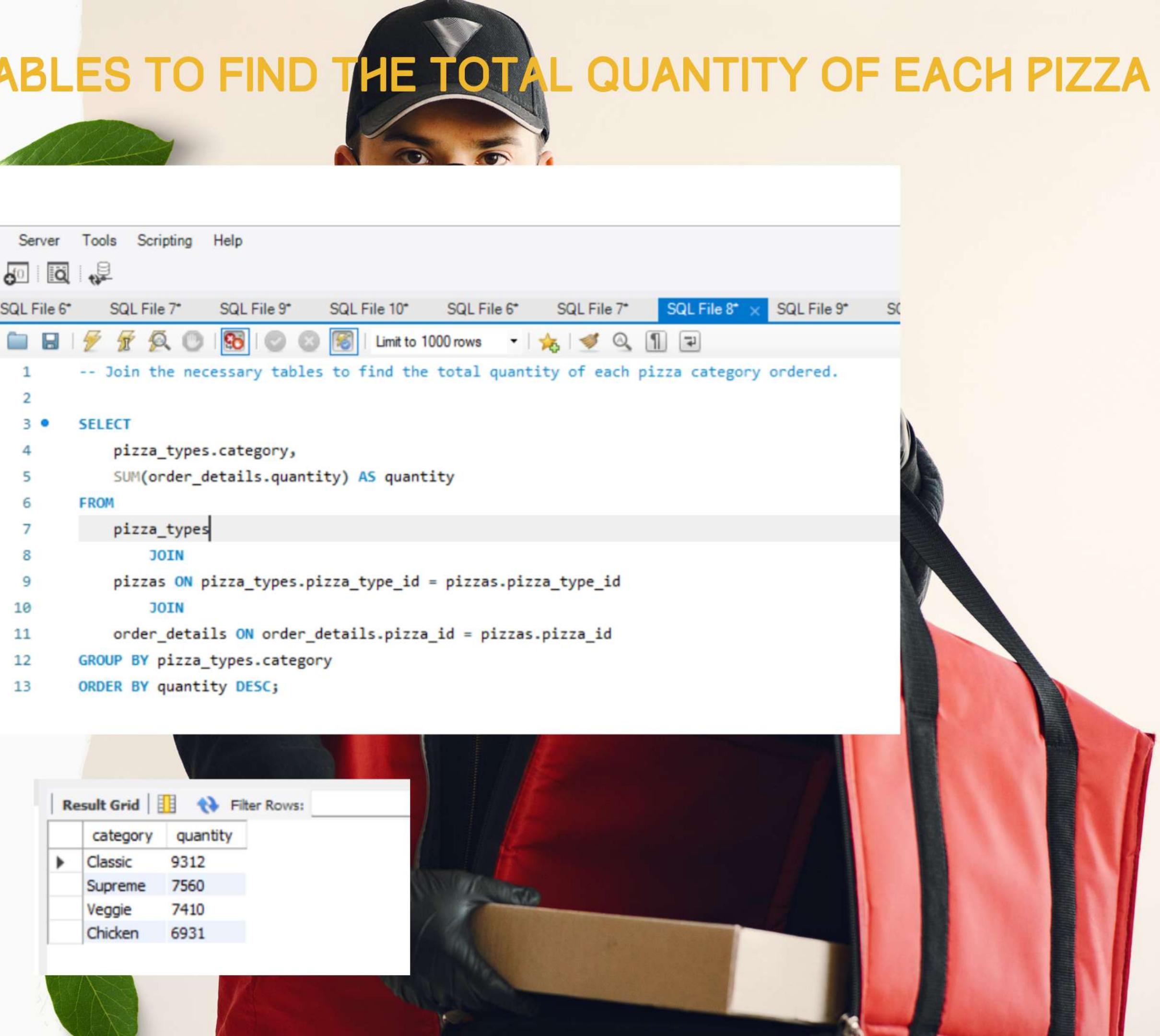
Functions

```
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:

	name	quantity
▶	The Barbecue Chicken Pizza	1562
	The Classic Deluxe Pizza	1512
	The Pepperoni Pizza	1511
	The Hawaiian Pizza	1496
	The California Chicken Pizza	1494

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



MySQL Workbench

New connection ×

File Edit View Query Database Server Tools Scripting Help

Navigator: SQL File 6* SQL File 7* SQL File 9* SQL File 10* SQL File 6* SQL File 7* SQL File 8* × SQL File 9* SQL File 10*

SCHEMAS

Filter objects

pizzahut

- Tables
 - order_details
 - orders
 - pizza_types
 - pizzas
- Views
- Stored Procedures
- Functions

```
1 -- Join the necessary tables 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 | Filter Rows:

	category	quantity
▶	Classic	9312
	Supreme	7560
	Veggie	7410
	Chicken	6931



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

Mysql Workbench

New connection ×

File Edit View Query Database Server Tools Scripting Help

SQL File 6* SQL File 7* SQL File 9* SQL File 10* SQL File 6* SQL File 7* SQL File 9* SQL File 10*

Navigator

SCHEMAS

Filter objects

pizzahut

Tables

- order_details
- orders
- pizza_types
- pizzas

Views

Stored Procedures

Functions

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

Result Grid | Filter Rows: _____

HOUR(order_time)	order_count
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920

Result 1 ×

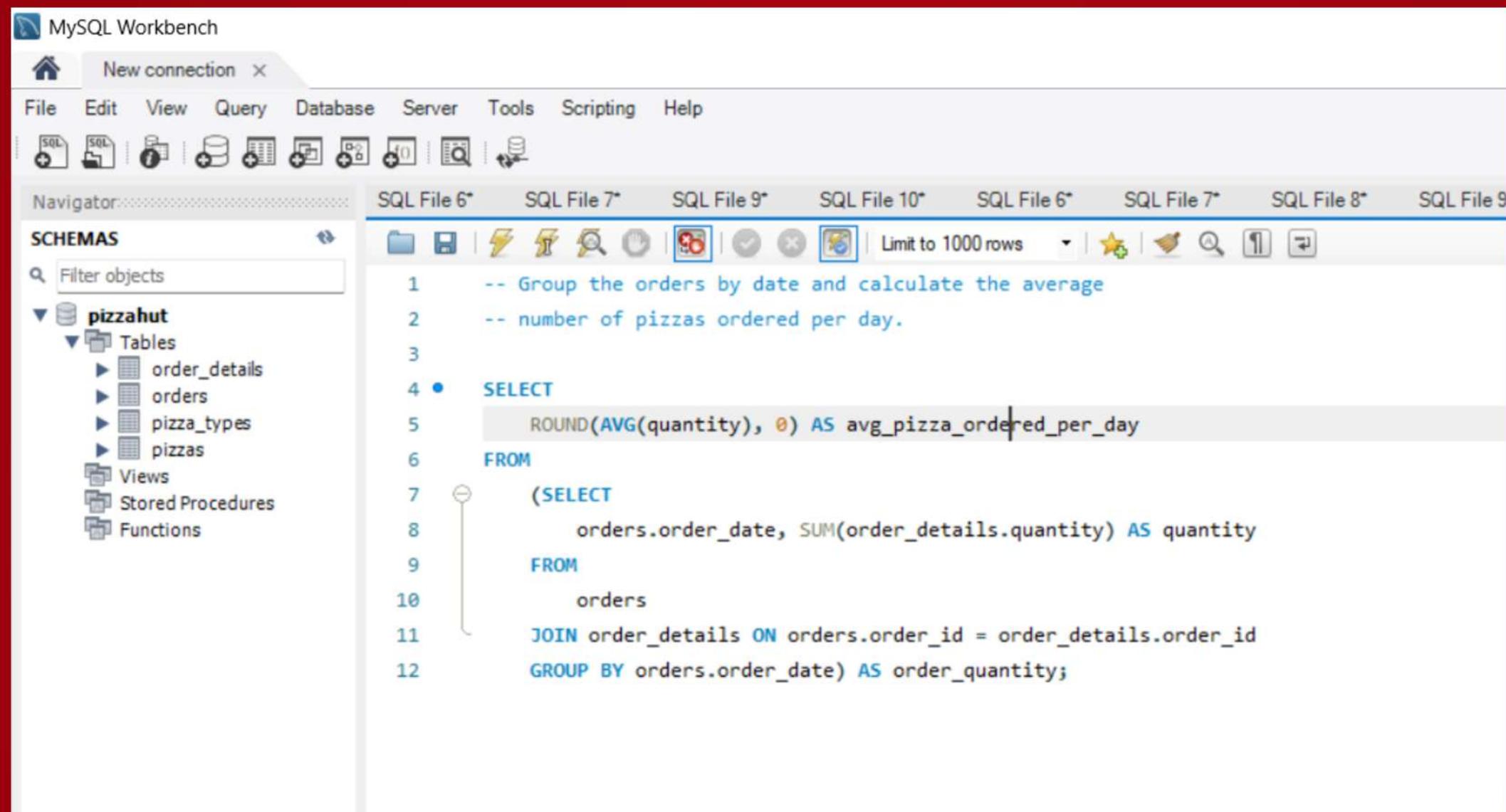


8.JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

```
use Server Tools Scripting Help  
SQL File 6* SQL File 7* SQL File 9* SQL File 10* SQL File 6* SQL File 7* SQL File 8* S  
1 -- Join relevant tables to find the category-wise distribution of pizzas  
2  
3 • SELECT category , COUNT(NAME) FROM pizza_types  
4 GROUP BY category;
```

Result Grid		Filter Rows:
	category	COUNT(NAME)
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

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



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree, with 'pizzahut' selected, revealing tables like 'order_details', 'orders', 'pizza_types', and 'pizzas'. The main area is the 'SQL File 6*' tab, containing the following SQL code:

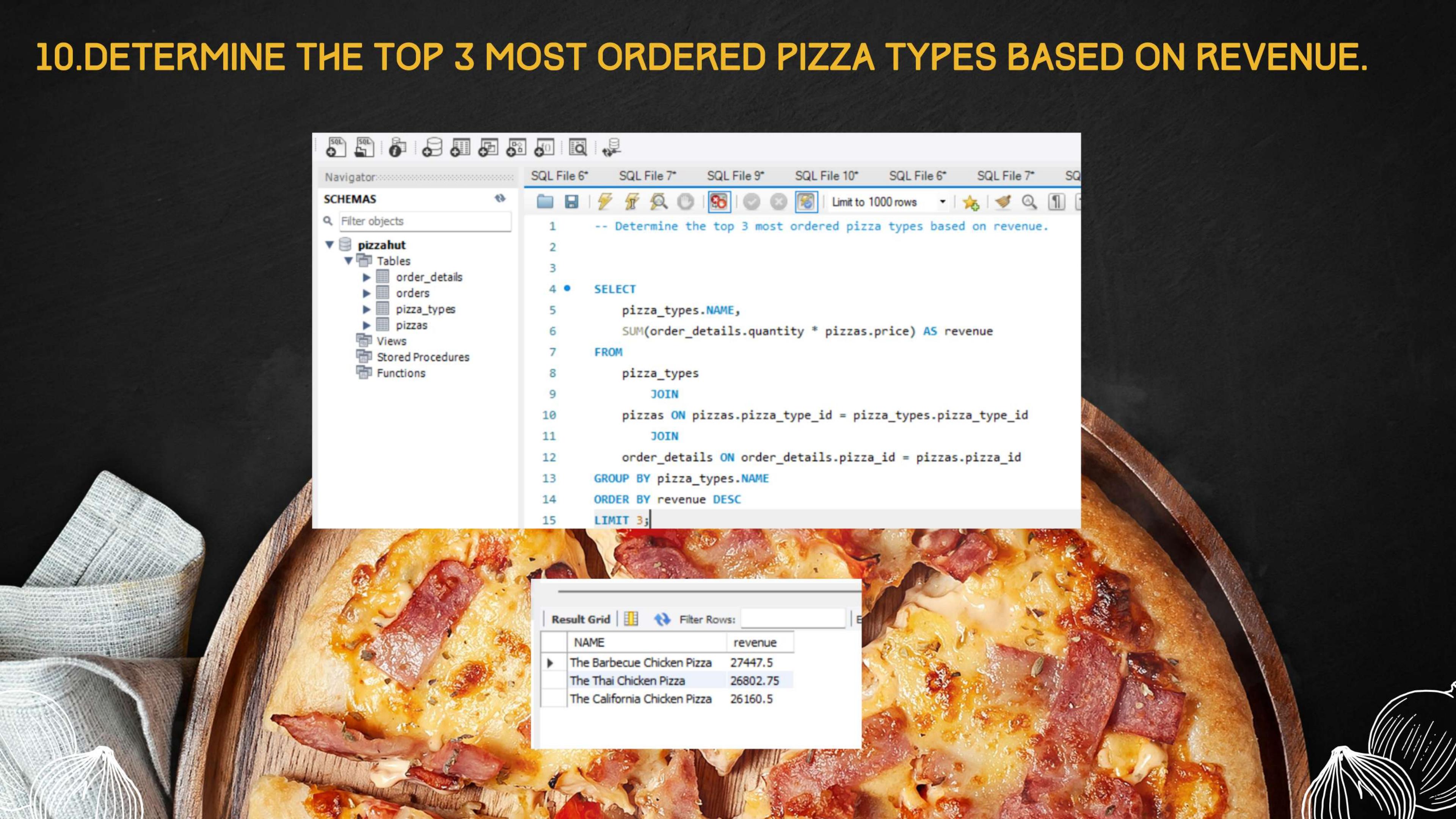
```
1 -- Group the orders by date and calculate the average
2 -- number of pizzas ordered per day.
3
4 • SELECT
5     ROUND(AVG(quantity), 0) AS avg_pizza_ordered_per_day
6 FROM
7     (SELECT
8         orders.order_date, SUM(order_details.quantity) AS quantity
9     FROM
10        orders
11    JOIN order_details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



A small window titled 'Result Grid' shows the output of the query. It contains a single row with one column:

avg_pizza_ordered_per_day
138

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



The image shows a large, delicious-looking pizza with various toppings like pepperoni, cheese, and onions, served in a wooden tray. In the background, there's a white cloth napkin and some garlic cloves.

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

NAME	revenue
The Barbecue Chicken Pizza	27447.5
The Thai Chicken Pizza	26802.75
The California Chicken Pizza	26160.5



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

New connection ×

File Edit View Query Database Server Tools Scripting Help

Navigator: SQL File 6* SQL File 7* SQL File 9* SQL File 10* SQL File 6* SQL File 7* SQL File 8* SQL File 9* SQL File 10*

SCHEMAS

Filter objects

pizzahut

Tables

- order_details
- orders
- pizza_types
- pizzas

Views

Stored Procedures

Functions

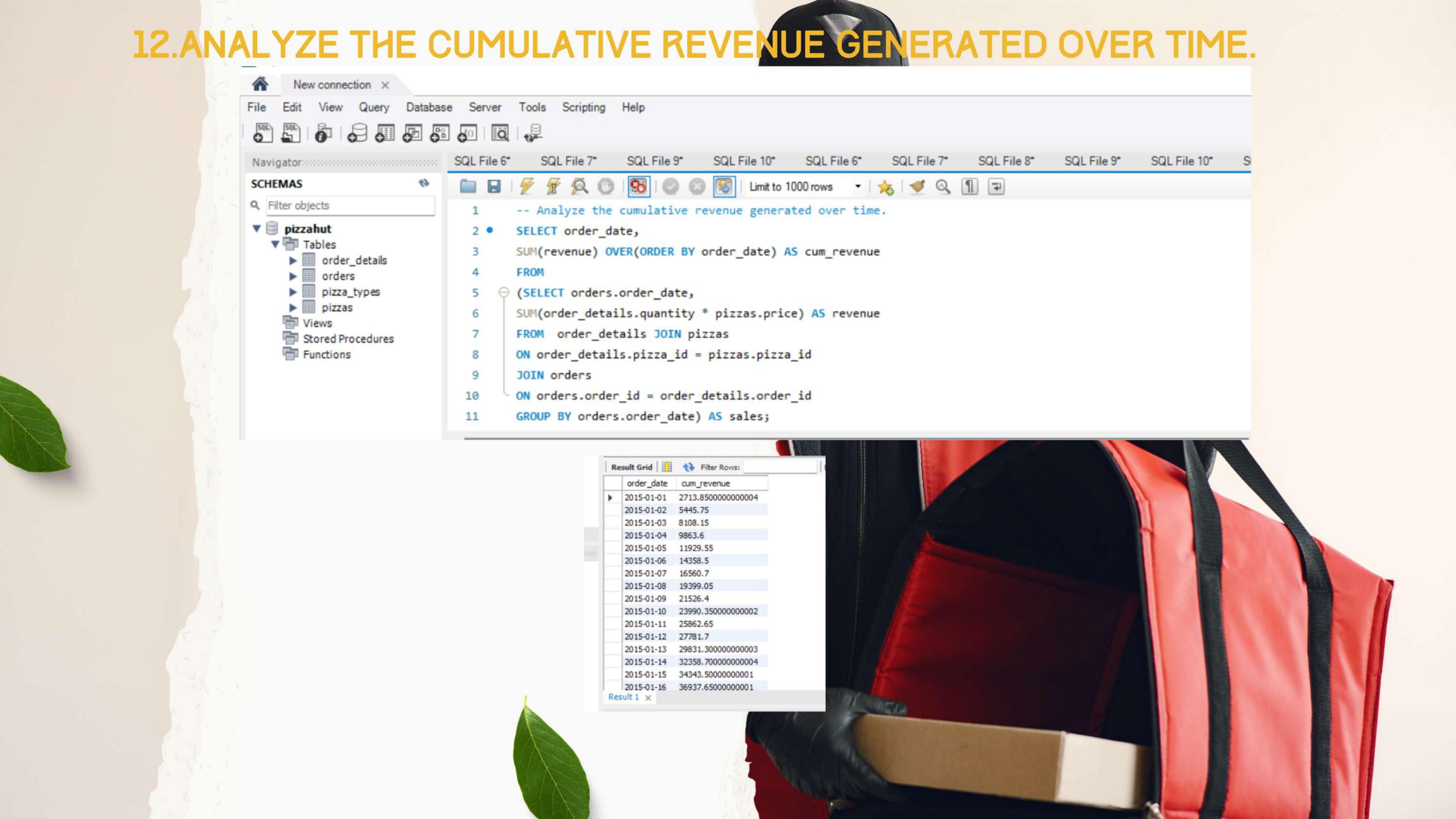
```
1 -- Calculate the percentage contribution of each pizza type to total revenue.
2 • SELECT
3     pizza_types.category,
4     ROUND(
5         (SUM(order_details.quantity * pizzas.price) /
6             (SELECT SUM(order_details.quantity * pizzas.price)
7                 FROM order_details
8     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

No object selected

Result Grid | Filter Rows:

	category	revenue_percentage
▶	Classic	26.73
	Supreme	25.52
	Veggie	23.92
	Chicken	23.83

12. ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.



A screenshot of a SQL management tool interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. Below the menu is a toolbar with various icons. The main area shows a query editor with the following SQL code:

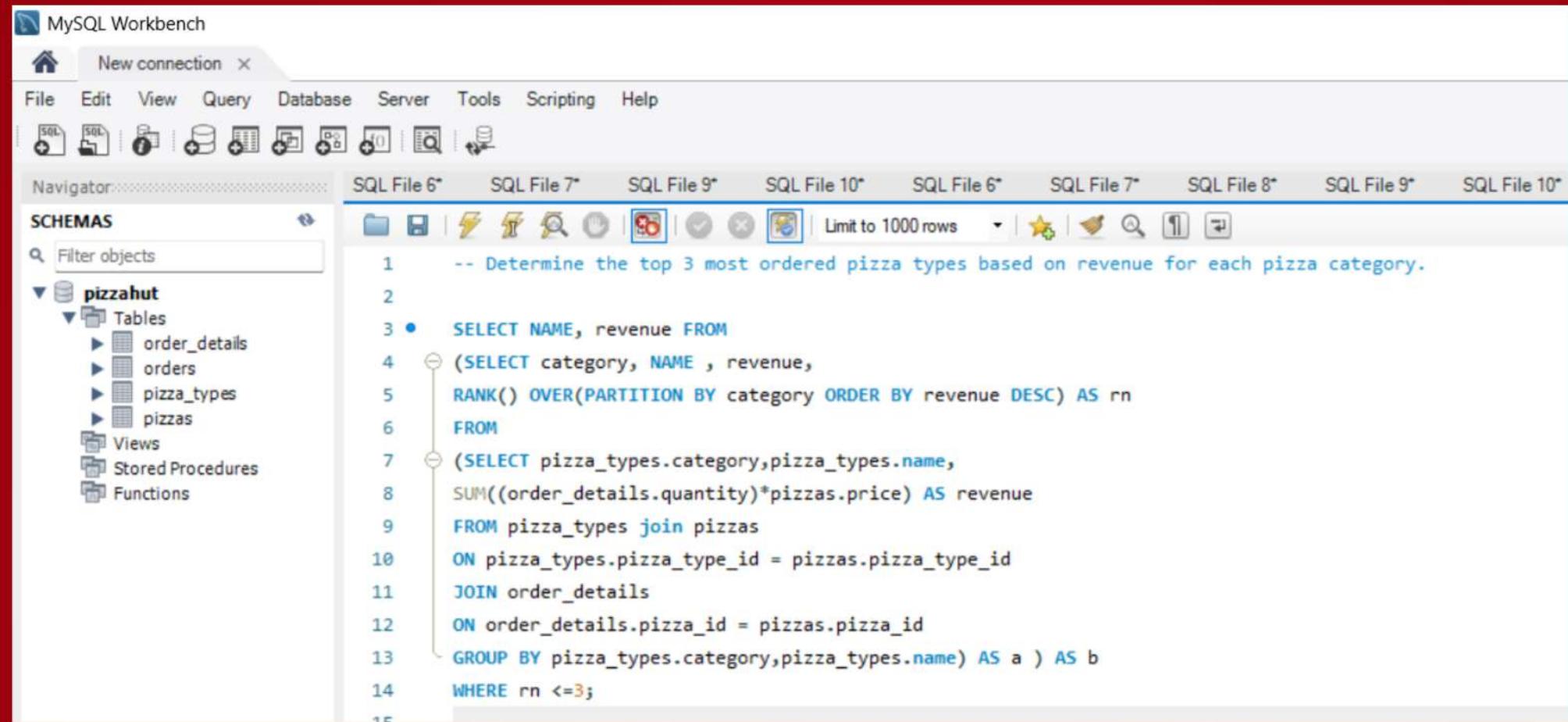
```
1 -- Analyze the cumulative revenue generated over time.
2 • SELECT order_date,
3     SUM(revenue) OVER(ORDER BY order_date) AS cum_revenue
4 FROM
5     (SELECT orders.order_date,
6         SUM(order_details.quantity * pizzas.price) AS revenue
7     FROM order_details JOIN pizzas
8     ON order_details.pizza_id = pizzas.pizza_id
9     JOIN orders
10    ON orders.order_id = order_details.order_id
11    GROUP BY orders.order_date) AS sales;
```

Result Grid | Filter Rows:

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
2015-01-09	21526.4
2015-01-10	23990.35000000002
2015-01-11	25862.65
2015-01-12	27781.7
2015-01-13	29831.30000000003
2015-01-14	32358.70000000004
2015-01-15	34343.50000000001
2015-01-16	36937.65000000001

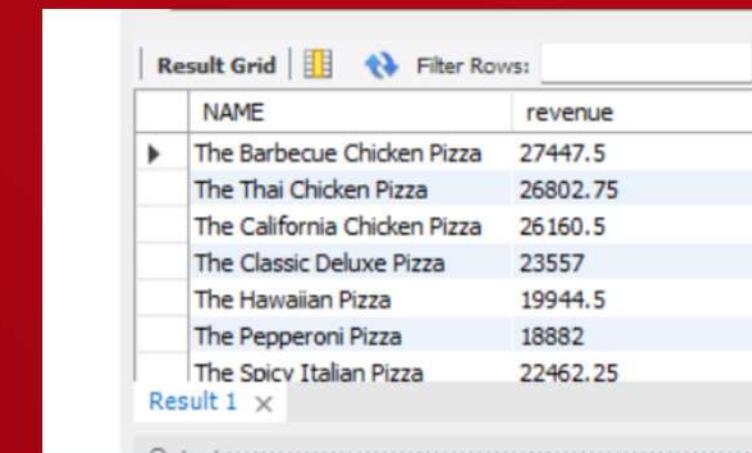
Result 1 ×

13. DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.



The screenshot shows the MySQL Workbench interface. The left pane displays the 'Navigator' with the 'SCHEMAS' section expanded, showing the 'pizzahut' schema with its tables: order_details, orders, pizza_types, and pizzas. The right pane contains a SQL editor window with the following query:

```
1 -- Determine the top 3 most ordered pizza types based on revenue for each pizza category.
2
3 • SELECT NAME, revenue FROM
4   (SELECT category, NAME , revenue,
5    RANK() OVER(PARTITION BY category ORDER BY revenue DESC) AS rn
6    FROM
7    (SELECT pizza_types.category,pizza_types.name,
8     SUM((order_details.quantity)*pizzas.price) AS revenue
9     FROM pizza_types join pizzas
10    ON pizza_types.pizza_type_id = pizzas.pizza_type_id
11   JOIN order_details
12   ON order_details.pizza_id = pizzas.pizza_id
13  GROUP BY pizza_types.category,pizza_types.name) AS a ) AS b
14 WHERE rn <=3;
15
```



The screenshot shows the 'Result Grid' from MySQL Workbench. The results are as follows:

NAME	revenue
The Barbecue Chicken Pizza	27447.5
The Thai Chicken Pizza	26802.75
The California Chicken Pizza	26160.5
The Classic Deluxe Pizza	23557
The Hawaiian Pizza	19944.5
The Pepperoni Pizza	18882
The Spicy Italian Pizza	22462.25



THANKS

e-mail:shivamraut890@gmail.com

[LINKEDIN.COM/IN/SHIVAM--RAUT](https://www.linkedin.com/in/shivam--raut)

