WWW.REALLYGREATSITE.COM

PIZZAHUT

Sneha Ghosh

23.07.2025

ABOUT OUR RESTAURANT

This SQL project analyzes Pizza Hut sales data to uncover key business insights using MySQL. It covers basic to advanced SQL queries including aggregations, joins, groupings, and window functions. Key questions addressed include total orders, revenue, most popular pizza types, order trends by hour, and top-performing categories. Advanced analysis includes cumulative revenue over time and ranking pizzas by revenue within categories. The project demonstrates practical SQL skills for data analysis and lays the foundation for future dashboard development in Power BI.

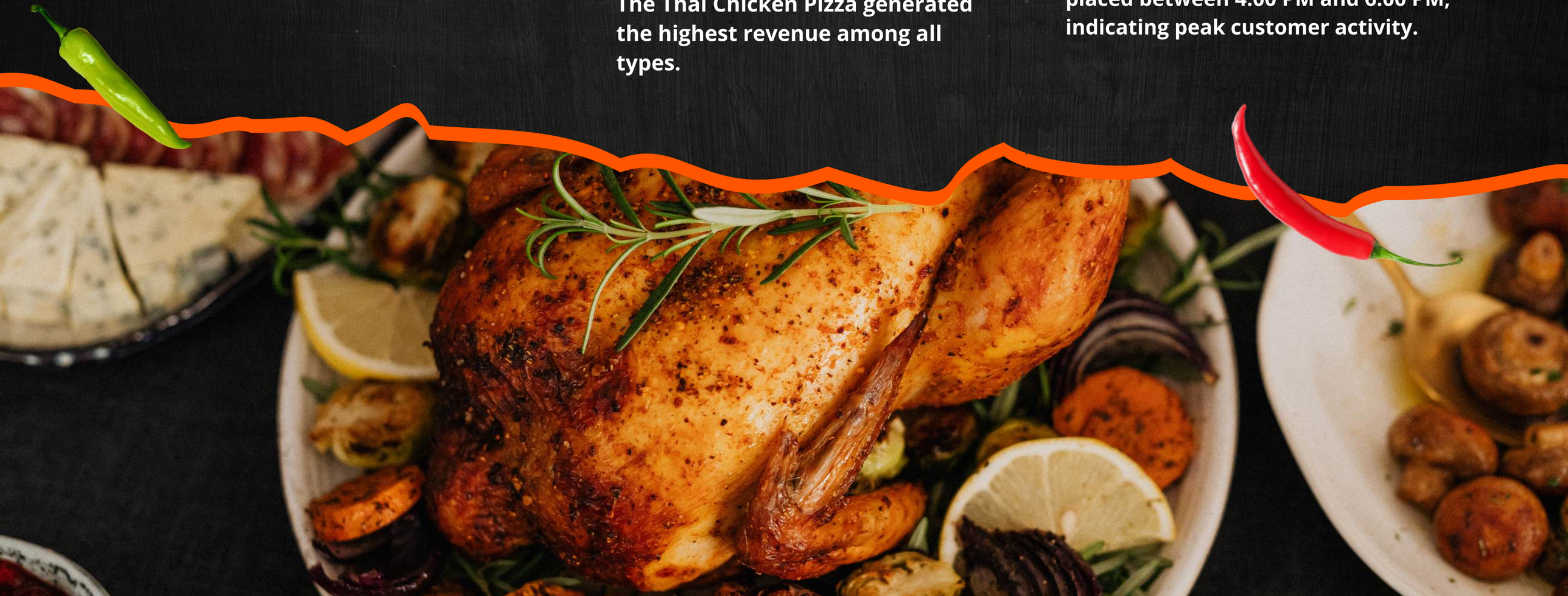
OUR BEST SELLER MENU

- Retrieve the total number of orders placed
- Calculate the total revenue generated from pizza sales
- Identify the highest-priced pizza
- Identify the most common pizza size ordered
- List the top 5 most ordered pizza types along with their quantities
- Join the necessary tables to find the total quantity of each pizza category ordered
- Determine the distribution of orders by hour of the day
- Join relevant tables to find the category-wise distribution of pizzas
- Group the orders by date and calculate the average number of pizzas ordered per day
- Determine the top 3 most ordered pizza types based on revenue
- Calculate the percentage contribution of each pizza type to total revenue
- Analyze the cumulative revenue generated over time
- Determine the top 3 most ordered pizza types based on revenue for each pizza category

Order Now

ABOUT RESTAURANT

- 🍕 **Most Popular Pizza**: The Classic Deluxe Pizza was the most ordered item across all orders.
-
- 💰 **Top Revenue-Generating Pizza:** The Thai Chicken Pizza generated the highest revenue among all types.
- 🏆 **Best Performing Category:** The Classic category earned the most total revenue across all pizza categories.
- ⏰ **Peak Order Time:** Most orders were placed between 4:00 PM and 6:00 PM, indicating peak customer activity.



OUR VISION

- Set up MySQL Server and Workbench from scratch
- Used JOIN, GROUP BY, and ORDER BY to analyze sales data
- Applied aggregate functions (SUM, COUNT, AVG) for insights
- Used time functions to identify peak order hours
- Implemented RANK() and OVER() for advanced revenue analysis
- Gained hands-on experience turning raw data into business insights

RETRIEVE THE TOTAL NUMBERS OF ORDERS PLACED.

```sql
USE PizzaHut;

SELECT
    COUNT(order_id) AS total_orders
FROM
    orders;
```

Result Grid

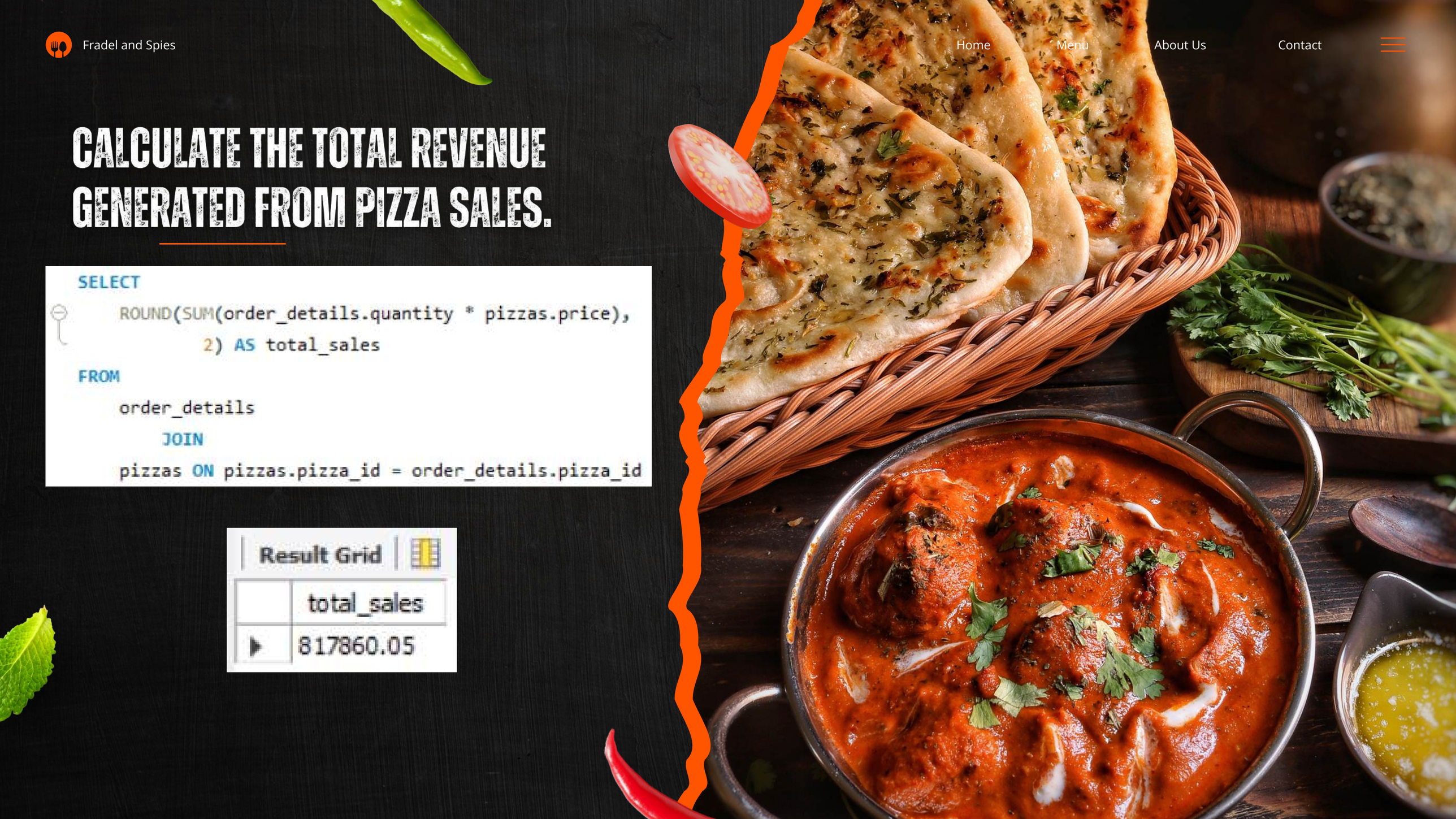
	total_orders
▶	21350

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
```

Result Grid

	total_sales
▶	817860.05

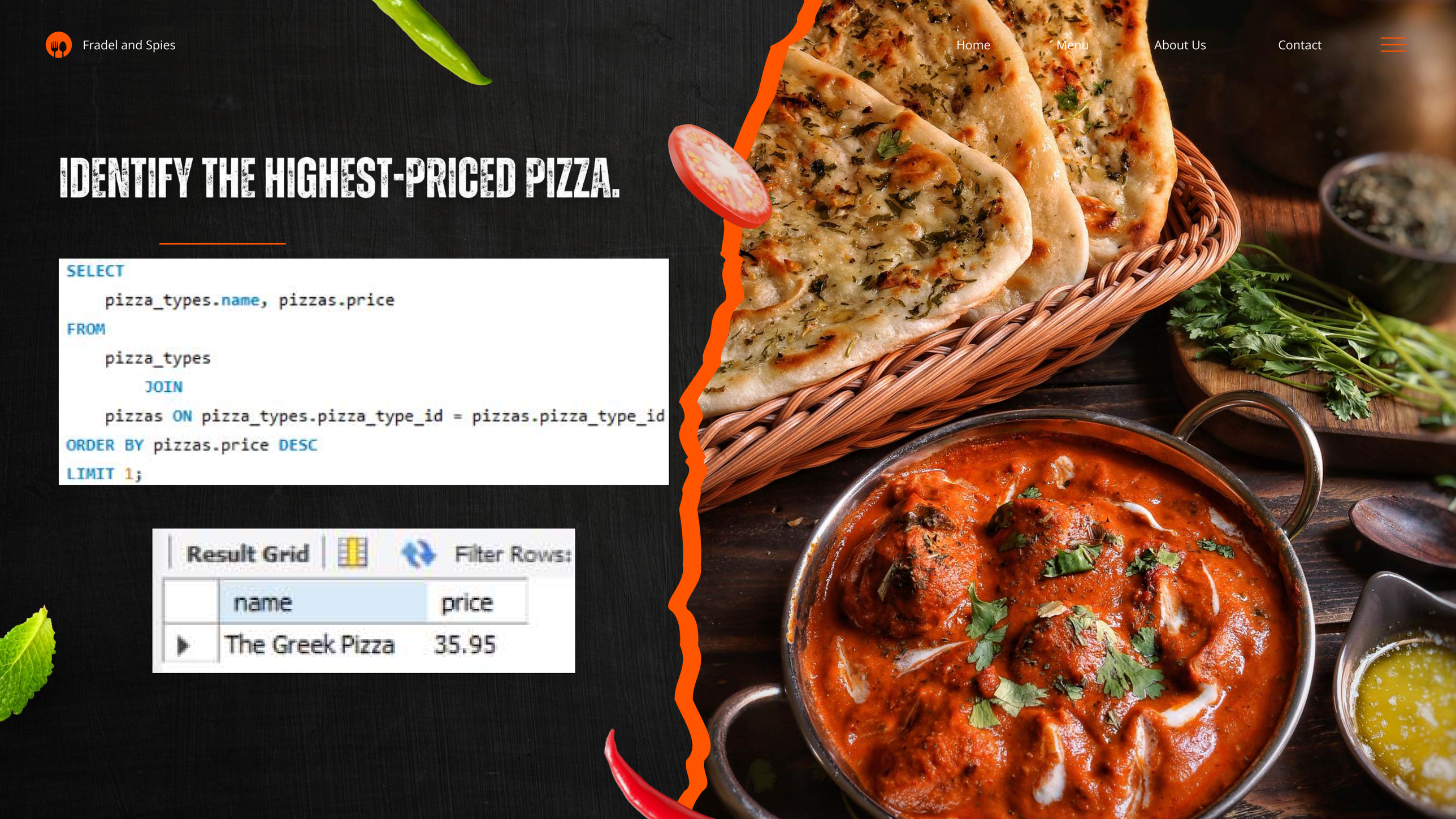


IDENTIFY THE HIGHEST-PRICED PIZZA.

```
SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
```

Result Grid | Filter Rows:

name	price
The Greek Pizza	35.95



IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

```sql
SELECT
    pizzas.size,
    COUNT(order_details.order_details_id) AS order_count
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC;
```

Result Grid | Filter

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.

```sql
SELECT
    pizza_types.name,
    SUM(order_details.quantity) AS total_quantity
FROM
    pizza_types
JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY
    pizza_types.name
ORDER BY
    total_quantity DESC
LIMIT 5;
```

Result Grid | Filter Rows:

name	total_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.

```sql
SELECT
    pizza_types.category,
    SUM(order_details.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity DESC;
```

Result Grid | Filter

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

```sql
SELECT
    HOUR(date) AS order_hour,
    COUNT(order_id) AS total_orders
FROM
    orders
GROUP BY
    HOUR(date)
ORDER BY
    order_hour;
```

Result Grid | Filter Rows:

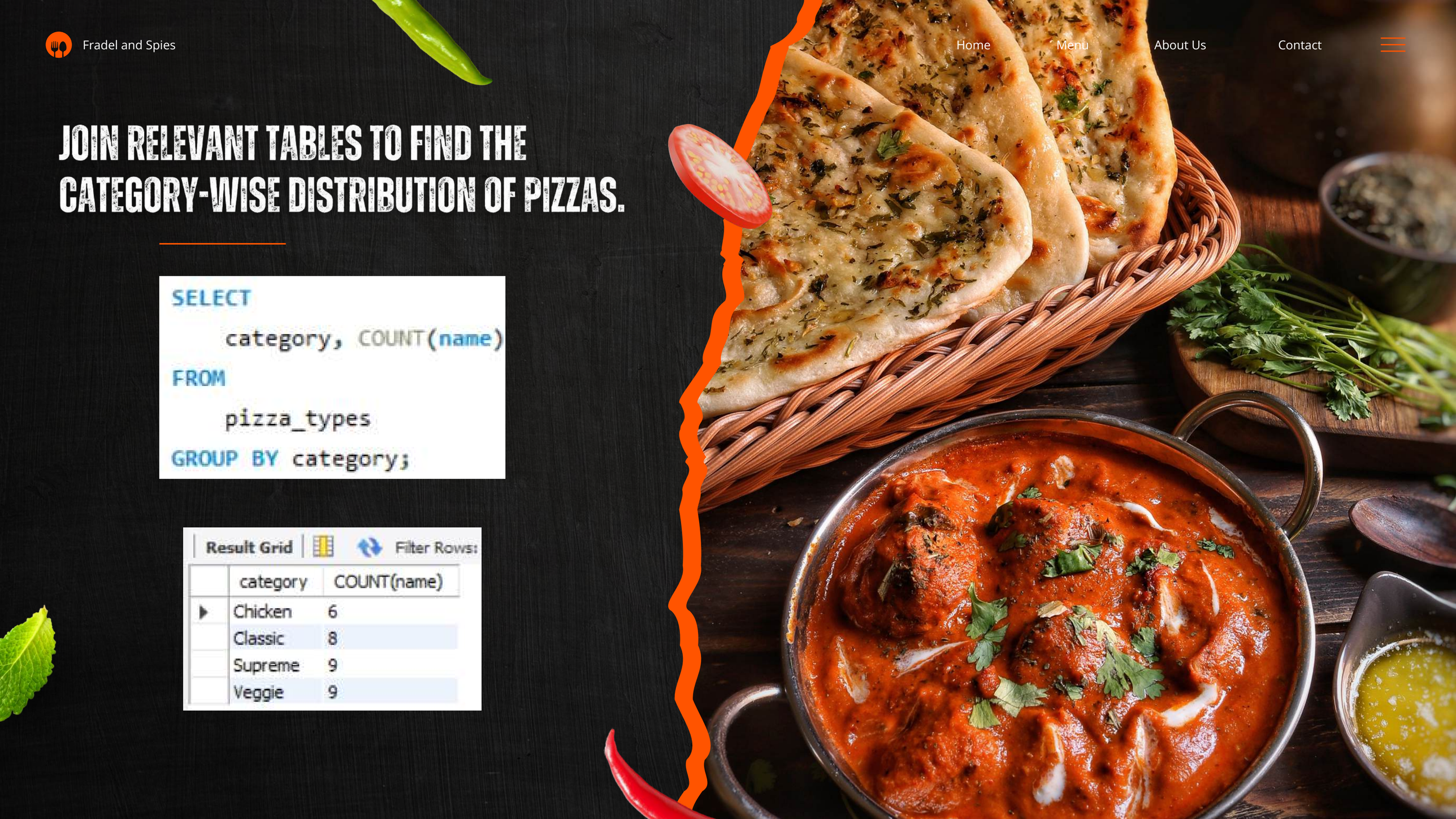
order_hour	total_orders
0	21350

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

```
SELECT
    category, COUNT(name)
FROM
    pizza_types
GROUP BY category;
```

Result Grid | Filter Rows:

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

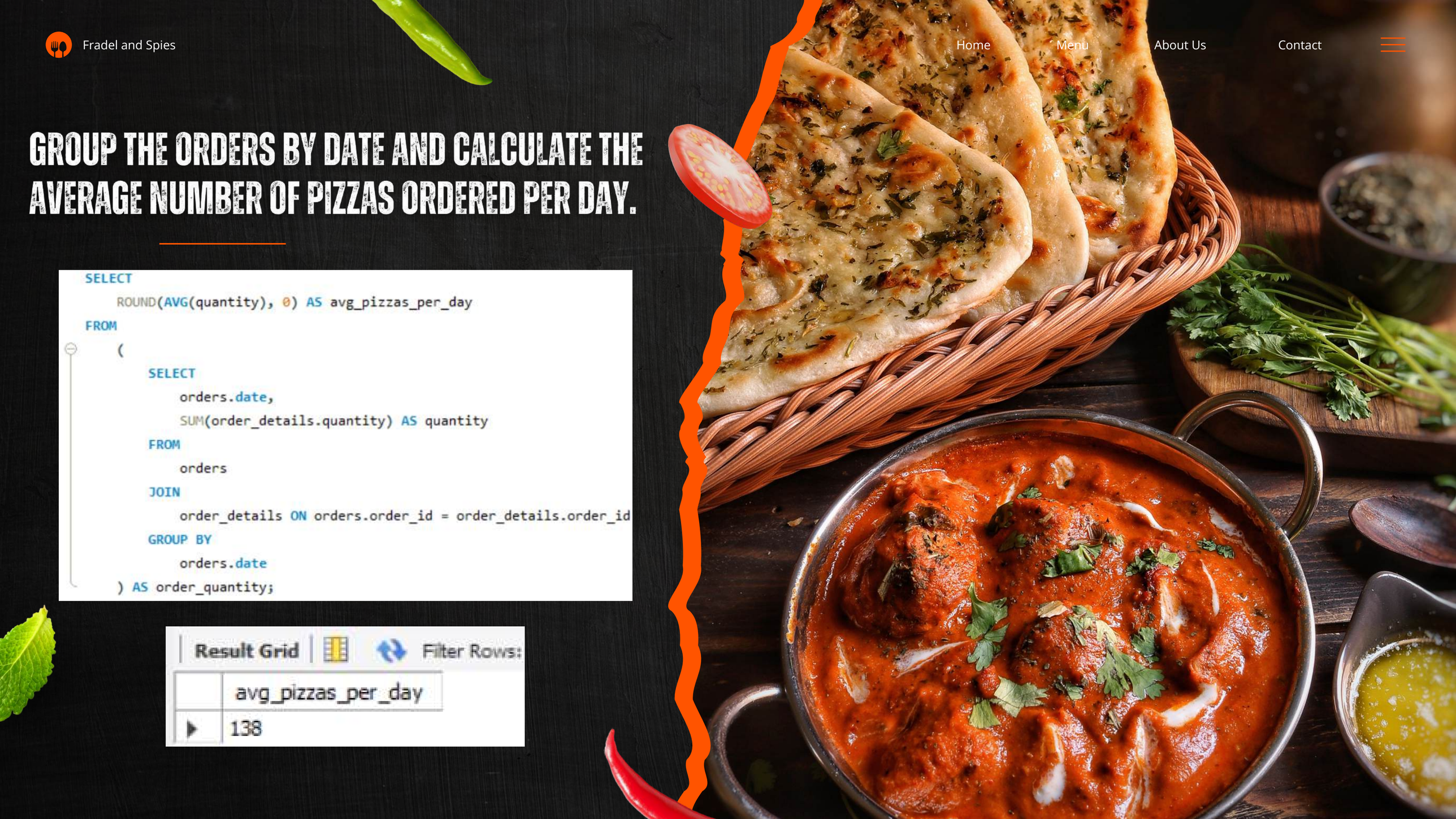


GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

```sql
SELECT
    ROUND(AVG(quantity), 0) AS avg_pizzas_per_day
FROM
    (
        SELECT
            orders.date,
            SUM(order_details.quantity) AS quantity
        FROM
            orders
        JOIN
            order_details ON orders.order_id = order_details.order_id
        GROUP BY
            orders.date
    ) AS order_quantity;
```

Result Grid | Filter Rows:

	avg_pizzas_per_day
▶	138



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

```sql
SELECT
    pizza_types.name,
    SUM(order_details.quantity * pizzas.price) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
        JOIN
    order_details ON order_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

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

```sql
SELECT
    pizza_types.category,
    ROUND(SUM(order_details.quantity * pizzas.price) / (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) * 100,
            2) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY revenue DESC;
```

Result Grid | Filter

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```sql
SELECT
    order_date,
    SUM(revenue) OVER (ORDER BY order_date) AS cum_revenue
FROM (
    SELECT
        orders.date AS order_date,
        SUM(order_details.quantity * pizzas.price) AS revenue
    FROM
        order_details
    JOIN
        pizzas ON order_details.pizza_id = pizzas.pizza_id
    JOIN
        orders ON orders.order_id = order_details.order_id
    GROUP BY
        orders.date
) AS sales;
```

Result Grid | Filter Rows:

order_date	cum_revenue
2015-01-01 00:00:00	2713.8500000000004
2015-01-02 00:00:00	5445.75
2015-01-03 00:00:00	8108.15
2015-01-04 00:00:00	9863.6
2015-01-05 00:00:00	11929.55
2015-01-06 00:00:00	14358.5
2015-01-07 00:00:00	16560.7
2015-01-08 00:00:00	19399.05
2015-01-09 00:00:00	21526.4
2015-01-10 00:00:00	23990.350000000002
2015-01-11 00:00:00	25862.65
2015-01-12 00:00:00	27781.7
2015-01-13 00:00:00	29831.300000000003
2015-01-14 00:00:00	32358.700000000004
2015-01-15 00:00:00	34343.50000000001
2015-01-16 00:00:00	36937.65000000001
2015-01-17 00:00:00	39001.75000000001
2015-01-18 00:00:00	40978.600000000006
2015-01-19 00:00:00	43365.75000000001
2015-01-20 00:00:00	45763.65000000001
2015-01-21 00:00:00	47804.20000000001
2015-01-22 00:00:00	50300.90000000001

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

```sql
SELECT
    name,
    revenue
FROM (
    SELECT
        category,
        name,
        revenue,
        RANK() OVER (PARTITION BY category ORDER BY revenue DESC) AS rn
    FROM (
        SELECT
            pizza_types.category,
            pizza_types.name,
            SUM(order_details.quantity * pizzas.price) AS revenue
        FROM
            pizza_types
        JOIN
            pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
            order_details ON order_details.pizza_id = pizzas.pizza_id
        GROUP BY
            pizza_types.category, pizza_types.name
    ) AS a
) AS b
WHERE rn <= 3;
```

Result Grid | Filter Rows:

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

ALL MENUS IN OUR RESTAURANT

This SQL project gave me hands-on experience in analyzing real-world sales data, deriving business insights, and strengthening my SQL skills — from basic queries to advanced analytics. It helped me understand how data drives decision-making in the food industry.

More Menu






MEET OUR TEAM

SQL

MySQL

MySQL Workbench

Ms Excel

Canva

LET'S CONNECT:

Feel free to reach out or connect with me on LinkedIn to discuss data, projects, or collaboration opportunities!

https://www.linkedin.com/in/sneha-ghosh-98aaa9337

THANK YOU