

Pizza Sales Data Analysis using SQL

[Home](#)

[About](#)

[Contact](#)

Transforming raw sales data into actionable insights

This project analyzes pizza sales data using SQL to uncover actionable business insights.

The analysis focuses on understanding **sales** performance, identifying **top-selling** and high-revenue pizzas, analyzing order trends over time, and evaluating **category-wise** revenue contribution to support data-driven business decisions.



Tools & Skills:

SQL | Joins | Aggregations | Window Functions | Business Analysis

Dataset Description

The project utilizes a relational database consisting of **four primary tables**:

- **orders**
 - **order_id**
 - **order_date**
 - **order_time**
- **order_details**
 - **order_details_id**
 - **order_id**
 - **pizza_id**
 - **quantity**
- **pizzas**
 - **pizza_id**
 - **pizza_type_id**
 - **size**
 - **price**
- **pizza_types**
 - **pizza_type_id**
 - **name**
 - **category**
 - **ingredients**





CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

About

Contact



SQL File 6* SQL File 7* x SQL File 8* SQL File 9* SQL File 10* SQL File 11*

Folder | Back | Refresh | Find | Search | Stop | Refresh | X | Help | Limit to 1000 rows | Star | Add to Favorites | Print | Find in Results | Search

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

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

total_sales
27664.7



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

The screenshot shows a MySQL query editor interface. At the top, there are tabs for "SQL File 6*", "SQL File 7*", "SQL File 8*", "SQL File 9*", "SQL File 10*" (which is selected), and "SQL File 11*". Below the tabs is a toolbar with various icons for file operations, search, and database management. The main area contains the following SQL code:

```
1  -- List the top 5 most ordered pizza types along with their quantities.
2  •  SELECT
3      pizza_types.name,
4          SUM(order_details.quantity) AS quantity
5  FROM pizza_types
6  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.name
11 ORDER BY quantity DESC
12 LIMIT 5;
13
```

Below the code, there is a "Result Grid" section with a table showing the results:

	name	quantity
▶	The Pepperoni Pizza	99
	The Classic Deluxe Pizza	80
	The Barbecue Chicken Pizza	80
	The Thai Chicken Pizza	78
	The Italian Supreme Pizza	76



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



```
3 • SELECT
4     pizza_types.category,
5     SUM(order_details.quantity) AS quantity
6   FROM pizza_types
7   JOIN pizzas
8     ON pizza_types.pizza_type_id = pizzas.pizza_type_id
9   JOIN order_details
10    ON order_details.pizza_id = pizzas.pizza_id
11 GROUP BY pizza_types.category
12 ORDER BY quantity DESC;
13
14
```

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

category	quantity
Classic	496
Supreme	416
Veggie	402
Chicken	359

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

[Home](#)[About](#)[Contact](#)

SQL File 6* SQL File 7* SQL File 8* SQL File 9* SQL File 10* SQL File 11*

Limit to 1000 rows

```
1 -- Group the orders by date and calculate the average
2 -- number of pizzas ordered per day.
3
4 • SELECT ROUND(AVG(quantity), 0) as avg_pizza_day
5   FROM (
6     SELECT
7       orders.order_date,
8       SUM(order_details.quantity) AS quantity
9     FROM orders
10    JOIN order_details
11      ON orders.order_id = order_details.order_id
12    GROUP BY orders.order_date
13  ) AS order_quantity;
14
```

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

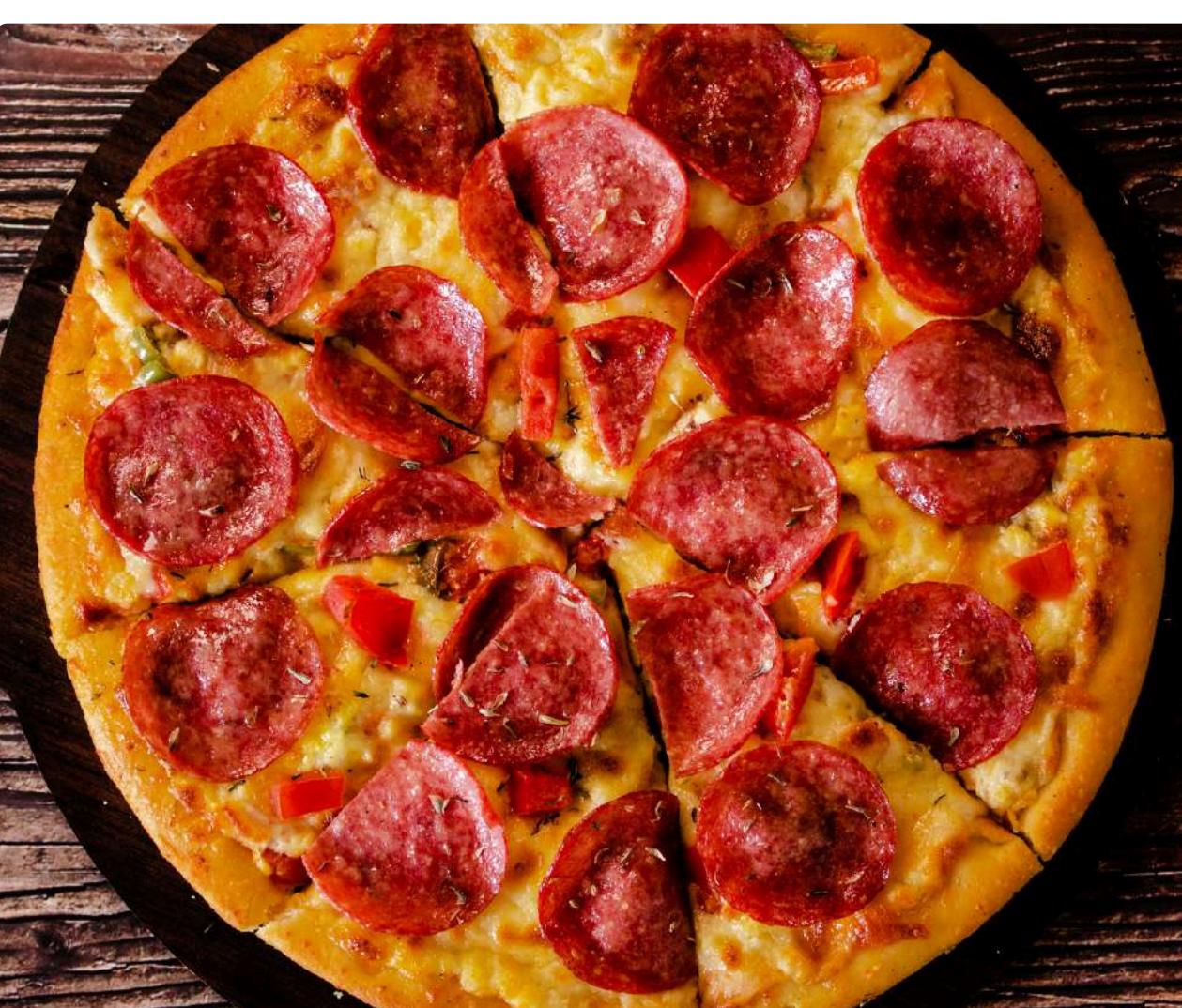
avg_pizza_day
129



SHODWE

Pizza Resto

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



```
SQL File 6* SQL File 7* SQL File 8* SQL File 9* SQL File 10* SQL File 11* SQL File 12*
[File] [New] [Open] [Save] [Print] [Help] | [Redo] [Undo] [Close] | [SQL] | Limit to 1000 rows ▾ | [Star] [Comment] [Search] [Find] [Replace] [Copy] [Paste] [Close]

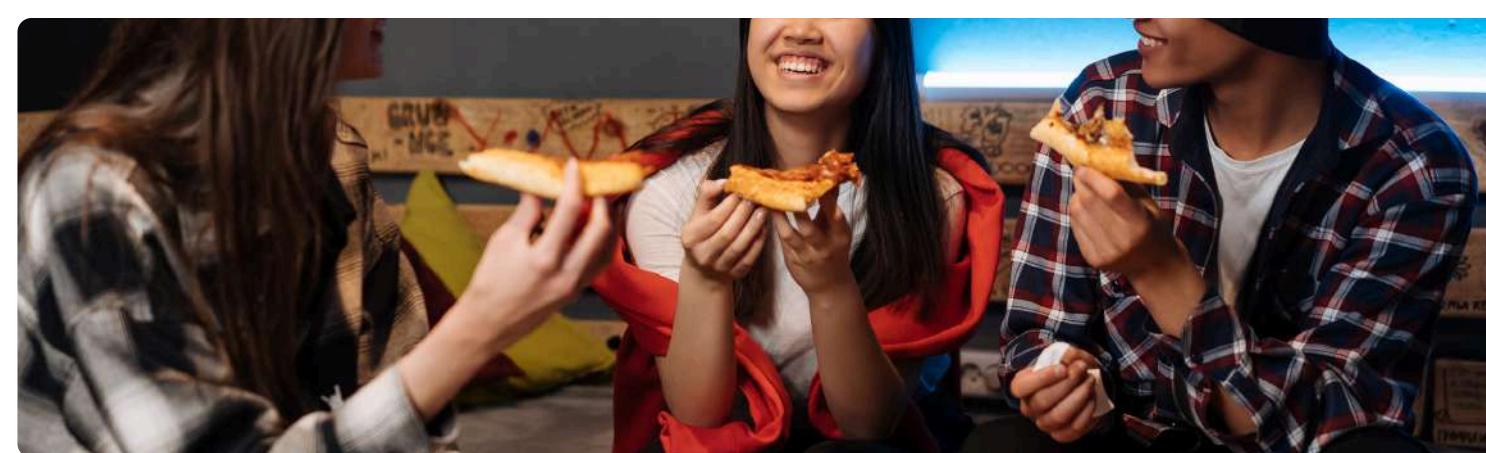
2
3 • SELECT
4     pizza_types.name,
5     SUM(order_details.quantity * pizzas.price) AS revenue
6 FROM pizza_types
7 JOIN pizzas
8     ON pizzas.pizza_type_id = pizza_types.pizza_type_id
9 JOIN order_details
10    ON order_details.pizza_id = pizzas.pizza_id
11 GROUP BY pizza_types.name
12 ORDER BY revenue DESC
13 LIMIT 3;
14

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


|   | name                       | revenue |
|---|----------------------------|---------|
| ▶ | The Thai Chicken Pizza     | 1446.5  |
|   | The Barbecue Chicken Pizza | 1416    |
|   | The Italian Supreme Pizza  | 1365.75 |


```

CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE OF TOTAL REVENUE

[Home](#)[About](#)[Contact](#)

```
SQL File 6* SQL File 7* SQL File 8* SQL File 9* SQL File 10* SQL File 11*
[File] [New] [Open] [Save] [Import] [Export] [Print] [Copy] [Paste] [Find] [Replace] [Help] [Limit to 1000 rows] [Star] [Search]
3 •  SELECT
4      pt.category,
5      ROUND(
6          SUM(od.quantity * p.price) /
7              (SELECT SUM(od2.quantity * p2.price)
8                  FROM order_details od2
9                      JOIN pizzas p2 ON p2.pizza_id = od2.pizza_id
10                 ) * 100,
11      2) AS revenue_percentage
12  FROM pizza_types pt
13  JOIN pizzas p
14      ON pt.pizza_type_id = p.pizza_type_id
15  JOIN order_details od
16      ON od.pizza_id = p.pizza_id
17  GROUP BY pt.category
18  ORDER BY revenue_percentage DESC;
19
```

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

	category	revenue_percentage
▶	Classic	26.28
	Supreme	26.2
	Veggie	24.47
	Chicken	23.04

ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

[Home](#)[About](#)[Contact](#)

SQL File 6* SQL File 7* SQL File 8* SQL File 9* SQL File 10* SQL File 11*

Limit to 1000 rows

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

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

order_date	cum_revenue
2015-01-01	2787.6000000000004
2015-01-02	5519.5
2015-01-03	8181.9





DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

The screenshot shows a MySQL Workbench interface with the following details:

- SQL File 11*** is the active tab in the tab bar.
- The toolbar includes icons for file operations, search, and connection management.
- The SQL editor contains the following code:

```
1 -- Determine the distribution of orders by hour of the day.
2
3 SELECT
4     HOUR(order_time), COUNT(order_id)
5 FROM
6     orders
7 GROUP BY HOUR(order_time)
8
```
- The results pane at the bottom displays the output of the query in a "Result Grid".
- The grid has two columns: "HOUR(order_time)" and "COUNT(order_id)".
- The data rows are:

HOUR(order_time)	COUNT(order_id)
11	1231
12	2520
13	2455



IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

[Home](#)

[About](#)

[Contact](#)

```
4 •   SELECT pizzas.size,  
5           COUNT(order_details.order_details_id)  
6     FROM pizzas  
7   JOIN order_details  
8     ON pizzas.pizza_id = order_details.pizza_id  
9   GROUP BY pizzas.size;  
10
```

The screenshot shows a MySQL Workbench interface. At the top, there's a code editor window containing the SQL query. Below it is a toolbar with icons for 'Result Grid' (selected), 'Filter Rows', 'Export', and 'Wrap Cell Content'. The main area displays a result grid with two rows of data. The first row has columns for 'size' and 'COUNT(order_details.order_details_id)'. The second row has columns for 'M' and '527'.

	size	COUNT(order_details.order_details_id)
▶	M	527

IDENTIFY THE HIGHEST-PRICED PIZZA.



SQL File 6* SQL File 7* **SQL File 8*** SQL File 9* SQL File 10* SQL File 11* SQL File 12*

Limit to 1000 rows

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

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

	name	price
▶	The Greek Pizza	35.95

Conclusion

✓ Key Insights

- Identified **top-selling and highest revenue**-generating pizzas using sales and quantity analysis.
- Analyzed **order trends by time and date** to understand customer ordering behavior.
- Evaluated **category-wise performance** and revenue contribution to total sales.
- Tracked **cumulative revenue growth** to understand overall business performance.

✓ Business Impact

- Helps optimize pricing strategy and **inventory planning**.
- Supports **data-driven decision-making** for sales and marketing.
- Provides clear visibility into **customer preferences** and peak demand periods.

✓ Skills Gained

- Advanced SQL querying using **JOINs, Aggregations, Subqueries, and Window Functions**.

This project demonstrates the practical use of SQL in real-world data analysis and business decision-making.





SHODWE
Pizza Resto

Home

About

THANK YOU FOR ATTENTION

• PIZZA RESTO PRESENTATION

