

# Pizza Sales Analysis

---

Using SQL

# Business Problem

---

- Analyze pizza sales data to find customer preferences, revenue patterns, and business insights.
- Why SQL? → Efficient querying & analysis on relational data



# Dataset Used

---

## File Name

orders.csv

order\_details.csv

pizzas.csv

pizza\_types.csv

[View Dataset](#)

## Description

Contains all order information: order\_id, Order\_date, Order\_Time, etc.

Contains item-level details for each order: Order\_details, order\_id, pizza\_id, quantity


Contains pizza details: pizza\_id, pizza\_type\_id, size, prize

Contains pizza category information: pizza\_type\_id, name, category, ingredient

# Retrieve the total number of orders placed.

---

```
3 • SELECT
4     COUNT(order_id) AS total_orders
5 FROM
6     orders;
7 |
```

Result Grid 	
	total_orders
▶	21350

Calculate the total revenue generated from pizza sales.

```
3 • SELECT
4   ROUND(SUM(order_details.quantity * pizzas.price),
5         2) AS Total_Sales
6 FROM
7   order_details
8   JOIN
9   pizzas ON pizzas.pizza_id = order_details.pizza_id;
```



Result Grid | 

	Total_Sales
▶	817860.05



# Identify the highest-priced pizza.

```
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;
```

Result Grid				 Filter Rows
	name	price		
▶	The Greek Pizza	35.95		

# Identify the most common pizza size ordered.

```
3 • SELECT
4     pizzas.size,
5     COUNT(order_details.order_details) AS order_count
6 FROM
7     pizzas
8     JOIN
9     order_details ON pizzas.pizza_id = order_details.pizza_id
10 GROUP BY pizzas.size
11 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.

```
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 pizzas.pizza_id = order_details.pizza_id
11 GROUP BY pizza_types.name
12 ORDER BY quantity DESC
13 LIMIT 5;
```

Result Grid			Filter Rows:
	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.

```
3 • SELECT
4     pizza_types.category,
5     SUM(order_details.quantity) AS Total_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 pizzas.pizza_id = order_details.pizza_id
12 GROUP BY pizza_types.category
13 ORDER BY Total_Quantity DESC;
```

Result Grid			Filter Rows
	category	Total_Quantity	
▶	Classic	14888	
	Supreme	11987	
	Veggie	11649	
	Chicken	11050	

# Determine the distribution of orders by hour of the day.

```
3 • SELECT
4     HOUR(order_time), COUNT(order_id)
5 FROM
6     orders
7 GROUP BY HOUR(order_time);
```

Result Grid			Filter Rows:
	HOUR(order_time)	COUNT(order_id)	
▶	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.

```
3 • SELECT
4     category, COUNT(name)
5 FROM
6     pizza_types
7 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.

```
3 • SELECT
4     ROUND(AVG(quantity), 0) AS Avg_Pizza_Order_PerDay
5 FROM
6     (SELECT
7         orders.order_date, SUM(order_details.quantity) AS quantity
8     FROM
9         orders
10    JOIN order_details ON orders.order_id = order_details.order_id
11   GROUP BY orders.order_date) AS Order_Quantity
```

Result Grid		Filter Rows:
	Avg_Pizza_Order_PerDay	
▶	138	

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

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

```
3 • SELECT
4     pizza_types.category,
5     ROUND(SUM(order_details.quantity * pizzas.price) / (SELECT
6         SUM(order_details.quantity * pizzas.price)
7         FROM
8             order_details
9             JOIN
10                pizzas ON order_details.pizza_id = pizzas.pizza_id) * 100, 2) AS revenue
11 FROM
12     pizza_types
13     JOIN
14     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
15     JOIN
16     order_details ON order_details.pizza_id = pizzas.pizza_id
17 GROUP BY pizza_types.category
18 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.

```
3 • select order_date ,  
4     sum(revenue) over (order by order_date)  
5     from  
6     (select orders.order_date ,  
7       sum(order_details.quantity * pizzas.price) as revenue  
8       from order_details join pizzas  
9       on order_details.pizza_id = pizzas.pizza_id  
10      join orders  
11      on orders.order_id = order_details.order_id  
12      group by orders.order_date) as sales;
```

Result Grid			Filter Rows:	Export:
	order_date	sum(revenue) over (order by order_date)		
▶	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		

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

```
3 • select name , revenue
4 from
5 (select category , name , revenue ,
6 rank() over (partition by category order by revenue) as rn
7 from
8 (select pizza_types.category , pizza_types.name,
9 sum(order_details.quantity * pizzas.price) as revenue
10 from order_details join pizzas
11 on order_details.pizza_id = pizzas.pizza_id
12 join pizza_types
13 on pizza_types.pizza_type_id = pizzas.pizza_type_id
14 group by pizza_types.category , pizza_types.name) as a) as b
15 where rn <= 3;
```

Result Grid			Filter Rows:	Export:
	name	revenue		
▶	The Chicken Pesto Pizza	16701.75		
	The Chicken Alfredo Pizza	16900.25		
	The Southwest Chicken Pizza	34705.75		
	The Pepperoni, Mushroom, and Peppers Pizza	18834.5		
	The Big Meat Pizza	22968		
	The Napolitana Pizza	24087		
	The Brie Carre Pizza	11588.499999999999		
	The Spinach Supreme Pizza	15277.75		
	The Calabrese Pizza	15934.25		
	The Green Garden Pizza	13955.75		
	The Mediterranean Pizza	15360.5		
	The Spinach Pesto Pizza	15596		



# Key Insights

---

- **Classic pizzas** contributed the highest share of total orders.
- **Large size pizzas** were the most commonly ordered size.
- **Thai Chicken Large** was among the highest revenue-generating pizzas.
- **Peak ordering hours:** 12 PM–2 PM and 6 PM–8 PM.
- **Monthly sales trend** showed December as the best-performing month.



# Conclusion

---

- SQL helped uncover **key insights** like top-selling pizzas, revenue drivers, and customer preferences.
- Showed how **data can guide business decisions**.
- This project improved my **SQL + data storytelling skills**.

# Let's Connect

---

