

# PIZZA SALES PERFORMANCE ANALYSIS

## Strategic Data Insights using Advanced MySQL

### ⌚ Executive Summary: Unlocking Business Value

This document details a comprehensive data analysis project focused on transforming raw transactional data from a fictional pizza chain into actionable business intelligence. The core objective was to move beyond simple reporting to uncover specific, high-value insights that can drive revenue growth and operational efficiency through the masterful application of SQL.

### 💡 Project Goal

To design, query, and analyse a large-scale sales dataset using **Advanced SQL (MySQL)** to quantify business performance, model growth trends, and generate strategic recommendations for product and operations teams.

### ❖ Key Results & Impact Snapshot

- Revenue Growth Modelling:** Successfully analysed the **Cumulative Revenue trend** over time, providing crucial visualization for business forecasting.
- Strategic Menu Focus:** Pinpointed the **Top 3 Revenue Drivers** within *every* pizza category, essential for inventory management and menu profitability optimization.
- Operational Efficiency:** Mapped the **Hourly Order Distribution** to identify peak periods, directly informing precise staff scheduling and resource allocation.
- Technical Mastery:** Verified expertise in **Advanced SQL techniques** including **Window Functions, CTEs, and Complex Joins** to solve challenging, real-world business questions.

### ❖ Technical Environment

Category	Tool / Method Used
Database Platform	MySQL Workbench / MySQL Server
Core Techniques	Advanced SQL, Data Modelling, Time-Series Analysis

So, the dataset we used for this project has 4 tables in it followed as below.

The image displays four separate windows from a database management tool, each showing the schema and data for one of the four tables used in the project.

- Top Left Window (order\_details):**
  - Schema:** pizzahut
  - Tables:** order\_details, orders, pizza\_types, pizzas
  - Result Grid:**

order_details_id	order_id	pizza_id	quantity
1	1	hawaiian_m	1
2	2	classic_dlx_m	1
3	2	five_cheese_l	1
4	2	ital_supr_l	1
5	2	mexicana_m	1
6	2	tha_dlx_l	1
- Top Right Window (pizzas):**
  - Schema:** pizzahut
  - Tables:** order\_details, orders, pizza\_types, pizzas
  - Result Grid:**

pizza_id	pizza_type_id	size	price
bbq_dkn_s	bbq_dkn	S	12.75
bbq_dkn_m	bbq_dkn	M	16.75
bbq_dkn_l	bbq_dkn	L	20.75
- Middle Left Window (pizza\_types):**
  - Schema:** pizzahut
  - Tables:** order\_details, orders, pizza\_types, pizzas
  - Result Grid:**

pizza_type_id	name	category	ingredients
bbq_dkn	The Barbecue Chicken Pizza	Chicken	Barbecued Chicken, Red Peppers, Green Pepe...
cali_dkn	The California Chicken Pizza	Chicken	Chicken, Artichoke, Spinach, Garlic, Jalapeno P...
dkn_alfredo	The Chicken Alfredo Pizza	Chicken	Chicken, Red Onions, Red Peppers, Mushrooms...
- Middle Right Window (orders):**
  - Schema:** pizzahut
  - Tables:** order\_details, orders, pizza\_types, pizzas
  - Result Grid:**

order_id	order_date	order_time
1	2015-01-01	11:38:36
2	2015-01-01	11:57:40
3	2015-01-01	12:12:28
4	2015-01-01	12:16:31

## Tasks Done in this project:

### Basic:

1. Retrieve the total number of orders placed.

Code:

```
1  #Retrieve the total no.of orders placed
2 • select count(order_id) as total_orders from orders;
3
```

Result Grid	
total_orders	
	21350

2. Calculate the total revenue generated from pizza sales.

Code:

```
1  #calculate total revenue generated from pizza sales.
2 • select round(sum(order_details.quantity*pizzas.price),2) as toatl_sales
3  From order_details join pizzas
4  On pizzas.pizza_id=order_details.pizza_id
```

Result Grid	
toatl_sales	
	817860.05

3. Identify the highest-priced pizza.

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;  
6
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
	name	price			
▶	The Greek Pizza	35.95			

4. Identify the most common pizza size ordered.

Code:

```
1 # Identify the most common pizza size ordered.  
2 • SELECT  
3     pizzas.size,  
4     COUNT(order_details.order_details_id) AS order_count  
5 FROM pizzas JOIN  
6     order_details ON pizzas.pizza_id = order_details.pizza_id  
7 GROUP BY pizzas.size  
8 ORDER BY order_count DESC;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
	size	order_count			
▶	L	18526			
	M	15385			
	S	14137			
	XL	544			
	XXL	28			

5. List the top 5 most ordered pizza types along with their quantities.

Code:

```
1 •  SELECT
2      pizza_types.name,
3          SUM(order_details.quantity) AS quantity -- Corrected spelling
4  FROM pizza_types JOIN
5      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id JOIN
6      order_details ON order_details.pizza_id = pizzas.pizza_id
7  GROUP BY pizza_types.name
8  ORDER BY quantity DESC -- Now correctly referencing the alias 'quantity'
9  LIMIT 5;
```

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

### Intermediate Level:

1. Join the necessary tables to find the total quantity of each pizza category ordered.

Code:

```
1 •  SELECT
2      pizza_types.category,
3          SUM(order_details.quantity) AS quantity
4  FROM pizza_types
5  JOIN pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
6  JOIN order_details ON order_details.pizza_id = pizzas.pizza_id
7  GROUP BY pizza_types.category
8  ORDER BY quantity DESC;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
category	quantity				
Classic	14888				
Supreme	11987				
Veggie	11649				
Chicken	11050				

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

Code:

```
1 • SELECT hour(order_time) AS hour,  
2   count(order_id) AS order_count  
3   FROM orders  
4   Group by hour(order_time);
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
hour	order_count			
11	1231			
12	2520			
13	2455			
14	1472			
15	1468			
16	1920			
...	---			

3. Join relevant tables to find the category-wise distribution of pizzas.

Code:

```
1 • Select category,  
2   count(name)  
3   from pizza_types  
4   group by category;
```

Result Grid		Filter Rows:	Exports:
category	count(name)		
Chicken	6		
Classic	8		
Supreme	9		
Veggie	9		

4. Group the orders by date and calculate the average number of pizzas ordered per day.

Code:

```
1 • Select round(avg(quantity),0)  
2   FROM (select orders.order_date,  
3     SUM(order_details.quantity)AS quantity  
4     FROM orders  
5     JOIN order_details  
6     ON orders.order_id=order_details.order_id  
7     GROUP BY orders.order_date)  
8   AS order_quantity;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	round(avg(quantity),0)			
▶	138			

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

Code:

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

The screenshot shows a MySQL query results grid. The columns are labeled 'name' and 'revenue'. The data rows are: 'The Thai Chicken Pizza' with a revenue of 43434.25, 'The Barbecue Chicken Pizza' with a revenue of 42768, and 'The California Chicken Pizza' with a revenue of 41409.5. The grid has standard MySQL interface elements like 'Result Grid', 'Filter Rows', 'Export', 'Wrap Cell Content', and 'Fetch rows'.

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

## Advanced Level:

1. Calculate the percentage contribution of each pizza type to total revenue.

Code:

```
1 •  SELECT
2      pizza_types.category,
3      ROUND(SUM(order_details.quantity * pizzas.price) / (SELECT
4          SUM(order_details.quantity * pizzas.price)
5      FROM order_details JOIN pizzas ON pizzas.pizza_id = order_details.pizza_id) * 100,2) AS revenue_percentage
6  FROM pizza_types JOIN pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
7  JOIN order_details ON order_details.pizza_id = pizzas.pizza_id
8  GROUP BY pizza_types.category
9  ORDER BY revenue_percentage DESC;
```

The screenshot shows a MySQL query results grid. The columns are labeled 'category' and 'revenue\_percentage'. The data rows are: 'Classic' with 26.91%, 'Supreme' with 25.46%, 'Chicken' with 23.96%, and 'Veggie' with 23.68%. The grid has standard MySQL interface elements like 'Result Grid', 'Filter Rows', 'Export', 'Wrap Cell Content', and 'Fetch rows'.

	category	revenue_percentage
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

2. Analyze the cumulative revenue generated over time.

Code:

```
1 •  Select order_date,SUM(revenue)
2      over(order by order_date) as cum_revenue
3  FROM (Select orders.order_date,
4            SUM(order_details.quantity*pizzas.price)as revenue
5      From order_details join pizzas on
6          order_details.pizza_id=pizzas.pizza_id
7      JOIN orders On
8          orders.order_id=order_details.order_id
9      GROUP BY orders.order_date) AS sales;
10
```

Result Grid		
	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	11629.55

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

Code:

```
1 •  Select name,revenue from
2      (Select category,name,revenue,rank() over(partition by category
3          order by revenue desc)as rn
4      FROM (Select pizza_types.category,pizza_types.name,
5            SUM((order_details.quantity)*pizzas.price)as revenue
6      From pizza_types join pizzas on
7          pizza_types.pizza_type_id=pizzas.pizza_type_id
8      JOIN order_details on order_details.pizza_id=pizzas.pizza_id
9      GROUP BY pizza_types.category,pizza_types.name)as a)as b
10     WHERE rn<=3;
```

Result Grid		
	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	37500.0