



# PIZZA SALES ANALYSIS REPORT



● BY PRIYANKA KUTE



# ABOUT REPORT



## Pizza Sales Analysis - SQL Project



### ◆ Objectives

- Find total orders & revenue
- Identify most popular pizza size & types
- Analyze sales distribution by category & time
- Calculate cumulative revenue trends
- Find top 3 pizzas by revenue in each category

### ◆ Tech Stack

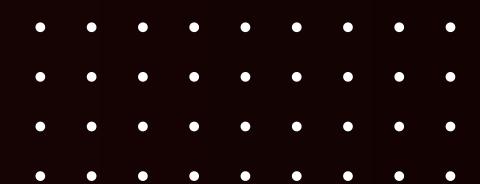
- SQL (Joins, Aggregations, Window Functions)
- Database: MySQL

### ◆ Key Insights

- Medium-sized pizzas were the most ordered
- Evening hours showed peak demand
- Top 3 pizzas contributed to ~40% of revenue

### ◆ Files

- `pizza\_sales\_analysis.sql` → All queries
- `schema.png` → Database schema
- `pizza\_sales\_insights.pdf` → Key results & charts



# RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.



SQL File 1\* Pizza sales analysis Pizza sales analysis SQL File 5\* ×

File Edit View Insert Tools Window Help

Folder Open Save As Import Export Refresh Undo Redo Find Replace Go To Home

1    SELECT COUNT(order\_id) AS total\_orders  
2    FROM orders;

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

	total_orders
▶	21350

# CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.



```
1 •   SELECT SUM(od.quantity * p.price) AS total_revenue
2     FROM order_details od
3       JOIN pizzas p ON od.pizza_id = p.pizza_id;
4
```

The screenshot shows a MySQL query interface. The query calculates the total revenue from pizza sales by joining the `order\_details` table with the `pizzas` table based on the `pizza\_id`. The result grid displays one row with the column name `total\_revenue` and its value `817860.049999993`.

total_revenue
817860.049999993

# IDENTIFY THE HIGHEST-PRICED PIZZA.



A screenshot of a MySQL Workbench interface showing a SQL query and its results. The query identifies the pizza size with the highest total orders. The result grid shows one row with size 'L' and total orders 18526.

```
SQL File 1* Pizza sales analysis Pizza sales analysis SQL File 5* ×
File Edit View Insert Object Data Tools Help
1 • SELECT p.size, COUNT(*) AS total_orders
2   FROM order_details od
3   JOIN pizzas p ON od.pizza_id = p.pizza_id
4   GROUP BY p.size
5   ORDER BY total_orders DESC
6   LIMIT 1;
```

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

	size	total_orders
▶	L	18526

# IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.



The screenshot shows a SQL query editor interface with a toolbar at the top and a result grid below. The toolbar includes icons for file operations, search, and execution. The tabs show "SQL File 1\*", "Pizza sales analysis", "Pizza sales analysis", and "SQL File 5\*". The active tab contains the following SQL code:

```
1 •  SELECT p.pizza_id, pt.name, p.size, p.price
2    FROM pizzas p
3    JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
4   ORDER BY p.price DESC
5   LIMIT 1;
```

The result grid displays the following data:

	pizza_id	name	size	price
▶	the_greek_xxL	The Greek Pizza	XXL	35.95

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

SQL File 1\* Pizza sales analysis Pizza sales analysis SQL File 5\* ×

1 • SELECT pt.name, SUM(od.quantity) AS total\_quantity  
2 FROM order\_details od  
3 JOIN pizzas p ON od.pizza\_id = p.pizza\_id  
4 JOIN pizza\_types pt ON p.pizza\_type\_id = pt.pizza\_type\_id  
5 GROUP BY pt.name  
6 ORDER BY total\_quantity DESC  
7 LIMIT 5;

R

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

	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.



The screenshot shows a SQL development environment with a dark theme. At the top, there are four tabs: "SQL File 1\*", "Pizza sales analysis", "Pizza sales analysis", and "SQL File 5\*". Below the tabs is a toolbar with various icons for file operations, search, and database management. The main area contains a SQL query:

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

At the bottom, there is a "Result Grid" section showing the results of the query:

	category	total_quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

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



A screenshot of a MySQL Workbench interface showing a SQL query and its results. The SQL query is:

```
1 •  SELECT HOUR(order_time) AS order_hour, COUNT(order_id) AS total_orders
2   FROM orders
3   GROUP BY HOUR(order_time)
4   ORDER BY order_hour;
```

The results are displayed in a table titled "Result Grid":

order_hour	total_orders
9	1
10	8
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

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



A screenshot of a MySQL Workbench interface showing an SQL query and its results. The query joins the pizzas table with the pizza\_types table to find the category-wise distribution of pizzas.

```
SQL File 1* Pizza sales analysis Pizza sales analysis SQL File 5* ×  
File Edit View Insert Object Tools Help  
1 • 1 SELECT pt.category, COUNT(DISTINCT p.pizza_id) AS total_pizzas  
2 FROM pizzas p  
3 JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id  
4 GROUP BY pt.category;  
5
```

The Result Grid shows the following data:

category	total_pizzas
Chicken	18
Classic	26
Supreme	25
Veggie	27

# RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.



SQL File 1\* Pizza sales analysis Pizza sales analysis SQL File 5\* ×

1 • SELECT order\_date, AVG(pizza\_count) AS avg\_pizzas\_per\_day  
2 FROM (  
3 SELECT o.order\_date, SUM(od.quantity) AS pizza\_count  
4 FROM orders o  
5 JOIN order\_details od ON o.order\_id = od.order\_id  
6 GROUP BY o.order\_date  
7 ) daily\_orders  
8 GROUP BY order\_date;

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

order_date	avg_pizzas_per_day
2015-01-01	162.0000
2015-01-02	165.0000
2015-01-03	158.0000
2015-01-04	106.0000
2015-01-05	125.0000
2015-01-06	147.0000
2015-01-07	138.0000
2015-01-08	173.0000
2015-01-09	127.0000
2015-01-10	146.0000
2015-01-11	116.0000
2015-01-12	119.0000
2015-01-13	120.0000
2015-01-14	150.0000
2015-01-15	123.0000

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



```
2   FROM order_details od
3   JOIN pizzas p ON od.pizza_id = p.pizza_id
4   JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
5   GROUP BY pt.name
6   ORDER BY revenue DESC
7   LIMIT 3;
8
```

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

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

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



```
1 •  SELECT order_date,
2           SUM(daily_revenue) OVER (ORDER BY order_date) AS cumulative_revenue
3   FROM (
4       SELECT o.order_date, SUM(od.quantity * p.price) AS daily_revenue
5         FROM orders o
6        JOIN order_details od ON o.order_id = od.order_id
7        JOIN pizzas p ON od.pizza_id = p.pizza_id
8        GROUP BY o.order_date
9   ) revenue_by_date;
10
```

order_date	cumulative_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	23990.350000000002

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



SQL File 1\* Pizza sales analysis Pizza sales analysis SQL File 5\* ×

File | New | Open | Save | Print | Find | Replace | Options | Help | Don't Limit | | | | | | | | | | |

```
1 •   SELECT pt.name,
2     ROUND(SUM(od.quantity * p.price) * 100.0 /
3         (SELECT SUM(od2.quantity * p2.price)
4             FROM order_details od2
5                 JOIN pizzas p2 ON od2.pizza_id = p2.pizza_id), 2) AS revenue_percentage
6     FROM order_details od
7     JOIN pizzas p ON od.pizza_id = p.pizza_id
8     JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
9     GROUP BY pt.name
10    ORDER BY revenue_percentage DESC;
11
```

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

name	revenue_percentage
The Thai Chicken Pizza	5.31
The Barbecue Chicken Pizza	5.23
The California Chicken Pizza	5.06
The Classic Deluxe Pizza	4.67
The Spicy Italian Pizza	4.26
The Southwest Chicken Pizza	4.24
The Italian Supreme Pizza	4.09
The Hawaiian Pizza	3.95
The Four Cheese Pizza	3.95
The Sicilian Pizza	3.78
The Danneroni Pizza	3.60

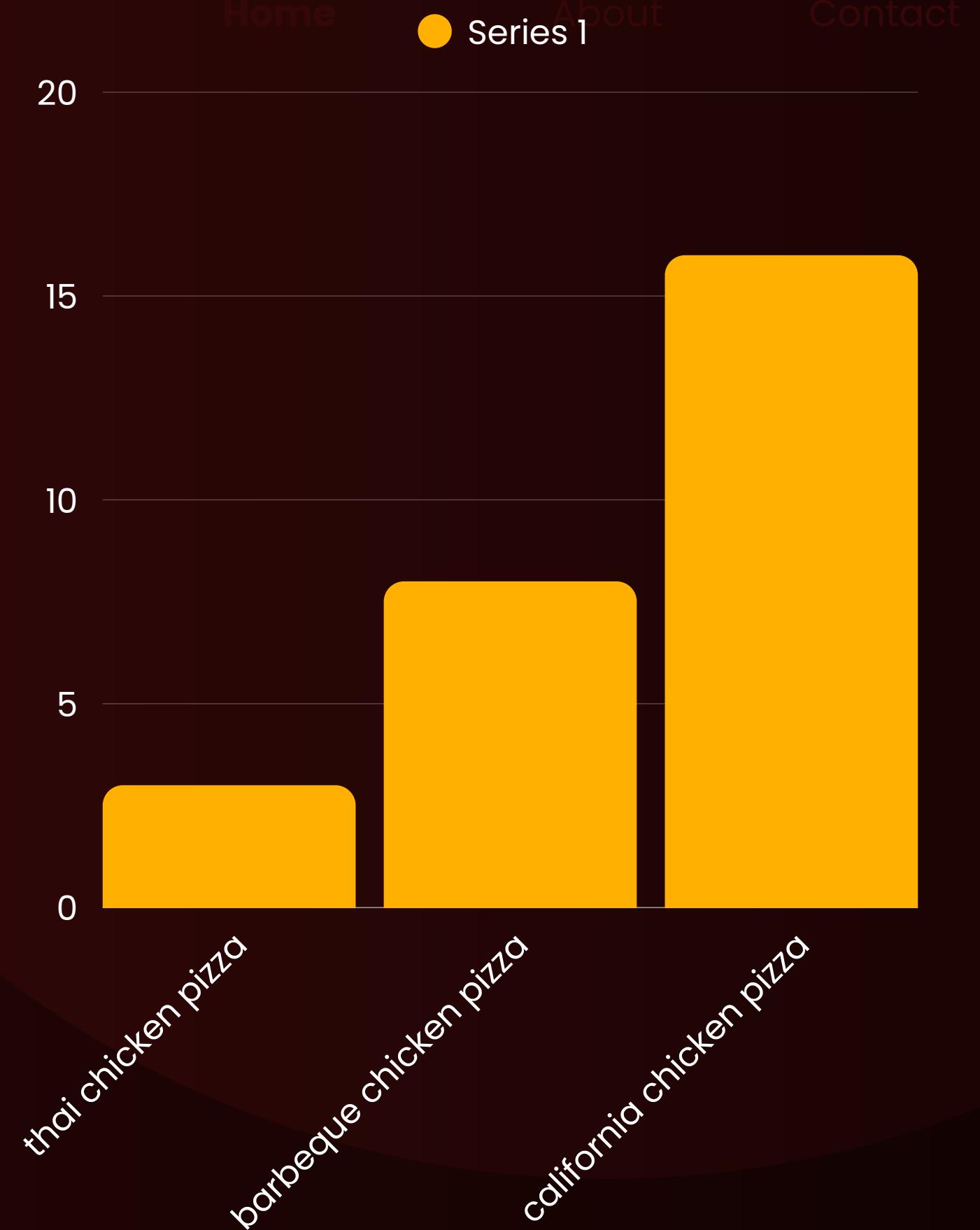
Result 11 ×



# SALES REPORT

OUR WEEKLY INCOME

sales report by category of chicken





Home

# THANK YOU