



# PIZZA SALES USING SQL

BY SAURABH SALADE

In this project i have utilizes  
SQL Quires to solve  
questions that were related  
to Pizza Sales

**Order\_Details Table**

Column Name	Data Type
order_details_id	int
order_id	int
pizza_id	string
quantity	int

**Orders table**

Column Name	Data Type
Order_id	int
order_date	date
order_time	time

**Pizza table**

Column Name	Data Type
pizza_id	string
pizza_type_id	string
size	string
price	float

**Pizza\_types table**

Column Name	Data Type
pizza_type_id	string
name	string
category	string
ingredients	string

RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

The screenshot shows a MySQL query editor window. At the top, there is a toolbar with various icons. Below the toolbar, the query text is displayed:

```
1 -- Q1 Retrieve the total number of orders placed.  
2  
3 • SELECT  
4     COUNT(order_id) AS total  
5 FROM  
6     orders;  
7
```

Below the query text, there is a toolbar with buttons for "Result Grid", "Filter Rows:", "Export:", and "Wrap Cell Content:". The "Result Grid" button is selected. The result grid itself shows one row with the column "total" containing the value "21350".

total
21350

## CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

```
1      -- Q2 Calculate the total revenue generated from pizza sales.  
2  
3 •  SELECT  
4      ROUND(SUM(quantity * price), 2) as total_revenue  
5  FROM  
6      orders_details o  
7      JOIN  
8      pizzas p ON o.pizza_id = p.pizza_id;
```

The screenshot shows a MySQL query results window. At the top, there are several buttons: 'Result Grid' (selected), 'Filter Rows:', 'Export:', and 'Wrap Cell Content:'. Below the buttons is a table with one row. The table has two columns: the first column is empty, and the second column is labeled 'total\_revenue' with the value '817860.05'.

	total_revenue
▶	817860.05

# IDENTIFY THE HIGHEST-PRICED PIZZA.

The screenshot shows a MySQL Workbench interface with two SQL queries. The first query uses a JOIN and ORDER BY clause to find the highest-priced pizza. The second query uses a window function (DENSE\_RANK) to achieve the same result. Both queries return a single row for 'The Greek Pizza' at a price of 35.95.

```
1 -- Q Identify the highest-priced pizza.
2
3 • SELECT
4     pt.name, p.price
5     FROM
6     pizza_types pt
7         JOIN
8     pizzas p ON pt.pizza_type_id = p.pizza_type_id
9     ORDER BY price DESC
10    LIMIT 1;
11
12 -- or using window function
13
14 • with temp as
15     (select dense_rank() over(order by price desc ) as rn from pizza_types pt join pizzas p on pt.pizza_type_id=p.pizza_type_id)
16     Select p.name from temp where rn=1;
```

Result Grid

	name	price
▶	The Greek Pizza	35.95

## IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

The screenshot shows a MySQL Workbench interface with two SQL queries:

```
1 -- Q4 Identify the most common pizza size ordered.
2 • SELECT
3     p.size, SUM(od.quantity) AS s
4     FROM
5         orders o
6             JOIN
7                 orders_details od ON o.order_id = od.order_id
8             JOIN
9                 pizzas p ON p.pizza_id = od.pizza_id
10            GROUP BY p.size
11            ORDER BY s DESC
12            LIMIT 1;
13
14 -- or using window
15
16 • with temp as
17     (select p.size, sum(od.quantity) as s, dense_rank () over(order by sum(quantity) desc) as rn
18      from orders o join orders_details od on o.order_id=od.order_id join pizzas p on p.pizza_id=od.pizza_id group by p.size)
19     Select size, s from temp where rn=1;
```

The Result Grid shows the output of the second query:

size	s
L	18956

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

```
-- Q List the top 5 most ordered pizza types along with their quantities.  
• SELECT  
    name, SUM(quantity) AS c  
FROM  
    pizza_types pt  
    JOIN  
    pizzas p ON pt.pizza_type_id = p.pizza_type_id  
    JOIN  
    orders_details od ON od.pizza_id = p.pizza_id  
GROUP BY name  
ORDER BY c DESC  
LIMIT 5;  
-- using Window  
• with temp as  
    (select name, sum(quantity)as s, dense_rank() over(order by sum(quantity) desc)as rn  
     from pizza_types pt join pizzas p on pt.pizza_type_id = p.pizza_type_id join orders_details od on od.pizza_id=p.pizza_id  
     group by name) Select name, s from temp where rn between 1 and 5;
```

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

name	s
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.

```
2      -- Q Join the necessary tables to find the total quantity of each pizza category ordered.
3 •  SELECT
4      category, SUM(quantity) AS s
5  FROM
6      pizza_types pt
7      JOIN
8      pizzas p ON pt.pizza_type_id = p.pizza_type_id
9      JOIN
10     orders_details od ON od.pizza_id = p.pizza_id
11    GROUP BY category
12   ORDER BY s;
13
14  -- or using window
15
16 •  select category, s  from
17  (select category, sum(quantity) as s, dense_rank() over(order by sum(quantity)) as rn from pizza_types pt join pizzas p on pt.pizza_type_id = p.pizza_type_id join orders_details
18  group by category) as temp
```

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

category	s
Chicken	11050
Veggie	11649
Supreme	11987
Classic	14888

Result 1 Read Only

Result Grid Form Editor

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

The screenshot shows a MySQL Workbench interface. At the top, there's a toolbar with various icons. Below the toolbar, a query editor window contains the following SQL code:

```
1 -- Q Determine the distribution of orders by hour of the day.
2 • SELECT
3     COUNT(order_time), HOUR(order_time)
4 FROM
5     orders
6 GROUP BY HOUR(order_time);
```

Below the query editor is a results grid titled "Result Grid". The grid has two columns: "COUNT(order\_time)" and "HOUR(order\_time)". The data is as follows:

	COUNT(order_time)	HOUR(order_time)
▶	1231	11
	2520	12
	2455	13
	1472	14
	1468	15
	1920	16
	2336	17
	2399	18
	2009	19
	1642	20
	1198	21
	663	22
	28	23
	8	10
	1	9

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

```
1 -- Q3 Join relevant tables to find the category-wise distribution of pizzas.  
2 • SELECT  
3     category, COUNT(name)  
4 FROM  
5     pizza_types  
6 GROUP BY category;
```

The screenshot shows a database query results window. At the top, there is a code editor with the SQL query shown above. Below the code editor is a toolbar with buttons for 'Result Grid' (selected), 'Filter Rows:', 'Export:', and 'Wrap Cell Content:'. The main area displays a table with four 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.

The screenshot shows a MySQL Workbench interface. The query editor window contains the following SQL code:

```
1 -- Q4 Group the orders by date and calculate the average number of pizzas ordered per day.
2 • SELECT
3     ROUND(AVG(qa), 0)
4 FROM
5     (SELECT
6         o.order_date, SUM(od.quantity) AS qa
7     FROM
8         orders o
9     JOIN orders_details od
10    WHERE
11        o.order_id = od.order_id
12    GROUP BY o.order_date) AS order_quant;
```

The results grid below the editor shows a single row of data:

ROUND(AVG(qa), 0)
138

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

```
1  -- Q Determine the top 3 most ordered pizza types based on revenue.
2 • SELECT
3      name, SUM(quantity * price) AS revenue
4  FROM
5      pizza_types pt
6          JOIN
7      pizzas p ON pt.pizza_type_id = p.pizza_type_id
8          JOIN
9      orders_details od ON od.pizza_id = p.pizza_id
10     GROUP BY pt.name
11     ORDER BY revenue DESC
12     LIMIT 3;
```

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

```
1  -- Q Calculate the percentage contribution of each pizza type to total revenue.
2
3  -- STEP 1, Calculate Total Revenue fir each pizza
4
5 • With pizza_revenue as
6   (Select pt.category, SUM(o.quantity*p.price) as revenue
7     from pizza_types pt join pizzas p on pt.pizza_type_id=p.pizza_type_id
8       join orders_details o on o.pizza_id= p.pizza_id group by pt.category order by revenue desc),
9
10 -- STEP 2 calculate the overall revenue
11   total_revenue as(
12     select SUM(revenue) as total_revenue from pizza_revenue)
13
14 -- STEP 3 compute the % contribution
15
16   Select pr.category, pr.revenue,
17     (pr.revenue/tr.total_revenue)*100 As percent_contribution
18   from pizza_revenue pr,total_revenue tr
19   order by percent_contribution;
20
```

Result Grid			
	category	revenue	percent_contribution
▶	Veggie	193690.45000000298	23.68259092738454
	Chicken	195919.5	23.955137556847248
	Supreme	208196.99999999822	25.45631126009858
	Classic	220053.1000000001	26.905960255669626

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

```
1  -- Q2 Determine the top 3 most ordered pizza types based on revenue for each pizza category.
2  Select name, revenue, category from
3  (select category, name, revenue, dense_rank() over(partition by category order by revenue desc)
4  as rn from
5
6  (Select pt.category,pt.name, sum((od.quantity)*p.price) as revenue
7  from pizza_types pt join pizzas p on pt.pizza_type_id=p.pizza_type_id
8  join orders_details od on od.pizza_id=p.pizza_id
9  group by pt.category, pt.name) as x ) as y where rn<=3;
```

Result Grid			
	name	revenue	category
▶	The Thai Chicken Pizza	43434.25	Chicken
	The Barbecue Chicken Pizza	42768	Chicken
	The California Chicken Pizza	41409.5	Chicken
	The Classic Deluxe Pizza	38180.5	Classic
	The Hawaiian Pizza	32273.25	Classic
	The Pepperoni Pizza	30161.75	Classic
	The Spicy Italian Pizza	34831.25	Supreme
	The Italian Supreme Pizza	33476.75	Supreme
	The Sicilian Pizza	30940.5	Supreme
	The Four Cheese Pizza	32265.70000000065	Veggie
	The Mexicana Pizza	26780.75	Veggie
	The Five Cheese Pizza	26066.5	Veggie