



LARANA PIZZA

# LA,PIZZERIA



# **SQL QUERIES AND RESULTS**

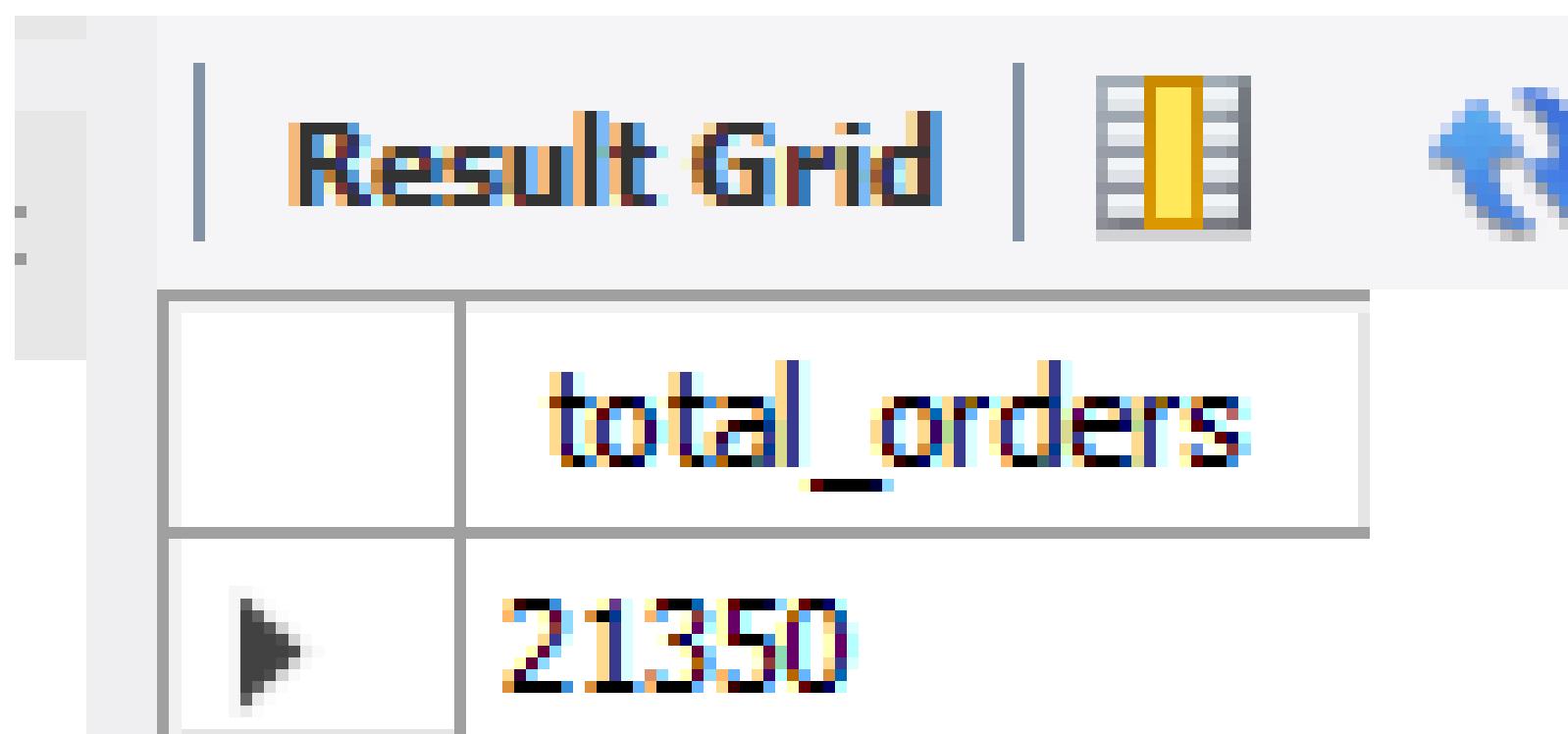
---

## QUERY 1: RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

CODE:

```
SELECT  
    COUNT(order_id) AS total_orders  
FROM  
    orders;
```

RESULT:



The screenshot shows the MySQL Workbench interface with the 'Result Grid' tab selected. The grid displays a single row of data with two columns. The first column is a navigation arrow pointing right, and the second column contains the value '21350'. The column header is 'total\_orders'.

	total_orders
→	21350

## QUERY 2: CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

CODE:

```
select
    round(sum(order_details.quantity * pizzas.price),2) as total_sales
from
    order_details join pizzas
on pizzas.pizza_id=order_details.pizza_id
```

RESULT:

	total_sales
▶	817860.05

## QUERY 3: IDENTIFY THE HIGHEST-PRICED PIZZA.

CODE:

```
SELECT pizza_types.name, pizzas.price  
FROM pizza_types  
JOIN pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id  
ORDER BY pizzas.price DESC  
LIMIT 1;
```

RESULT:

The screenshot shows a MySQL Workbench interface with a result grid. The grid has two columns: 'name' and 'price'. A single row is displayed, representing the highest-priced pizza found by the query. The row contains the name 'The Greek Pizza' and the price '35.95'.

	name	price
▶	The Greek Pizza	35.95

## QUERY 4: IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

CODE:

```
SELECT
    pizzas.size,
    COUNT(order_details.order_details_id) AS order_count
FROM
    pizzas
    JOIN
        order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC
LIMIT 1;
```

RESULT:

The screenshot shows a MySQL Workbench result grid. The top bar includes tabs for 'Result Grid' (which is selected), 'SQL Editor', and 'Filter'. The result grid has two columns: 'size' and 'order\_count'. A single row is displayed, showing 'L' in the 'size' column and '18526' in the 'order\_count' column. The 'order\_count' value is highlighted with a yellow background.

	size	order_count
→	L	18526

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

CODE:

```
SELECT
    pizza_types.name, SUM(order_details.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY quantity DESC
LIMIT 5;
```

RESULT:

	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

## QUERY 6: JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

CODE:

```
SELECT
    pizza_types.category,
    SUM(order_details.quantity) AS quantity
FROM
    pizza_types
    JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
    JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity DESC;
```

RESULT:

	category	quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

## QUERY 7: DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

CODE:

```
SELECT  
    HOUR(order_time), COUNT(order_id) as order_count  
FROM  
    orders  
GROUP BY HOUR(order_time);
```

RESULT:

Result Grid		
	HOUR(order_time)	order_count
▶	11	1231
	12	2520
	13	2455
	14	1472
	15	1468
	16	1920
	17	2336
	18	2399

## QUERY 8: JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

CODE:

```
SELECT  
    category, COUNT(name)  
FROM  
    pizza_types  
GROUP BY category;
```

RESULT:

	category	COUNT(name)
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

## QUERY 9: GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

CODE:

```
SELECT  
    ROUND(AVG(quantity), 0) as avg_pizza_ordered_per_day  
FROM  
    (SELECT  
        orders.order_data, SUM(order_details.quantity) AS quantity  
    FROM  
        orders  
    JOIN order_details ON orders.order_id = order_details.order_id  
    GROUP BY orders.order_data) AS order_quantity;
```

RESULT:

	avg_pizza_ordered_per_day
▶	138

## QUERY 10: DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

CODE:

```
SELECT
    pizza_types.name,
    SUM(order_details.quantity * pizzas.price) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```

RESULT:

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

# QUERY 11: CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

CODE:

```
SELECT
    pizza_types.category,
    ROUND(SUM(order_details.quantity * pizzas.price) / (SELECT
        ROUND(SUM(order_details.quantity * pizzas.price),
        2) AS total_sales
    FROM order_details
    JOIN
        pizzas ON pizzas.pizza_id = order_details.pizza_id) * 100,
    2) AS revenue
FROM pizza_types
JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY revenue DESC;
```

RESULT:

	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

## QUERY 12: ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

CODE:

```
SELECT
    orders.order_data,
    SUM(order_details.quantity * pizzas.price) AS revenue
FROM
    order_details
    JOIN
    pizzas ON order_details.pizza_id = pizzas.pizza_id
    JOIN
    orders ON orders.order_id = order_details.order_id
GROUP BY orders.order_data;
```

RESULT:

	order_data	revenue
	2015-01-01	2713.8500000000004
	2015-01-02	2731.8999999999996
	2015-01-03	2662.3999999999996
	2015-01-04	1755.4500000000003
	2015-01-05	2065.95

## QUERY 13: DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.

CODE:

```
SELECT
    name, revenue
FROM
    (
        SELECT category, name, revenue,
        RANK() OVER(PARTITION BY category ORDER BY revenue DESC) AS rn
        FROM
            (
                SELECT pizza_types.category, pizza_types.name,
                SUM((order_details.quantity)*pizzas.price) AS revenue
                FROM
                    pizza_types JOIN pizzas
                    ON pizza_types.pizza_type_id=pizzas.pizza_type_id
                    JOIN order_details
                    ON order_details.pizza_id=pizzas.pizza_id
                    GROUP BY pizza_types.category, pizza_types.name) AS a) AS b
    WHERE rn <=3;
```

RESULT:

	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	32273.25

*THANK you*