

# PIZZA SALES REPORT USING SQL

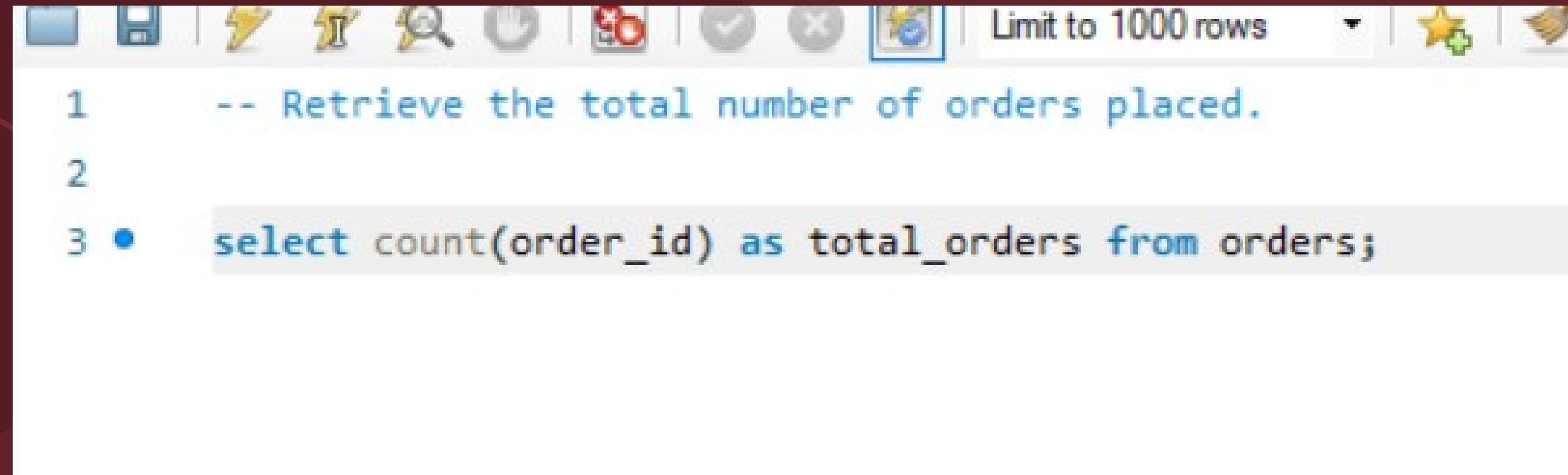
By - Swaroop Khadke



# INTRODUCTION

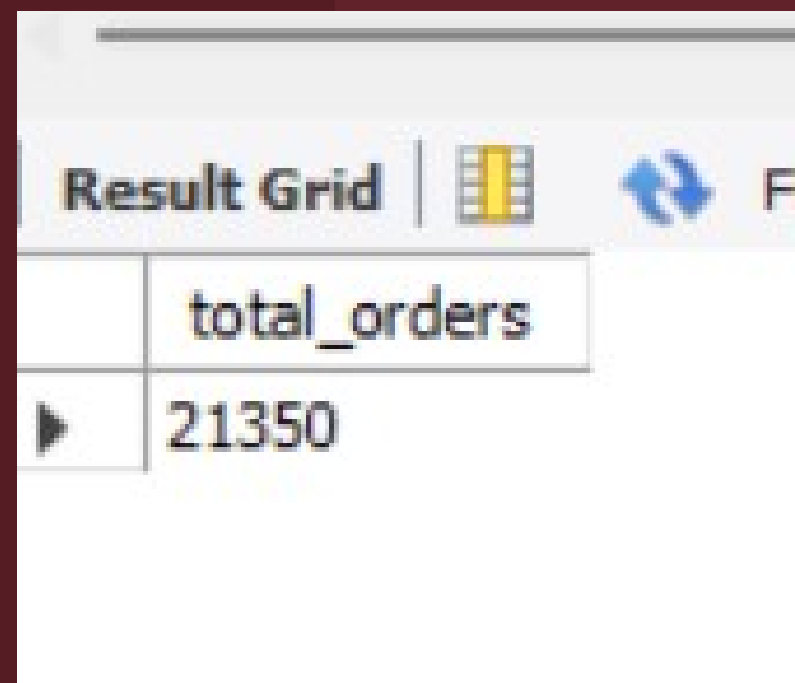
This report offers a concise analysis of pizza sales utilizing SQL. Extracting data from our sales database, SQL queries were employed to dissect sales trends, popular pizza choices, and revenue generation. Through this analysis, we aim to provide actionable insights for optimizing pizza sales strategies and enhancing overall profitability.

# RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.



A screenshot of a SQL query editor window. The window has a toolbar at the top with various icons including a folder, save, lightning bolt, magnifying glass, refresh, and a dropdown menu showing 'Limit to 1000 rows'. The query text is as follows:

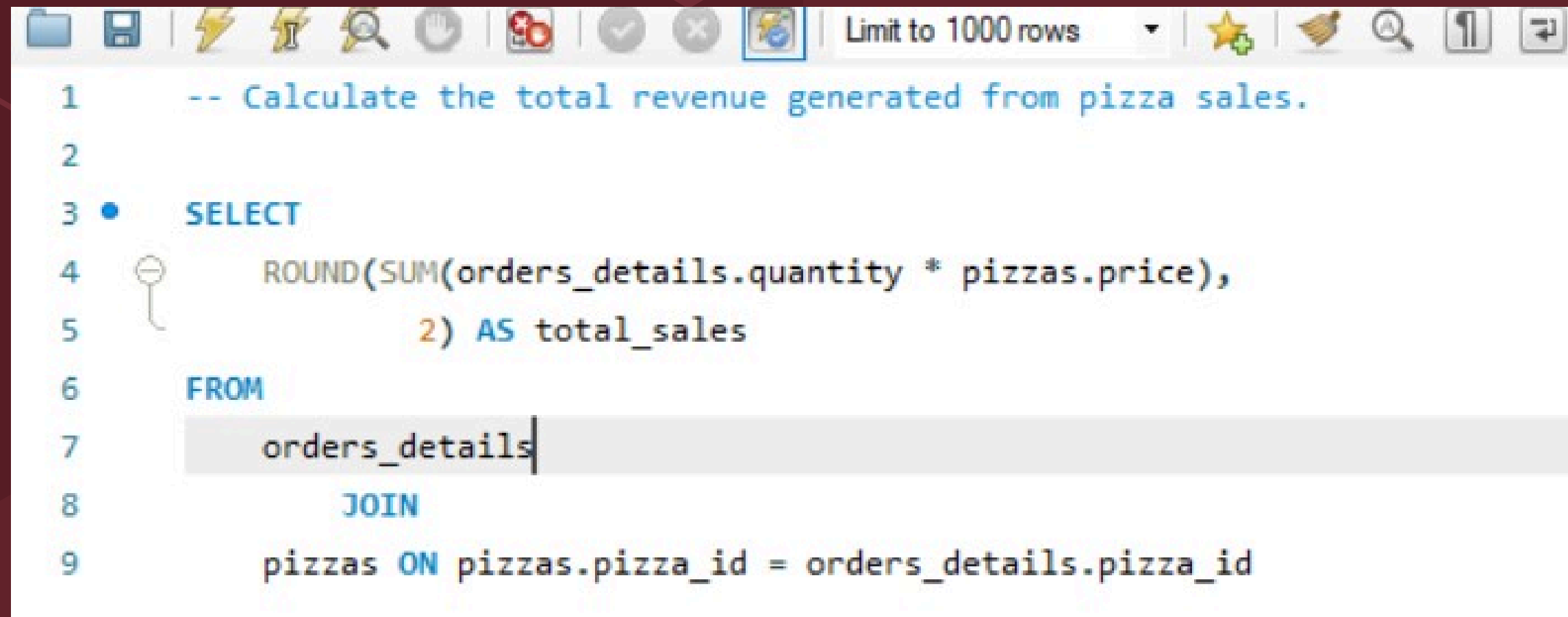
```
1  -- Retrieve the total number of orders placed.  
2  
3  • select count(order_id) as total_orders from orders;
```



A screenshot of a 'Result Grid' window. The window has a title bar and a toolbar with a 'Result Grid' tab, a grid icon, and a refresh icon. The grid contains one column named 'total\_orders' and one row with the value '21350'.

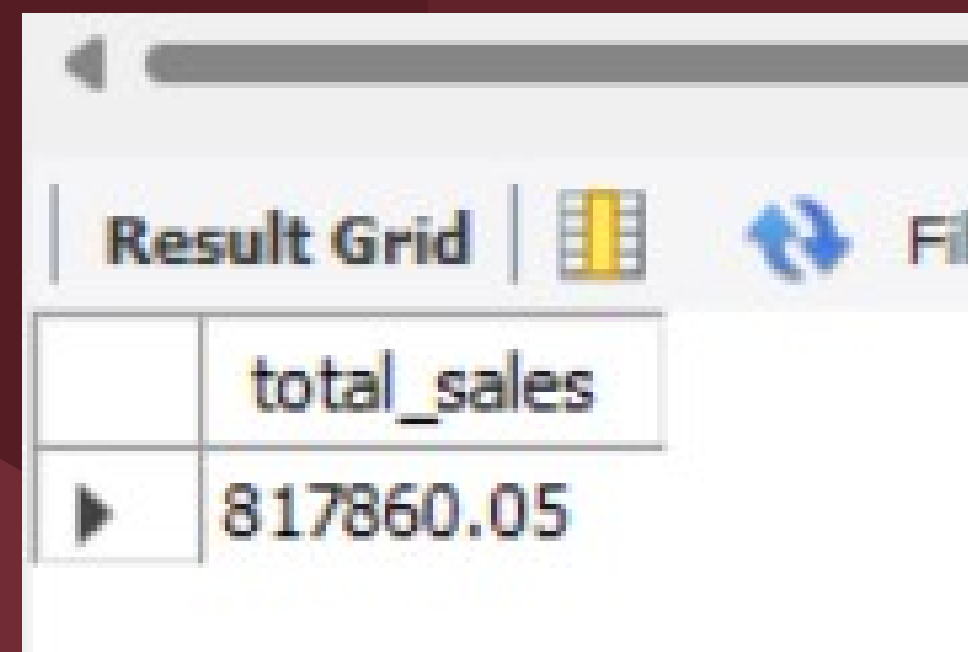
	total_orders
▶	21350

# CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

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



The screenshot shows a 'Result Grid' window with a single row of data. The column is labeled 'total\_sales' and the value is 817860.05.

	total_sales
▶	817860.05

# IDENTIFY THE HIGHEST-PRICED PIZZA.

```
1  -- Identify the highest-priced pizza.
2
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.

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

Result Grid | Filter Rows:

	size	order_count
▶	L	18526
	M	15385
	S	14137
	XL	544
	XXL	28



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

```
1  -- Join the necessary tables to find the total quantity of each pizza category ordered.
2
3  • SELECT
4      pizza_types.category,
5      SUM(orders_details.quantity) AS quantity
6  FROM
7      pizza_types
8      JOIN
9      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
10     JOIN
11     orders_details ON orders_details.pizza_id = pizzas.pizza_id
12 GROUP BY pizza_types.category
13 ORDER BY quantity DESC;
```

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

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

```
1  -- Calculate the percentage contribution of each
2  -- pizza type to total revenue.
3
4  SELECT
5      pizza_types.category,
6      round((SUM(orders_details.quantity * pizzas.price) / (SELECT
7          ROUND(SUM(orders_details.quantity * pizzas.price),
8              2) AS total_sales
9
10         FROM
11             orders_details
12             JOIN
13                 pizzas ON pizzas.pizza_id = orders_details.pizza_id)) *100,2 ) AS revenue
14
15     FROM
16         pizza_types
17         JOIN
18             pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
19         JOIN
20             orders_details ON orders_details.pizza_id = pizzas.pizza_id
21
22     GROUP BY pizza_types.category
23     ORDER BY revenue DESC;
```

Result Grid     Filter Rows:		
	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68



# ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```
1  -- Analyze the cumulative revenue generated over time.
2
3
4  • select order_date,
5     sum(revenue) over(order by order_date) as cum_revenue
6  from
7  (select orders.order_date,
8     sum(orders_details.quantity * pizzas.price)as revenue
9   from orders_details join pizzas
10    on orders_details.pizza_id = pizzas.pizza_id
11   join orders
12    on orders.order_id = orders_details.order_id
13   group by orders.order_date) as sales;
```

Result Grid			Filter Rows:
	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	11929.55	
	2015-01-06	14358.5	
	2015-01-07	16560.7	
	2015-01-08	19399.05	
	2015-01-09	21526.4	

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

```
1  -- Determine the top 3 most ordered pizza types
2  -- based on revenue for each pizza category.
3
4  • select name,revenue from
5
6  (select category,name,revenue,
7   rank() over(partition by category order by revenue desc) as rn
8   from
9   (select pizza_types.category,pizza_types.name,
10    sum((orders_details.quantity)*pizzas.price) as revenue
11   from pizza_types join pizzas
12    on pizza_types.pizza_type_id = pizzas.pizza_type_id
13   join orders_details
14    on orders_details.pizza_id = pizzas.pizza_id
15   group by pizza_types.category, pizza_types.name) as a) as b
16  where rn <= 3 ;
17
```

Result Grid | Filter Rows: | Export:

	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
	The Pepperoni Pizza	30161.75
	The Spicy Italian Pizza	34831.25
	The Italian Supreme Pizza	33476.75
	The Sicilian Pizza	30940.5
	The Four Cheese Pizza	32265.7000000000065

Result 2 x