



Restaurant Analysis Using SQL

Uncover insights and trends with SQL



by [Sireetummuudr Samiuz Zaman](#)



Made with Gamma

Understanding the Data Landscape

2015 (Jan - Dec)

Order Details

Customer details, pizza
order information

Orders

Order ID, date, time, total
amount

Pizza Types

Pizza category, description,
price

Pizzas

Pizza ID, size, price

```
18
19      -- total Revenue (2015 whole year)
20 • with cte as (Select order_id, od.pizza_id, price, quantity, (quantity * price) as Total_price
21   from
22     order_details as od
23   left join pizzas as p
24     on od.pizza_id = p. pizza_id)
25
26   Select round(sum(total_price)) as revenue
27   from cte;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	revenue			
▶	817860			

Total Revenue: 8,17,860\$

for the year 2015

Pizza Category Diversity

Classic

Pepperoni, cheese, etc.

Chicken

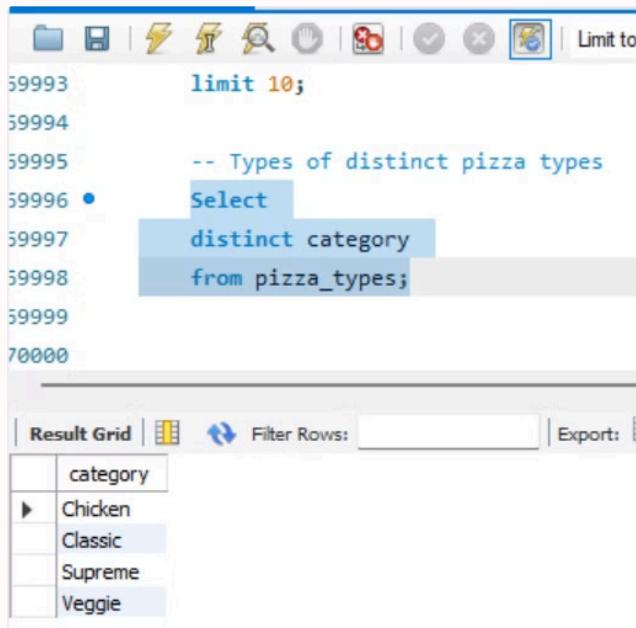
Unique combinations,
gourmet ingredients

Veggie

Meat-free options, fresh
veggies

Supreme

Best ingredients



A screenshot of a MySQL command-line interface. The query is:

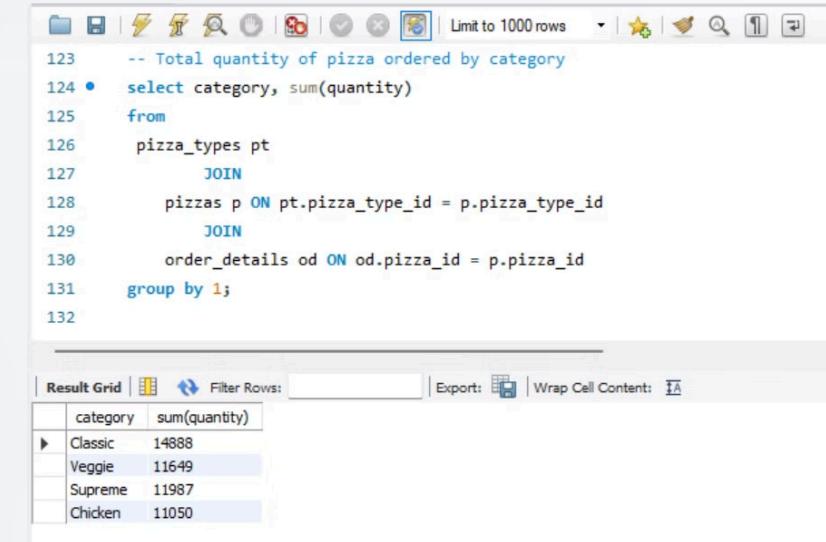
```
59993      limit 10;
59994
59995      -- Types of distinct pizza types
59996 •   Select
59997       distinct category
59998   from pizza_types;
```

The result grid shows the following data:

category
Chicken
Classic
Supreme
Veggie

Quantity ordered by category

Highest ordered category - classic



The screenshot shows a database interface with a query editor and a result grid. The query is as follows:

```
123 -- Total quantity of pizza ordered by category
124 • select category, sum(quantity)
125   from
126     pizza_types pt
127       JOIN
128         pizzas p ON pt.pizza_type_id = p.pizza_type_id
129       JOIN
130         order_details od ON od.pizza_id = p.pizza_id
131   group by 1;
132
```

The result grid displays the following data:

category	sum(quantity)
Classic	14888
Veggie	11649
Supreme	11987
Chicken	11050

pizza store analysis*

pizza analysis* X



```
34      -- max revenue by size -- L size
35 • Select
36      size, round(sum(price * quantity)) as total_Revenue_by_size
37      from order_details od
38      join pizzas as p
39      on od.pizza_id = p.pizza_id
40      group by 1
41      order by 2 desc
42
```

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

	size	total_Revenue_by_size
▶	L	375319
	M	249382
	S	178076
	XL	14076
	XXL	1007

Pizza Size Popularity: L size

Peak Order Times

```
82 -- Peak Hour
83 • with cte as (Select
84     hour(str_to_date(time, '%H:%i:%s')) as timess, order_id
85     from orders)
86     select
87     case
88         when timess between 0 and 11 then 'morning'
89         when timess between 12 and 17 then 'afternoon'
90         when timess between 18 and 23 then 'evening'
91     end as time_of_the_day,
92     count(order_id) as total_order
93     from cte
94     group by 1
95     order by 2 desc;
96
```

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

time_of_the_day	total_order
afternoon	12171
evening	7939
morning	1240



Afternoon

2

Evening

3

Morning

pizza store analysis* pizza analysis*

```
80
81 • with cte as (Select
82     hour(str_to_date(time, '%H:%i:%s')) as timess, order_id
83     from orders)
84
85     select timess, count(order_id) total_order
86     from cte
87     group by 1
88     order by 2 desc;
89
```

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

timess	total_order
12	2520
13	2455
18	2399
17	2336
19	2009
16	1920

Highest Hourly order

12 PM

Highest Monthly Order

July, 2015

```
--  
73 -- Monthly order -- Highest order in July  
74 • with cte as(select * from orders)  
75  
76 select  
77 extract(month from `date`) as month, count(order_id) total_order  
78 from cte  
79 group by 1  
80 order by 2 desc;  
81
```

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

	month	total_order
▶	7	1935
	5	1853
	1	1845
	8	1841
	3	1840
	4	1799
	11	1792
	6	1773
	2	1685



Made with Gamma

The screenshot shows a database query editor window. At the top, there are various icons for file operations, search, and navigation. A toolbar includes a dropdown for 'Limit to 1000 rows' and other standard database tools. The main area displays a SQL script with line numbers 96 through 108. The script uses a Common Table Expression (CTE) to select the highest-priced pizza from a join of two tables: 'pizza_types' and 'pizzas'. The CTE is defined with a self-referencing join on the primary key 'pizza_type_id'. The final query selects all columns from the CTE where the price matches the maximum price found in the CTE. Below the script is a results grid with a header row for 'name', 'price', and 'size'. A single row of data is shown: 'The Greek Pizza' with a price of '35.95' and size 'XXL'. There are buttons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'.

```
96
97      -- Highest priced pizza
98 •  with cte as(
99      select name, price, size
100     from pizza_types as pt
101    join
102      pizzas as p on
103        pt.pizza_type_id = p.pizza_type_id
104      )
105
106    select *
107    from cte
108   where price = (select max(price) from cte);
109
```

	name	price	size
▶	The Greek Pizza	35.95	XXL

Highest Priced pizza

The Greek pizza

TOP PIZZAS

The screenshot shows a MySQL query editor interface. At the top, there's a toolbar with various icons for file operations, search, and navigation. Below the toolbar, the SQL query is displayed:

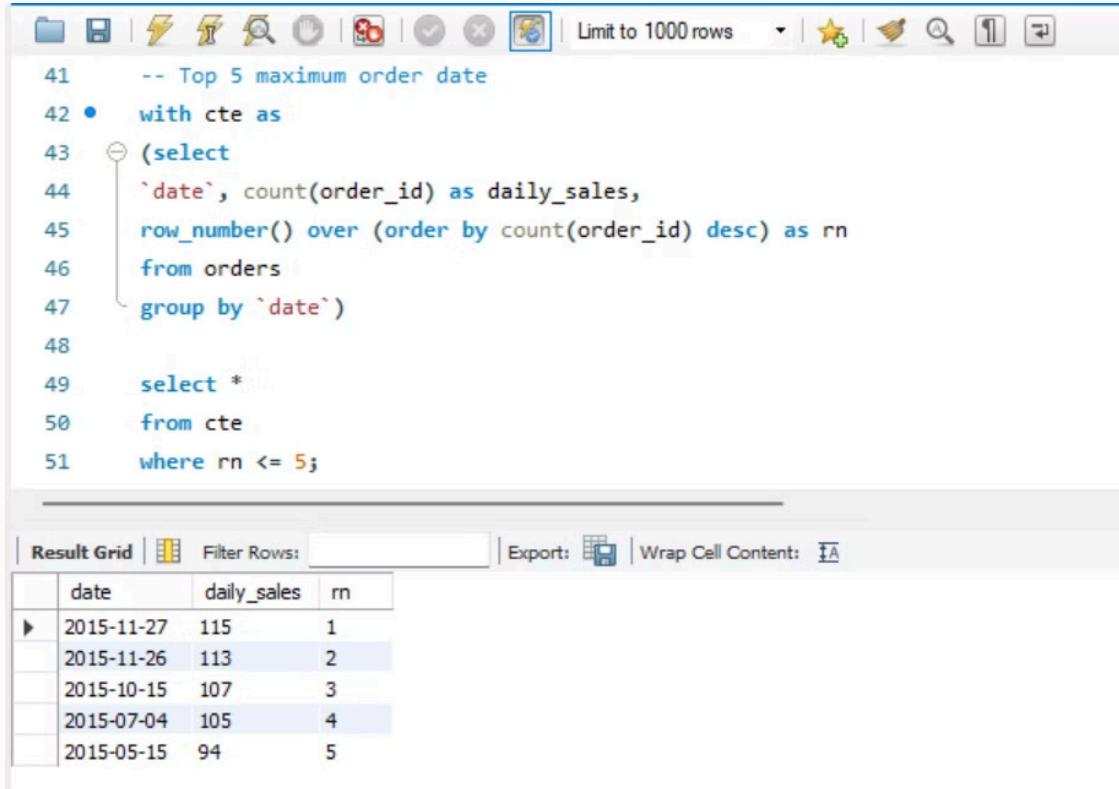
```
110 -- Top 5 most ordered pizza
111 • SELECT
112     name, SUM(quantity)
113 FROM
114     pizza_types pt
115     JOIN
116         pizzas p ON pt.pizza_type_id = p.pizza_type_id
117     JOIN
118         order_details od ON od.pizza_id = p.pizza_id
119 GROUP BY pt.name
120 ORDER BY 2 DESC
121 LIMIT 5;
```

Below the query, there's a "Result Grid" section with a table showing the results:

name	SUM(quantity)
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

Top 5 most ordered pizza (quantity)

TOP PIZZAS



The screenshot shows a database query interface with a toolbar at the top and a code editor below it. The code editor contains the following SQL script:

```
41 -- Top 5 maximum order date
42 • with cte as
43   (select
44     `date`, count(order_id) as daily_sales,
45     row_number() over (order by count(order_id) desc) as rn
46   from orders
47   group by `date`)
48
49 select *
50 from cte
51 where rn <= 5;
```

Below the code editor is a result grid table with three columns: date, daily_sales, and rn. The table displays the following data:

	date	daily_sales	rn
▶	2015-11-27	115	1
	2015-11-26	113	2
	2015-10-15	107	3
	2015-07-04	105	4
	2015-05-15	94	5

Top 5 Max order date

TOP PIZZAS

```
133    -- Top 3 most ordered pizza by revenue
134 •  SELECT
135      name,
136      SUM(quantity) total_quantity,
137      ROUND(SUM(quantity * price)) AS revenue
138  FROM
139    pizza_types pt
140    JOIN
141      pizzas p ON pt.pizza_type_id = p.pizza_type_id
142    JOIN
143      order_details od ON od.pizza_id = p.pizza_id
144  GROUP BY 1
145  ORDER BY 3 DESC
146  LIMIT 3;
```

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

	name	total_quantity	revenue
▶	The Thai Chicken Pizza	2371	43434
	The Barbecue Chicken Pizza	2432	42768
	The California Chicken Pizza	2370	41410

Top 3 most ordered pizza and revenue