



Pizza Sales Performance Using SQL

JOINS, WINDOW FUNCTIONS, COMMON TABLE EXPRESSION, GROUP BY,
HAVING

Dataset

- The dataset that I have worked on for this project is a random Pizza_hut sales.
- This dataset has four tables which are order_details, orders, pizza_types and pizza.
- I have imported these tables into MySQL database and then I have written multiple queries.
- I have utilized CTEs, subquery, WINDOW functions and other important SQL commands to filter, sort and summarize data.
- Created a database to store above tables
- create database pizzahut;

Key questions the project answers

- Retrieve the total number of orders placed.
- Calculate the total revenue generated from pizza sales.
- Identify the highest-priced pizza.
- Identify the most common pizza size ordered.
- List the top 5 most ordered pizza types along with their quantities.
- Join the necessary tables to find the total quantity of each pizza category ordered.
- Determine the distribution of orders by hour of the day.
- Join relevant tables to find the category-wise distribution of pizzas.

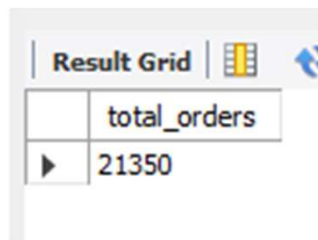


Key questions the project answers

- Group the orders by date and calculate the average number of pizzas ordered per day.
- Determine the top 3 most ordered pizza types based on revenue.
- Calculate the percentage contribution of each pizza type to total revenue.
- Analyze the cumulative revenue generated over time.
- Determine the top 3 most ordered pizza types based on revenue for each pizza category.

Retrieve the total number of orders placed.

- `select count(order_id) as total_orders`
- `from orders;`



A screenshot of a database query result grid. The grid has a header row with the column name 'total_orders' and a data row with the value '21350'. The grid is titled 'Result Grid' and includes icons for a grid, a table, and a refresh button.

Result Grid	
	total_orders
▶	21350

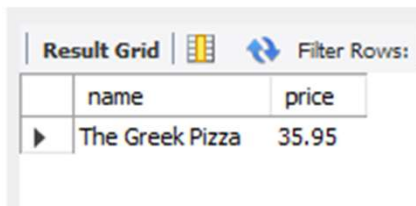
Calculate the total revenue generated from pizza sales.

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

Result Grid	
	total_sales
▶	817860.05

Identify the highest-priced pizza.

- 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;

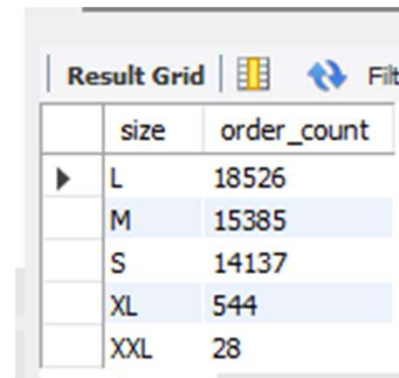


The screenshot shows a database interface with a 'Result Grid' tab. It contains a table with two columns: 'name' and 'price'. The first row of data is 'The Greek Pizza' with a price of 35.95. There are icons for 'Filter Rows' and a table structure icon above the grid.

	name	price
▶	The Greek Pizza	35.95

Identify the most common pizza size ordered.

- 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;



A screenshot of a SQL query result grid. The grid has two columns: 'size' and 'order_count'. The data is sorted in descending order of 'order_count'. The rows are: L (18526), M (15385), S (14137), XL (544), and XXL (28). The grid is titled 'Result Grid' and has a 'Filter' button.

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



List the top 5 most ordered pizza types along with their quantities.

- `select pizza_types.name, sum(order_details.quantity) as total_count`
- `from pizza_types join pizzas`
- `on pizza_types.pizza_type_id=pizzas.pizza_type_id`
- `join order_details`
- `on pizzas.pizza_id=order_details.pizza_id`
- `group by pizza_types.name order by total_count desc`
- `limit 5;`

Result Grid



Filter Rows:

	name	total_count
▶	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.

- `select pizza_types.category, sum(order_details.quantity) as total_count`
`from pizza_types join pizzas`
- `on pizza_types.pizza_type_id=pizzas.pizza_type_id`
- `join order_details`
- `on pizzas.pizza_id=order_details.pizza_id`
- `group by pizza_types.category order by total_count;`

Result Grid

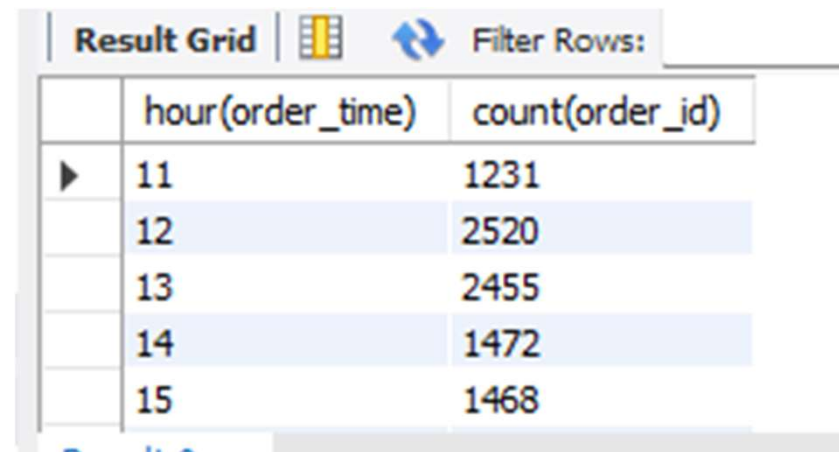


Filter Rows:

	category	total_count
▶	Chicken	11050
	Veggie	11649
	Supreme	11987
	Classic	14888

Determine the distribution of orders by hour of the day.

- `select hour(order_time), count(order_id)`
- `from orders`
- `group by hour(order_time);`

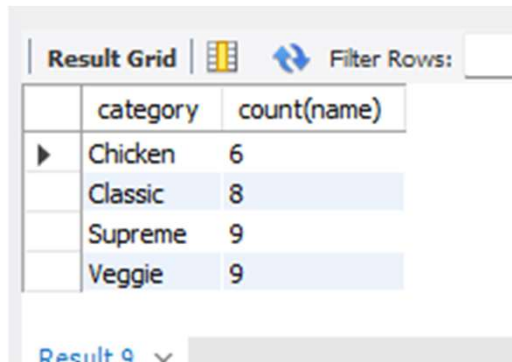


The screenshot shows a 'Result Grid' interface with a table of query results. The table has two columns: 'hour(order_time)' and 'count(order_id)'. The data is as follows:

	hour(order_time)	count(order_id)
▶	11	1231
	12	2520
	13	2455
	14	1472
	15	1468

Join relevant tables to find the category-wise distribution of pizzas.

- select category, count(name)
- from pizza_types
- group by category;



The screenshot shows a database interface with a 'Result Grid' tab. It displays a table with two columns: 'category' and 'count(name)'. The table contains four rows of data: 'Chicken' with a count of 6, 'Classic' with a count of 8, 'Supreme' with a count of 9, and 'Veggie' with a count of 9. The 'Result Grid' tab is selected, and there is a 'Filter Rows:' input field to the right. At the bottom, it says 'Result 9' with a dropdown arrow.

	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.

- `select avg(quantity)`
- `from(select orders.order_date, sum(order_details.quantity) as quantity`
- `from orders join order_details`
- `on orders.order_id=order_details.order_id`
- `group by orders.order_date) as order_quantity;`

Result Grid		Filter
	avg(quantity)	
▶	138.4749	

Determine the top 3 most ordered pizza types based on revenue.

- `select pizza_types.name, sum(order_details.quantity*pizzas.price) 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 3;`

Result Grid





Filter Rows:

Es

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

Analyze the cumulative revenue generated over time.

- `select order_date, sum(revenue) over(order by order_date) as cum_revenue`
- `from (select orders.order_date, sum(order_details.quantity*pizzas.price) as revenue`
- `from orders join order_details`
- `on orders.order_id=order_details.order_id`
- `join pizzas`
- `on order_details.pizza_id=pizzas.pizza_id`
- `group by orders.order_date) as sales;`

Result Grid   Filter Rows:

	order_date	cum_revenue
▶	2015-01-01	2713.85000000000004
	2015-01-02	5445.75
	2015-01-03	8108.15
	2015-01-04	9863.6
	2015-01-05	11929.55

Result 12 ▼

Determine the top 3 most ordered pizza types based on revenue for each pizza category.

- ▶ `select *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 pizzas.pizza_id=order_details.pizza_id`
- ▶ `group by pizza_types.category, pizza_types.name) as a) as b`
- ▶ `where rn <=3;`



Result Grid				
		Filter Rows:		
		Export:		
		Wrap Ce		
	category	name	revenue	rn
►	Chicken	The Thai Chicken Pizza	43434.25	1
	Chicken	The Barbecue Chicken Pizza	42768	2
	Chicken	The California Chicken Pizza	41409.5	3
	Classic	The Classic Deluxe Pizza	38180.5	1
	Classic	The Hawaiian Pizza	32273.25	2
	Classic	The Pepperoni Pizza	30161.75	3
	Supreme	The Spicy Italian Pizza	34831.25	1
	Supreme	The Italian Supreme Pizza	33476.75	2
	Supreme	The Sicilian Pizza	30940.5	3
	Veggie	The Four Cheese Pizza	32265.700000000065	1

Result 14

Calculate the percentage contribution of each pizza type to total revenue.

- `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 order_details.pizza_id=pizzas.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 pizzas.pizza_id=order_details.pizza_id`
- `group by pizza_types.category order by revenue desc;`

Result Grid

Filter Rows

	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68