# Basic:

Retrieve the total number of orders placed.

SELECT COUNT (order id) as total orders FROM orders;

```
Your SQL query has been executed successfully.

-- Retrieve the total number of orders placed. SELECT COUNT(order_id) as total_orders FROM orders;

[Edit inline] [Edit] [Create PHP code]

Extra options

total_orders

21350
```

Calculate the total revenue generated from pizza sales.

SELECT Round(SUM(order\_details.quantity \* pizzas.price),2) as total\_sales FRO
M order\_details JOIN pizzas on order\_details.pizza\_id = pizzas.pizza\_id;

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;
[Edit   Create PHP code ]
□ Show all   Number of rows: 25 ∨ Filter rows: Search this table
Extra options
total_sales 817860.05

Identify the highest-priced pizza.

SELECT pizza\_types.name, pizzas.price FROM pizza\_types JOIN pizzas ON pizza\_t
ypes.pizza type id = pizzas.pizza type id ORDER BY pizzas.price DESC LIMIT 1;

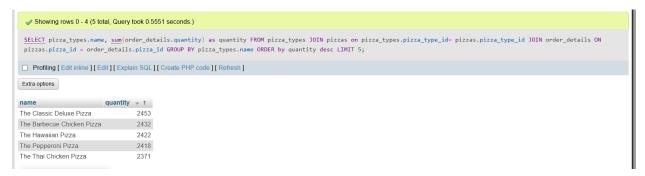
Identify the most common pizza size ordered.

SELECT pizzas.size, COUNT(order\_details.order\_details\_id) as common\_size from
pizzas JOIN order\_details on pizzas.pizza\_id=order\_details.pizza\_id GROUP BY
pizzas.size;



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

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 pizzas.pizza\_id = order\_details.pizza\_id GROUP BY pizza\_types.name ORDER by quantity desc LIMIT 5;



# Intermediate:

Join the necessary tables to find the total quantity of each pizza category ordered.

SELECT pizza\_types.category, SUM (order\_details.quantity) AS totalprice\_catego
ry FROM pizza\_types JOIN pizzas ON pizza\_types.pizza\_type\_id = pizzas.pizza\_t
ype\_id JOIN order\_details ON pizzas.pizza\_id = order\_details.pizza\_id GROUP B
Y pizza types.category ORDER BY quantity DESC;



Determine the distribution of orders by hour of the day.

SELECT HOUR(time) AS HOUR, COUNT (order\_id) AS HOUR\_COUNT FROM orders GROUP B
Y HOUR(time);



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

SELECT pizza\_types.category, COUNT(pizzas.pizza\_id) AS total\_pizzas FROM pizz
as JOIN pizza\_types ON pizza\_types.pizza\_type\_id = pizzas.pizza\_type\_id GROUP
BY pizza\_types.category;



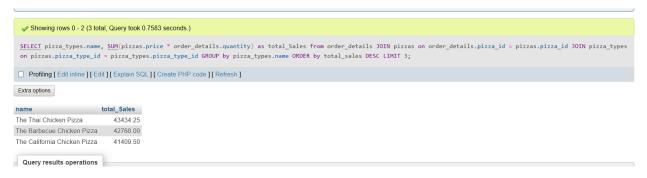
Group the orders by date and calculate the average number of pizzas ordered per day.

SELECT round(AVG(daily\_total),0) AS avg\_pizzas\_per\_day FROM ( SELECT orders.d
ate, SUM(order\_details.quantity) AS daily\_total FROM orders JOIN order\_detail
s ON orders.order\_id = order\_details.order\_id GROUP BY orders.date ) AS daily
\_summary;



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

SELECT pizza\_types.name, SUM(pizzas.price \* order\_details.quantity) as total\_
Sales from order\_details JOIN pizzas on order\_details.pizza\_id = pizzas.pizza
\_id JOIN pizza\_types on pizzas.pizza\_type\_id = pizza\_types.pizza\_type\_id GROU
P by pizza types.name ORDER by total sales DESC LIMIT 3;



### Advanced:

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

SELECT pizza\_types.category, ROUND(SUM(pizzas.price \* order\_details.quantity), 2) AS category\_sales, ROUND(100.0 \* SUM(pizzas.price \* order\_details.quantity) / (SELECT SUM(order\_details.quantity \* pizzas.price) FROM order\_details

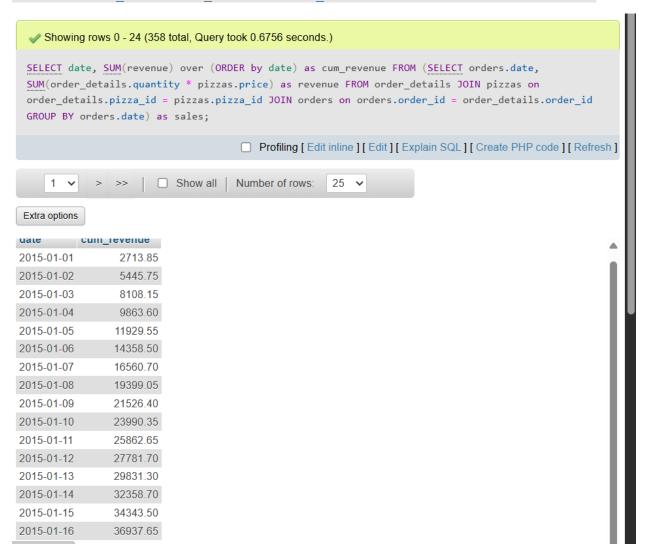
JOIN pizzas ON order\_details.pizza\_id = pizzas.pizza\_id), 2) AS percentage\_of

\_total\_sales FROM order\_details JOIN pizzas ON order\_details.pizza\_id = pizza
s.pizza\_id JOIN pizza\_types ON pizzas.pizza\_type\_id = pizza\_types.pizza\_type\_
id GROUP BY pizza types.category ORDER BY percentage of total sales DESC;



# Analyze the cumulative revenue generated over time.

SELECT date, SUM(revenue) over (ORDER by date) as cum\_revenue FROM (SELECT or
ders.date, SUM(order\_details.quantity \* pizzas.price) as revenue FROM order\_d
etails JOIN pizzas on order\_details.pizza\_id = pizzas.pizza\_id JOIN orders on
 orders.order id = order details.order id GROUP BY orders.date) as sales;



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

```
SELECT name, revenue FROM ( SELECT category, name, revenue, RANK() OVER (PART ITION BY category ORDER BY revenue DESC) AS rn FROM ( SELECT pizza_types.cate gory, 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;
```

# Showing rows 0 - 11 (12 total, Query took 0.7174 seconds.) 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; Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh] Show all | Number of rows: 25 \times Extra options

### name revenue The Thai Chicken Pizza 43434.25 The Barbecue Chicken Pizza 42768.00 The California Chicken Pizza 41409.50 The Classic Deluxe Pizza 38180.50 The Hawaiian Pizza 32273.25 The Pepperoni Pizza 30161.75 The Spicy Italian Pizza 34831.25 33476.75 The Italian Supreme Pizza The Sicilian Pizza 30940.50

32265.70

26780.75

26066.50

The Four Cheese Pizza

The Five Cheese Pizza

The Mexicana Pizza