

# PIZZA SALES ANALYSIS



#### Introduction

This project is based on a Pizza Sales Dataset and focuses on analyzing sales performance using SQL. The goal was to explore business insights from raw transactional data by solving queries ranging from basic to advanced levels.



# Retrieve the total number of orders placed.

#### Solution

```
• SELECT

COUNT(order_id) AS total_orders

FROM

orders;
```

Result Grid

21350

total\_orders





### Calculate the total revenue generated from pizza sales.

#### Solution

```
* SELECT

PROUND(SUM(order_details.quantity * pizzas.price),
2) AS Total_Revenue

FROM

order_details

JOIN

pizzas ON order_details.pizza_id = pizzas.pizza_id;
```

Total\_Revenue

817860.05

Result Grid



#### Identify the highest-priced pizza.

#### Solution

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

Filter Rows:

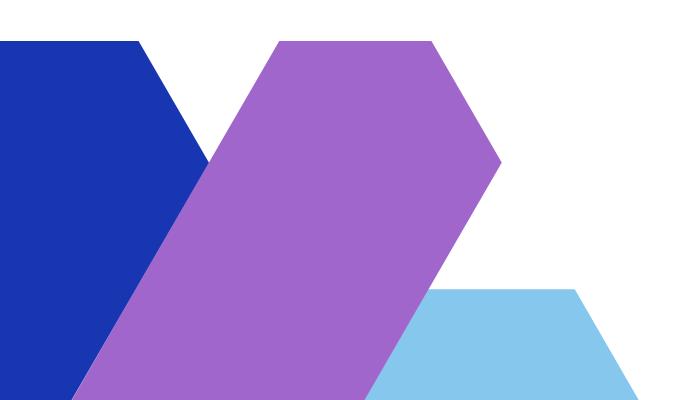
price

35.95

Result Grid

name

The Greek Pizza





### Identify the most common pizza size ordered.

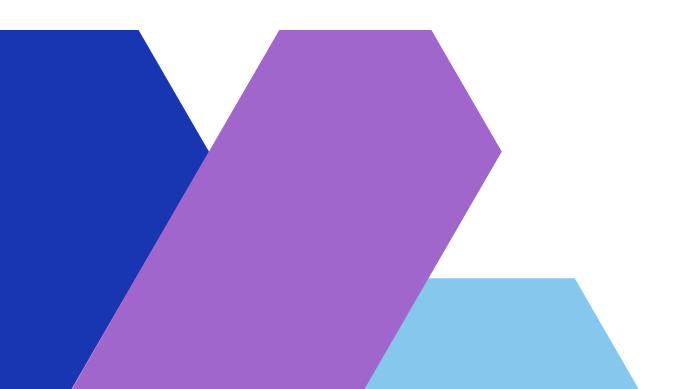
```
pizzas.size, COUNT(order_details.quantity) AS order_count
FROM

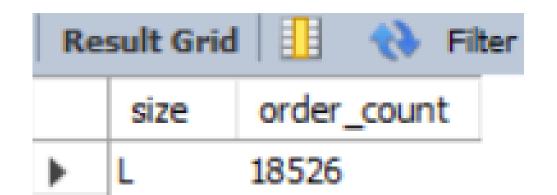
pizzas

JOIN

order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size

ORDER BY order_count DESC
LIMIT 1;
```

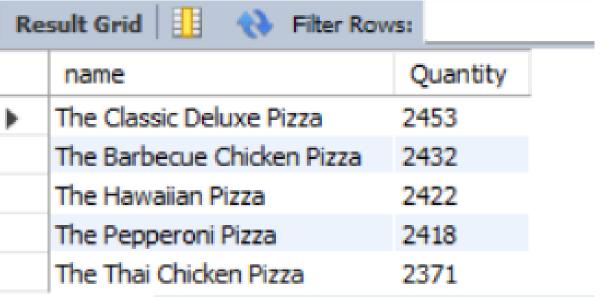






# **Problem 5** List the top 5 most ordered pizza types along with their quantities.

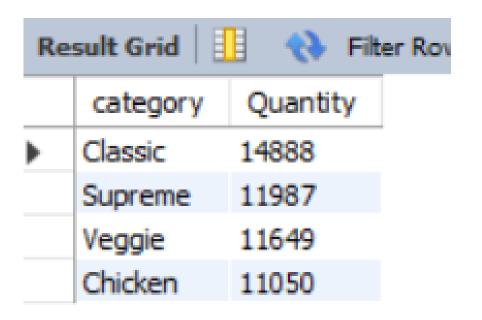
SELECT





### **Problem 6** Join the necessary tables to find the total quantity of each pizza category ordered.

```
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;
```





# **Problem 7** Determine the distribution of orders by hour of the day.

Result Grid

HOUR(order\_time)

#### Solution

```
SELECT

HOUR(order_time), COUNT(order_id)

FROM

orders

GROUP BY HOUR(order_time);
```

Filter Rows:

1231

2520

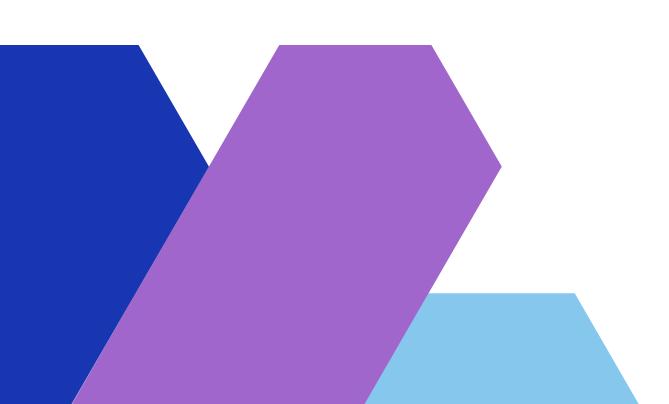
2455

1472

1468

1920

COUNT(order\_id)





#### Join relevant tables to find the categorywise distribution of pizzas.

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

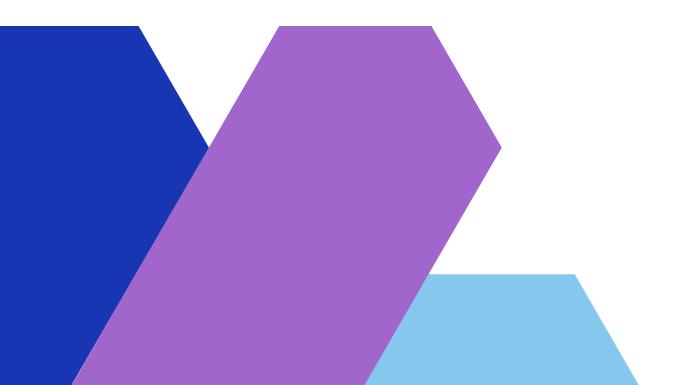


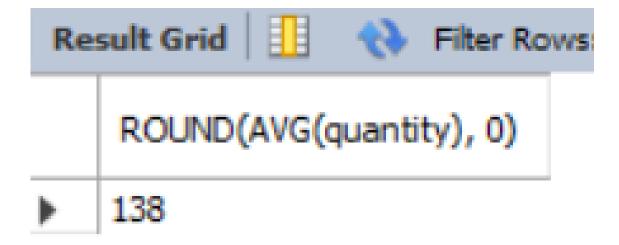


# Group the orders by date and calculate the average number of pizzas ordered per day.

```
SELECT
    ROUND(AVG(quantity), 0)
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;
```



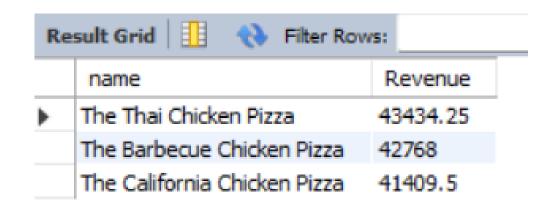




#### Solution

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

```
SELECT
    pizza_types.name,
    SUM(pizzas.price * order_details.quantity) AS Revenue
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
GROUP BY pizza_types.name
ORDER BY Revenue DESC
LIMIT 3;
```



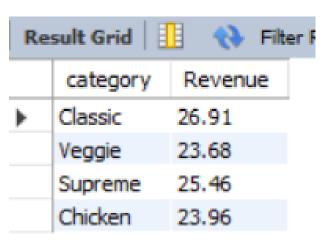


#### Solution



### 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_Revenue
                FROM
                    order_details
                        JOIN
                    pizzas ON order_details.pizza_id = pizzas.pizza_id) * 100,
            2) AS Revenue
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
        JOIN
    pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY category;
```





## Analyze the cumulative revenue generated over time.

```
select order_date,sum(revenue) over(order by order_date) as cumulative_revenue
from

(select orders.order_date,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_date) as sales;
```





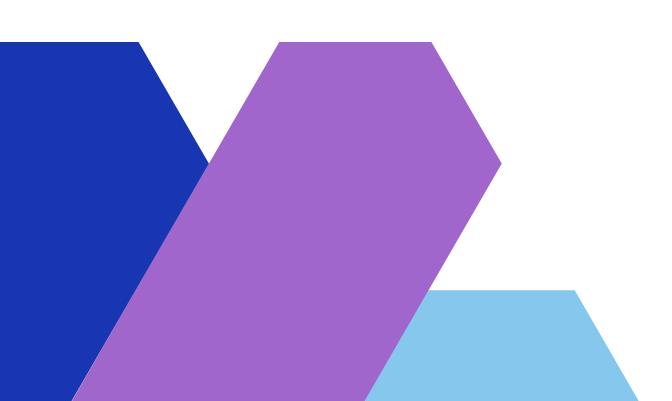
### Determine the top 3 most ordered pizza types based on revenue for each pizza category.

| Result Grid |          |                              |                   |
|-------------|----------|------------------------------|-------------------|
|             | category | name                         | revenue           |
| •           | Chicken  | The Thai Chicken Pizza       | 43434.25          |
|             | Chicken  | The Barbecue Chicken Pizza   | 42768             |
|             | Chicken  | The California Chicken Pizza | 41409.5           |
|             | Classic  | The Classic Deluxe Pizza     | 38180.5           |
|             | Classic  | The Hawaiian Pizza           | 32273.25          |
|             | Classic  | The Pepperoni Pizza          | 30161.75          |
|             | Supreme  | The Spicy Italian Pizza      | 34831.25          |
|             | Supreme  | The Italian Supreme Pizza    | 33476.75          |
|             | Supreme  | The Sicilian Pizza           | 30940.5           |
|             | Veggie   | The Four Cheese Pizza        | 32265.70000000065 |
|             | Veggie   | The Mexicana Pizza           | 26780.75          |
|             | Veggie   | The Five Cheese Pizza        | 26066.5           |

```
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;
```



#### THANK YOU



Project By: Sunil Shetty