



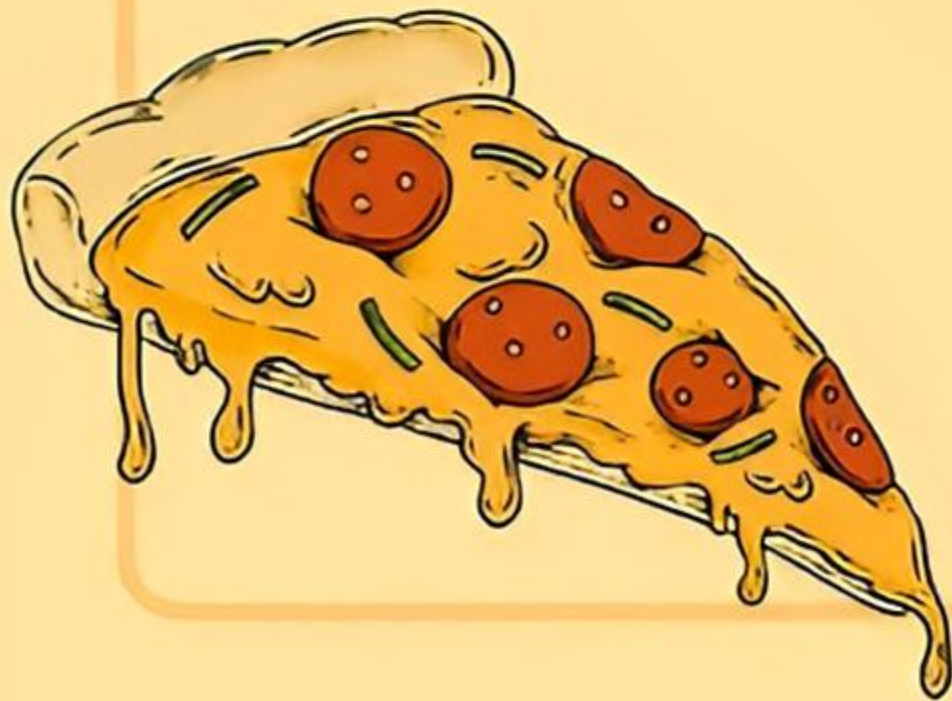
PIZZA

Sales Analysis using SQL

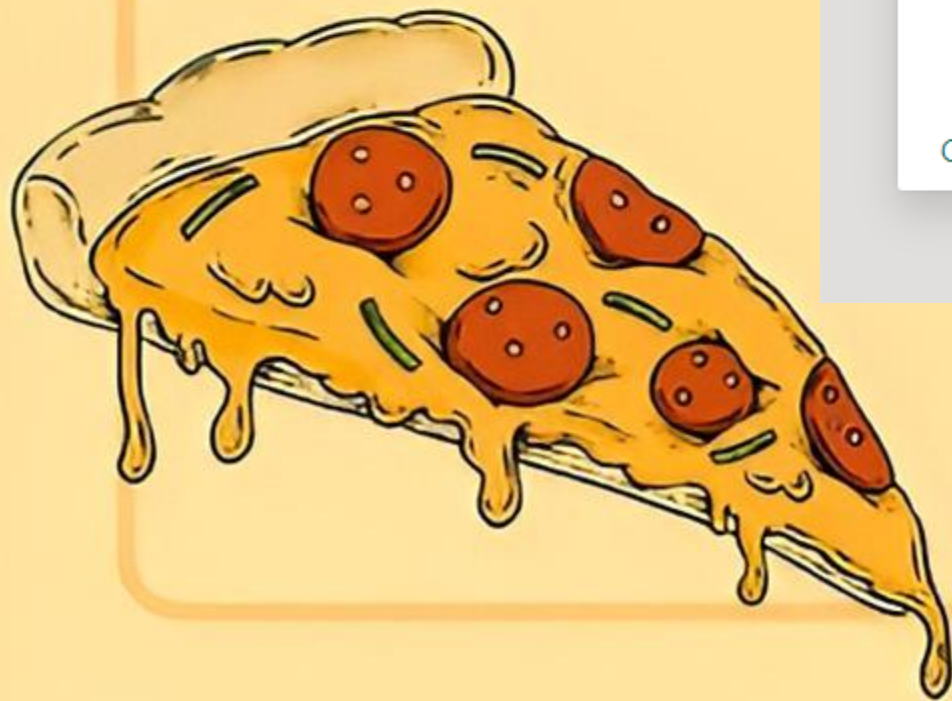
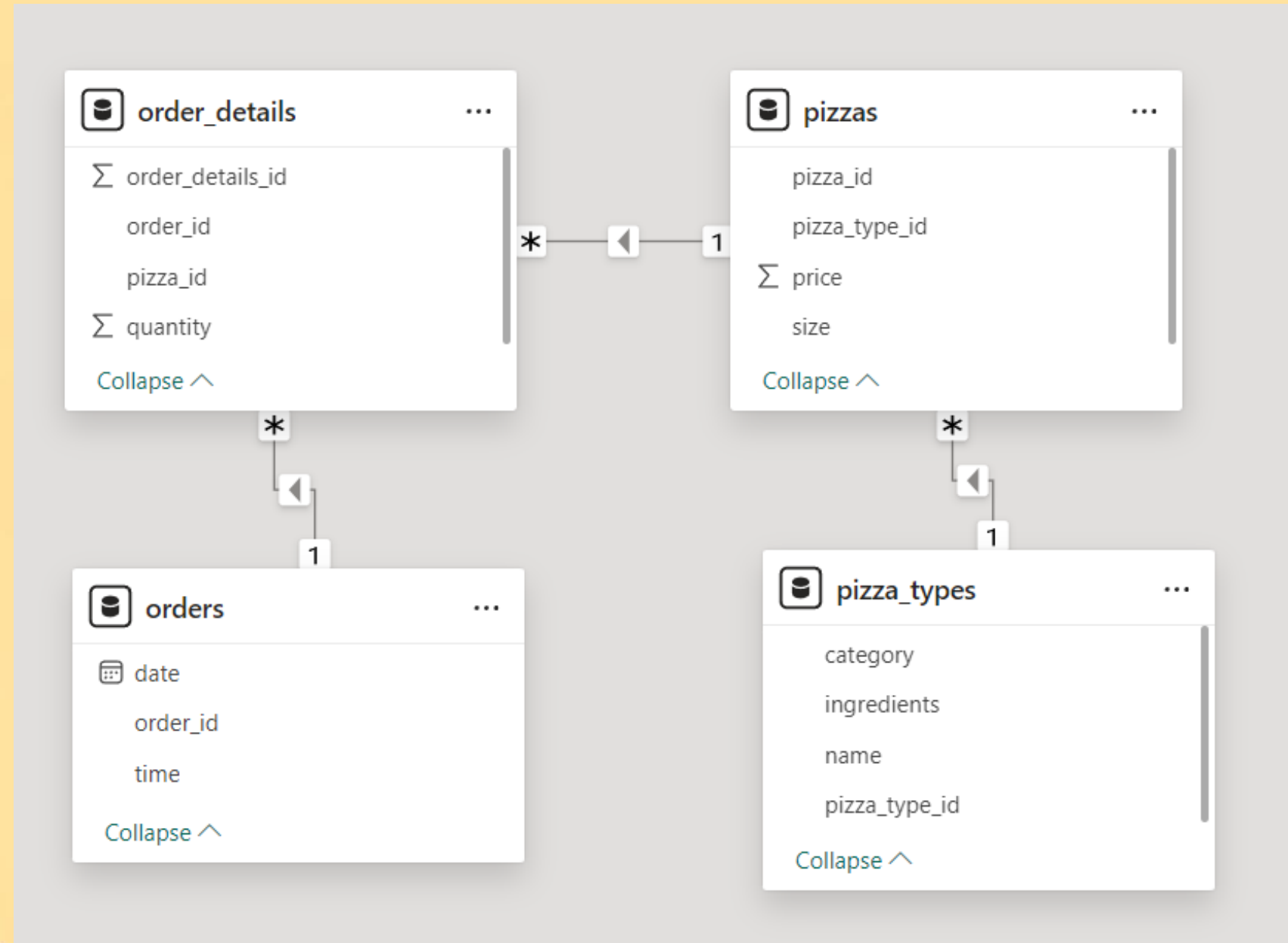
Presented by Sanket Taral

Preface

This project focuses on pizza sales analysis, showcased on a SQL platform. It demonstrates the utilization of aggregate functions, joins and more to extract insights from the pizza database.



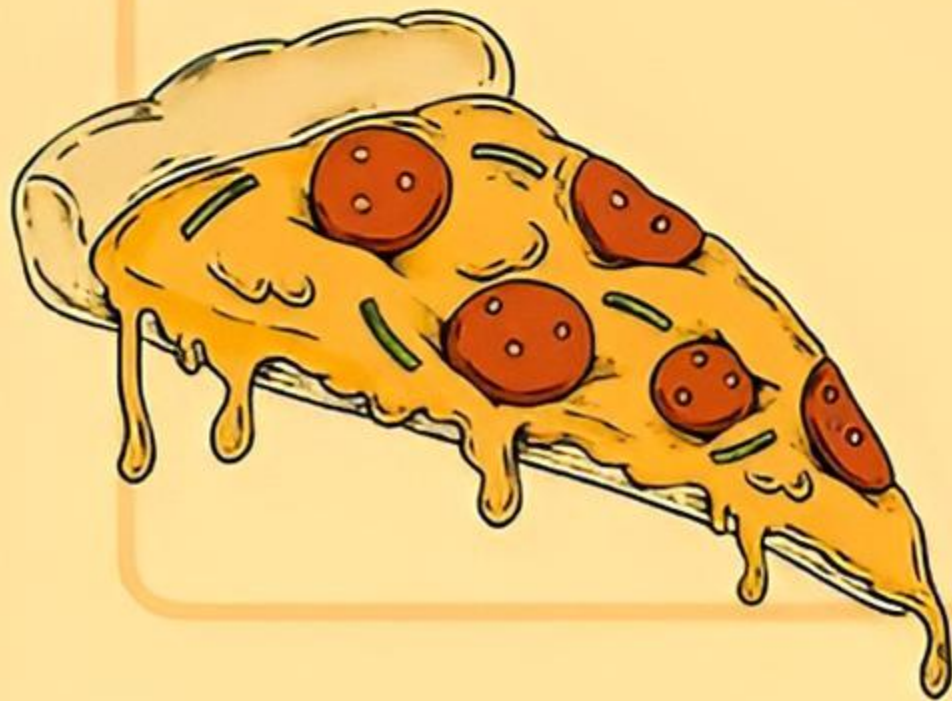
Data Model View



1) Retrieve the total number of orders placed.

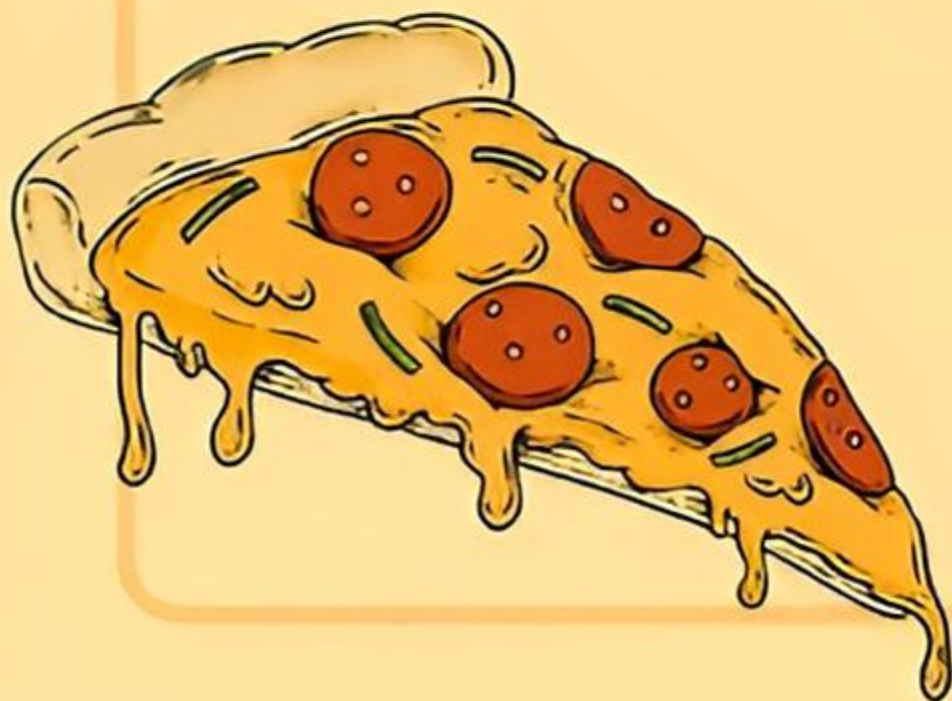
```
SELECT  
    COUNT(order_id) AS Total_orders  
FROM  
    orders;
```

	Total_orders
▶	21350



2) Calculate the total revenue generated from pizza sales.

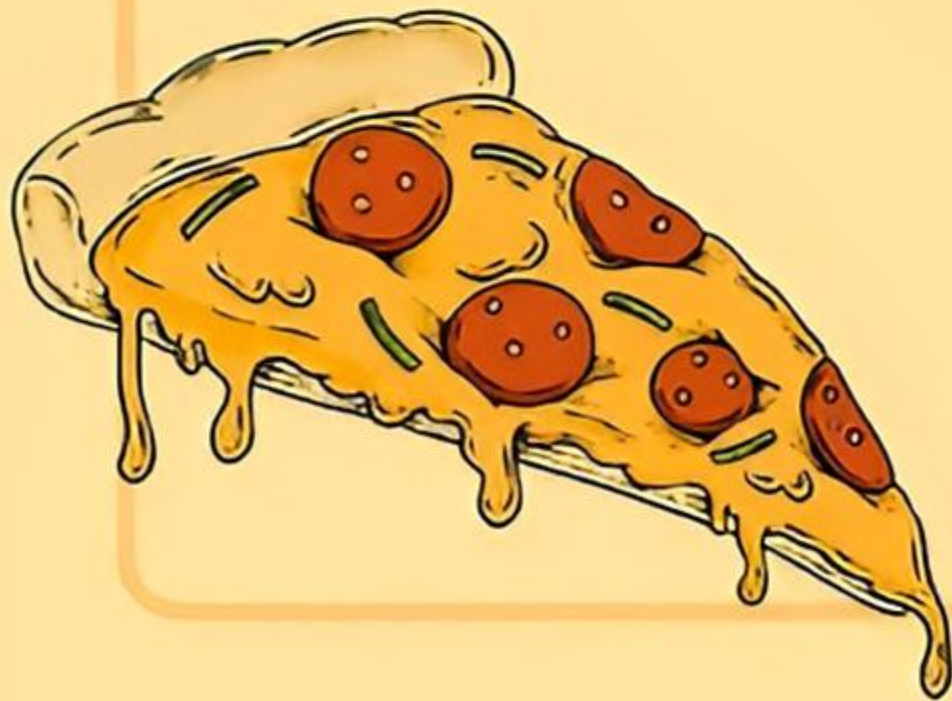
```
SELECT
    ROUND(SUM(order_details.quantity * pizzas.price),
          2) AS Total_sales
FROM
    order_details
    JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id;
```



	Total_sales
▶	817860.05

3) Identify the highest-priced pizza.

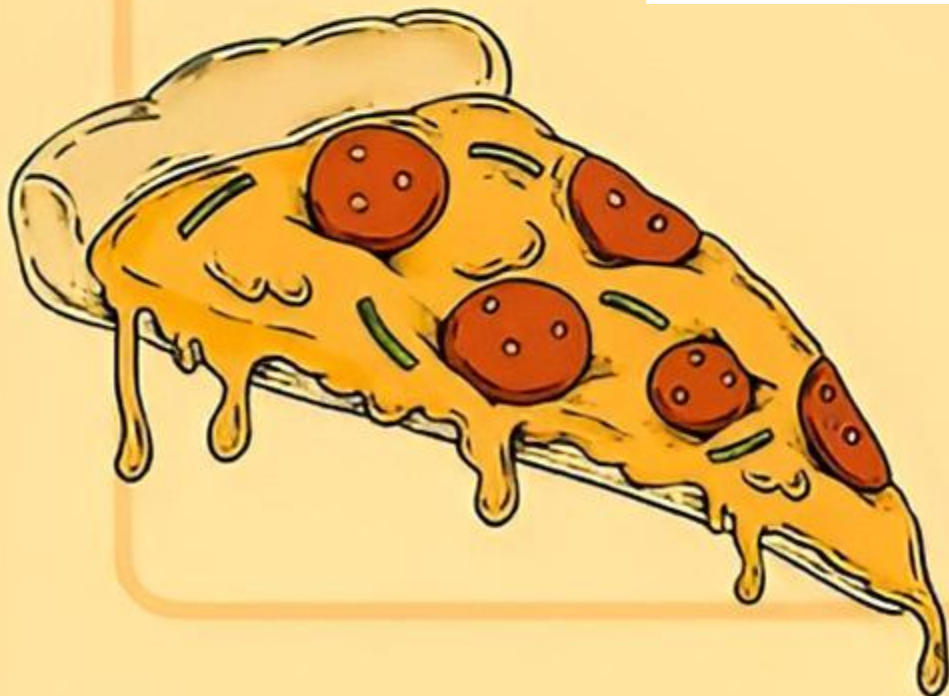
```
SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
    JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
```



	name	price
►	The Greek Pizza	35.95

4) Identify the most common pizza size ordered.

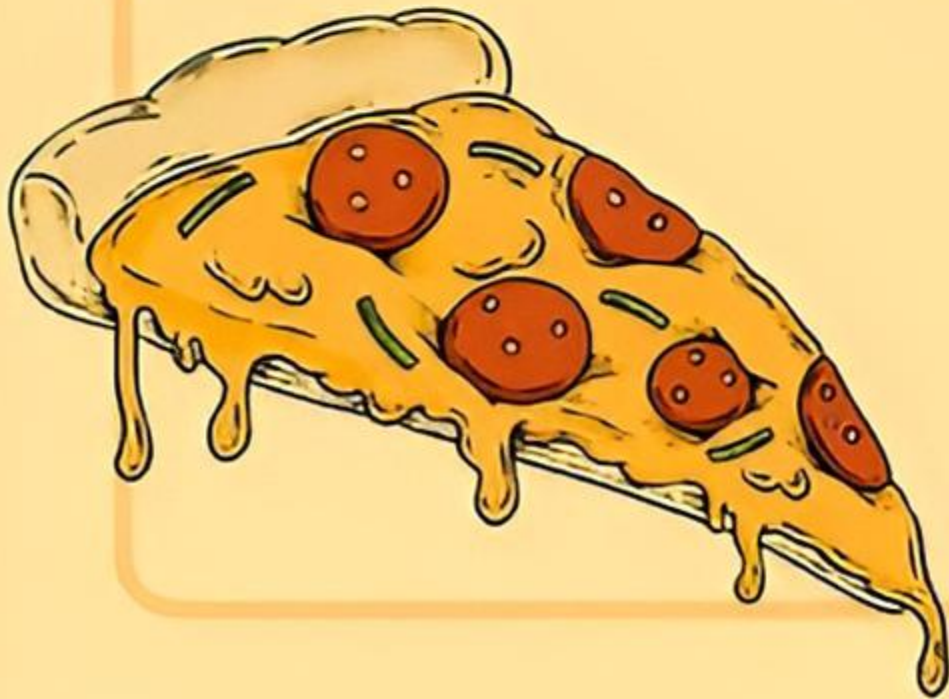
```
SELECT
    pizzas.size,
    COUNT(order_details.order_details_id) AS order_count
FROM
    pizzas
    JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC;
```



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

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

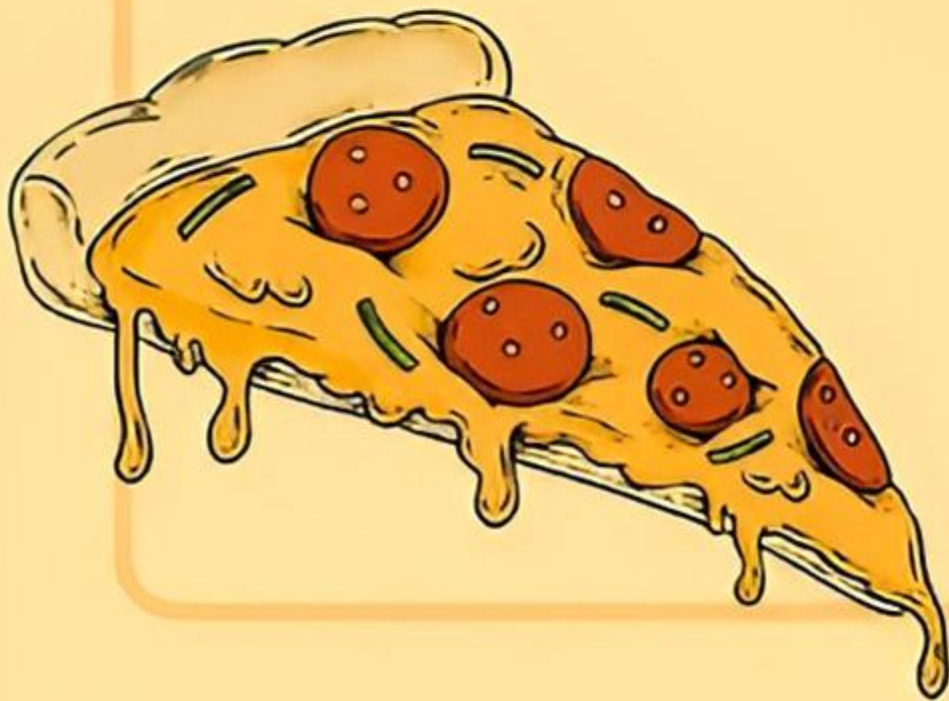
```
SELECT
    pizza_types.name, SUM(order_details.quantity) AS Total_sum
FROM
    pizza_types
    JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
    JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY Total_sum DESC
LIMIT 5;
```



	name	Total_sum
►	The Classic Deluxe Pizza	2453
	The Barbecue Chicken Pizza	2432
	The Hawaiian Pizza	2422
	The Pepperoni Pizza	2418
	The Thai Chicken Pizza	2371

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

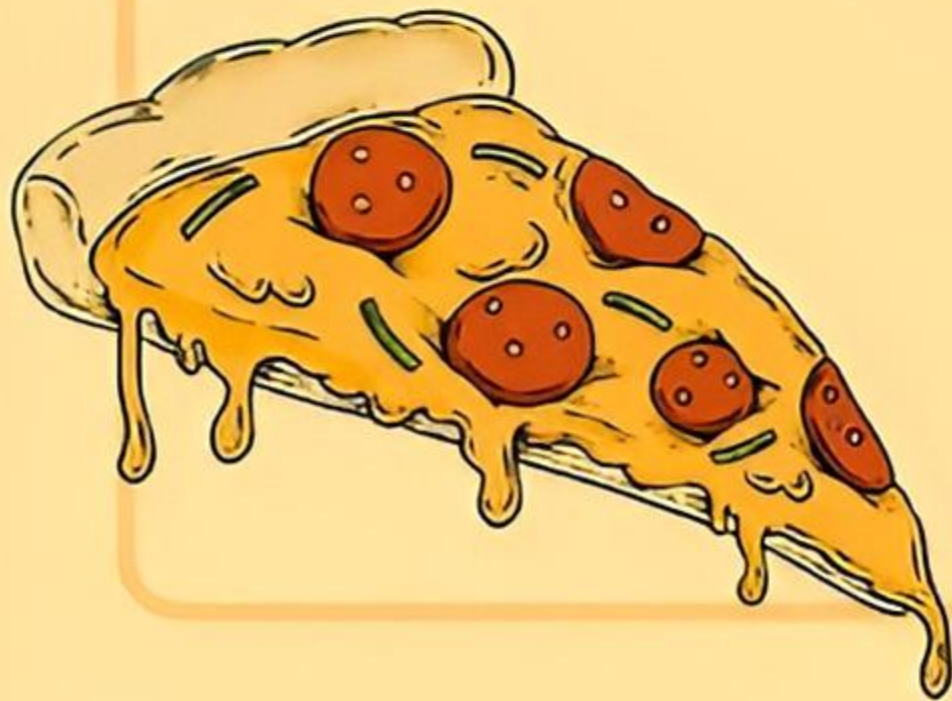
```
SELECT
    pizza_types.category, SUM(order_details.quantity)
FROM
    pizza_types
    JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
    JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizza_types.category;
```



	category	SUM(order_details.quantity)
▶	Classic	14888
	Veggie	11649
	Supreme	11987
	Chicken	11050

7) Determine the distribution of orders by hour of the day.

```
SELECT
    HOUR(order_time) AS hour, COUNT(order_id) AS order_count
FROM
    orders
GROUP BY hour;
```

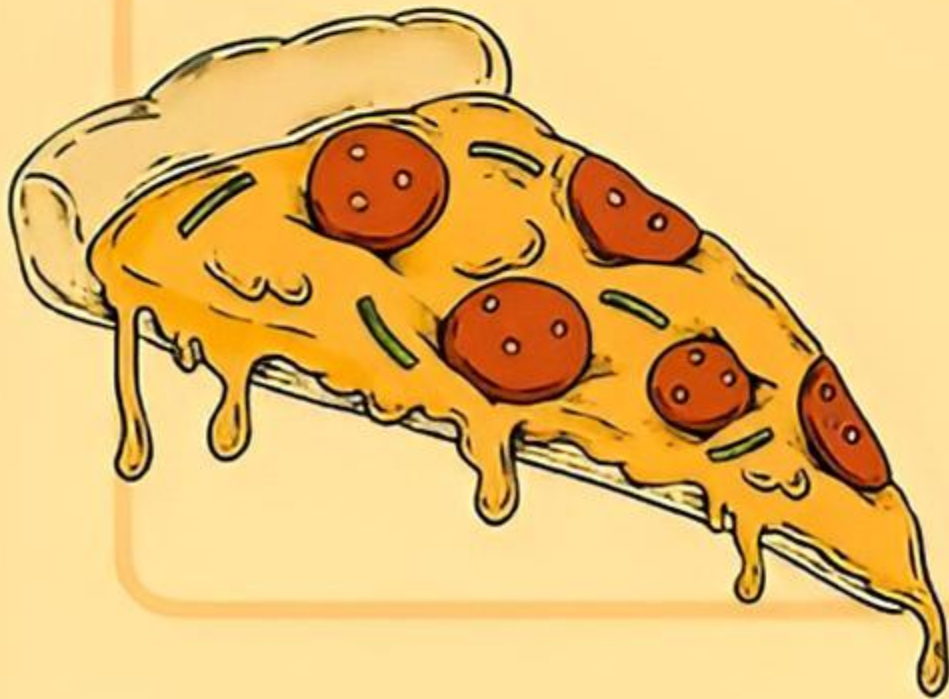


	hour	order_count
▶	11	1231
	12	2520
	13	2455
	14	1472
	15	1468
	16	1920
	17	2336
	18	2399
	19	2009
	20	1642
	21	1198
	22	663
	23	28
	10	8
	9	1

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

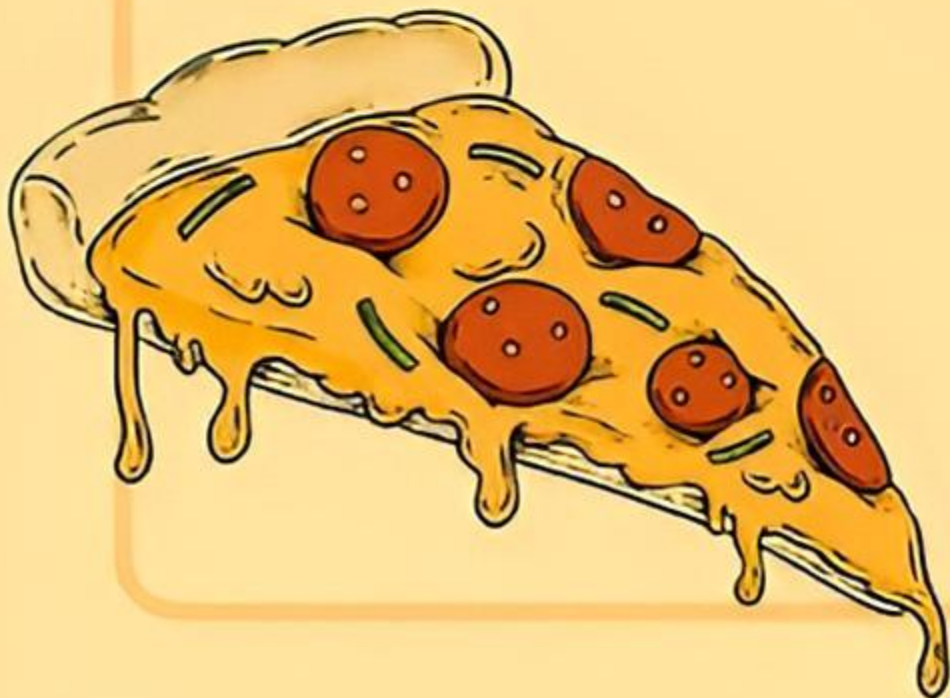
```
SELECT  
    category, COUNT(name)  
FROM  
    pizza_types  
GROUP BY category;
```

	category	COUNT(name)
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9



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

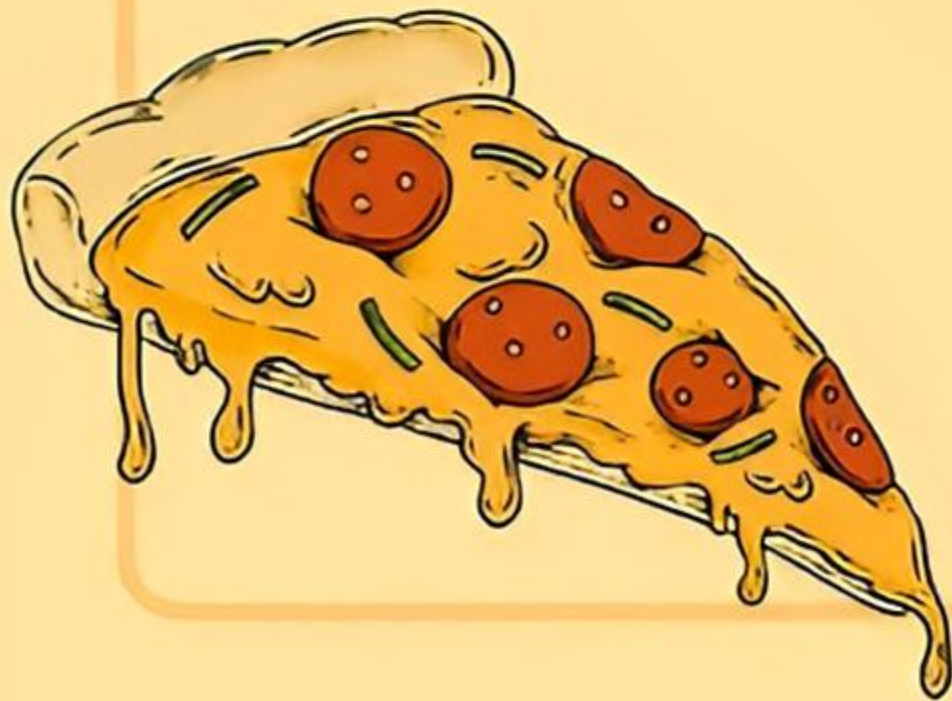
```
SELECT
    ROUND(AVG(quantity), 0) AS avg_pizza_order_per_day
FROM
    (SELECT
        orders.order_date, SUM(order_details.quantity) AS quantity
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY orders.order_date) AS order_quantity;
```



	avg_pizza_order_per_day
▶	138

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

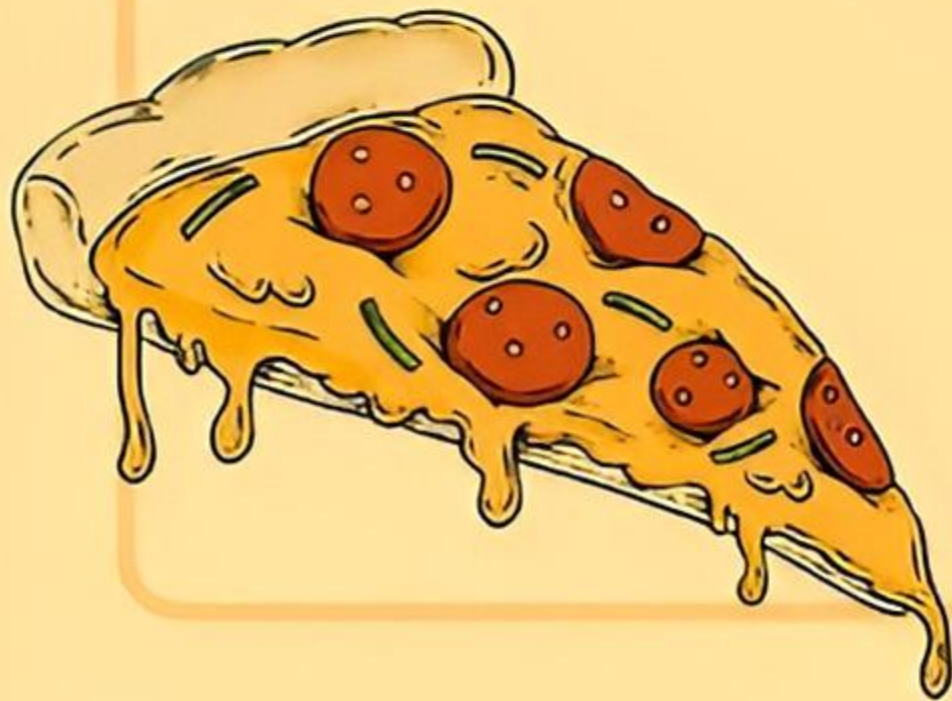
```
SELECT
    pizza_types.name,
    SUM(order_details.quantity * pizzas.price) AS revenue
FROM
    pizza_types
    JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
    JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```



	name	revenue
►	The Thai Chicken Pizza	43434.25
	The Barbecue Chicken Pizza	42768
	The California Chicken Pizza	41409.5

11) 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 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;
```



	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
	2015-01-10	23990.350000000002
	2015-01-11	25862.65
	2015-01-12	27781.7
	2015-01-13	29831.300000000003
	2015-01-14	32358.700000000004
	2015-01-15	34343.500000000001

Thank You ;)

