



# SQL PROJECT ON PIZZA SALES



**Himanshu Rajput**  
Data Analyst Trainee  
AlmaBetter



# Project Overview

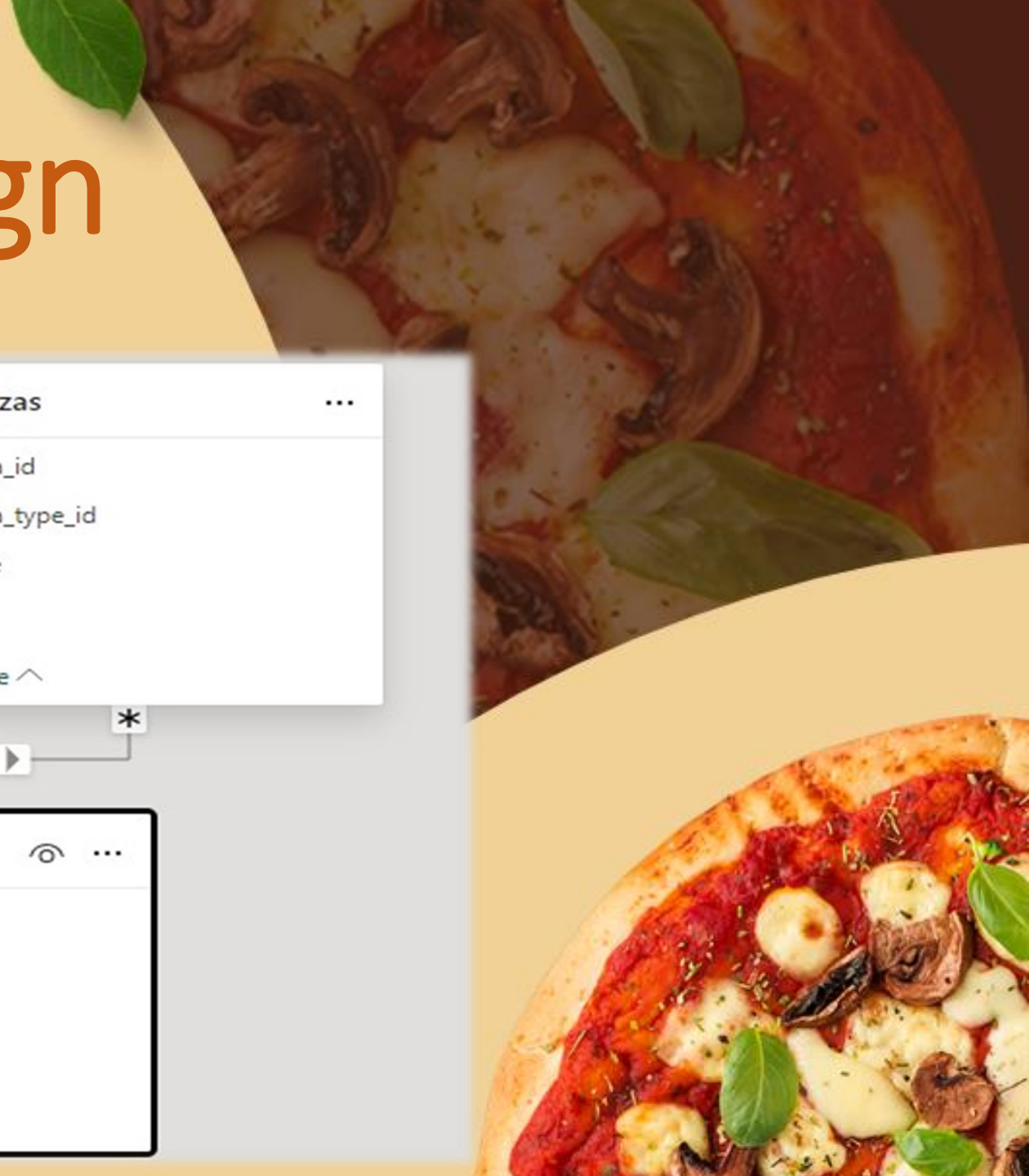
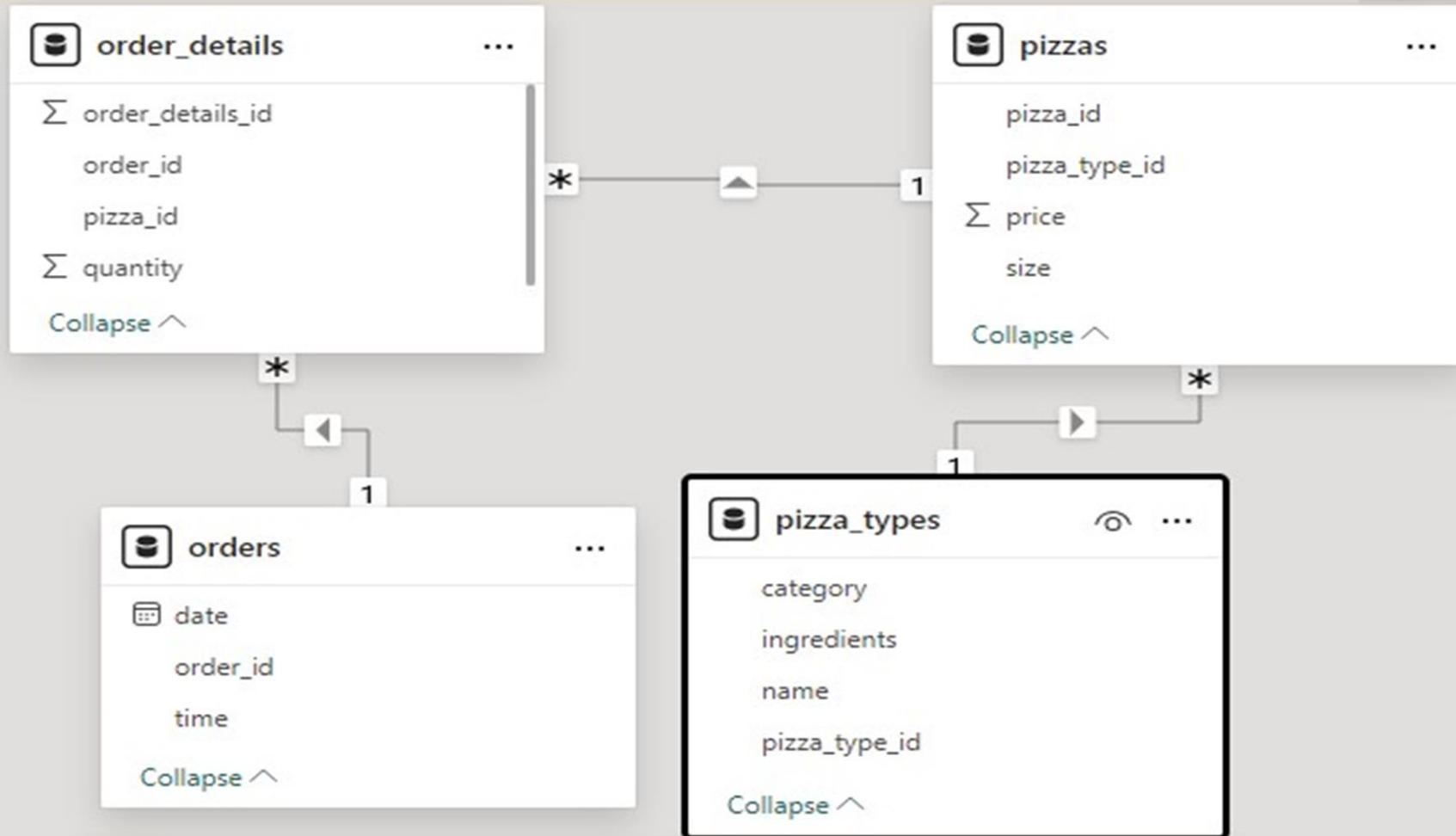
This SQL project revolves around a database schema designed to manage and analyze data for a pizza store. The database consists of four primary tables: `order_details`, `pizzas`, `orders`, and `pizza_types`. Each table plays a crucial role in storing different facets of the business operations, from individual orders to the types of pizzas offered. Below is a detailed description of each table and its columns:








# Schema Design





Retrieve the total number of orders placed.

```
2 • SELECT
3     COUNT(order_id) AS total_orders
4 FROM
5     orders;
```



Result Grid



Filter Rows:

Export

	total_orders
▶	21350





Calculate the total revenue generated from pizza sales.

SELECT

ROUND(SUM(od.quantity \* ps.price), 2) AS total\_sales

FROM

order\_details AS od

JOIN

pizzas AS ps ON ps.pizza\_id = od.pizza\_id;

	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;
```

	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;
```

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

XXL 28




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

```
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 t;
```

	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
	Chicken	The Southwest Chicken Pizza	34705.75	4
	Chicken	The Chicken Alfredo Pizza	16900.25	5
	Chicken	The Chicken Pesto Pizza	16701.75	6
	Classic	The Classic Deluxe Pizza	38180.5	1
	Classic	The Hawaiian Pizza	32273.25	2
	Classic	The Pepperoni Pizza	30161.75	3
	Classic	The Greek Pizza	28454.100000000013	4
	Classic	The Italian Capocollo Pizza	25094	5
	Classic	The Italian Capocollo Pizza	52084	2
	Classic	The Greek Pizza	58424.100000000013	4
	Classic	The Pepperoni Pizza	30161.75	3






# 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;
```

	category	quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

Chicken 11020



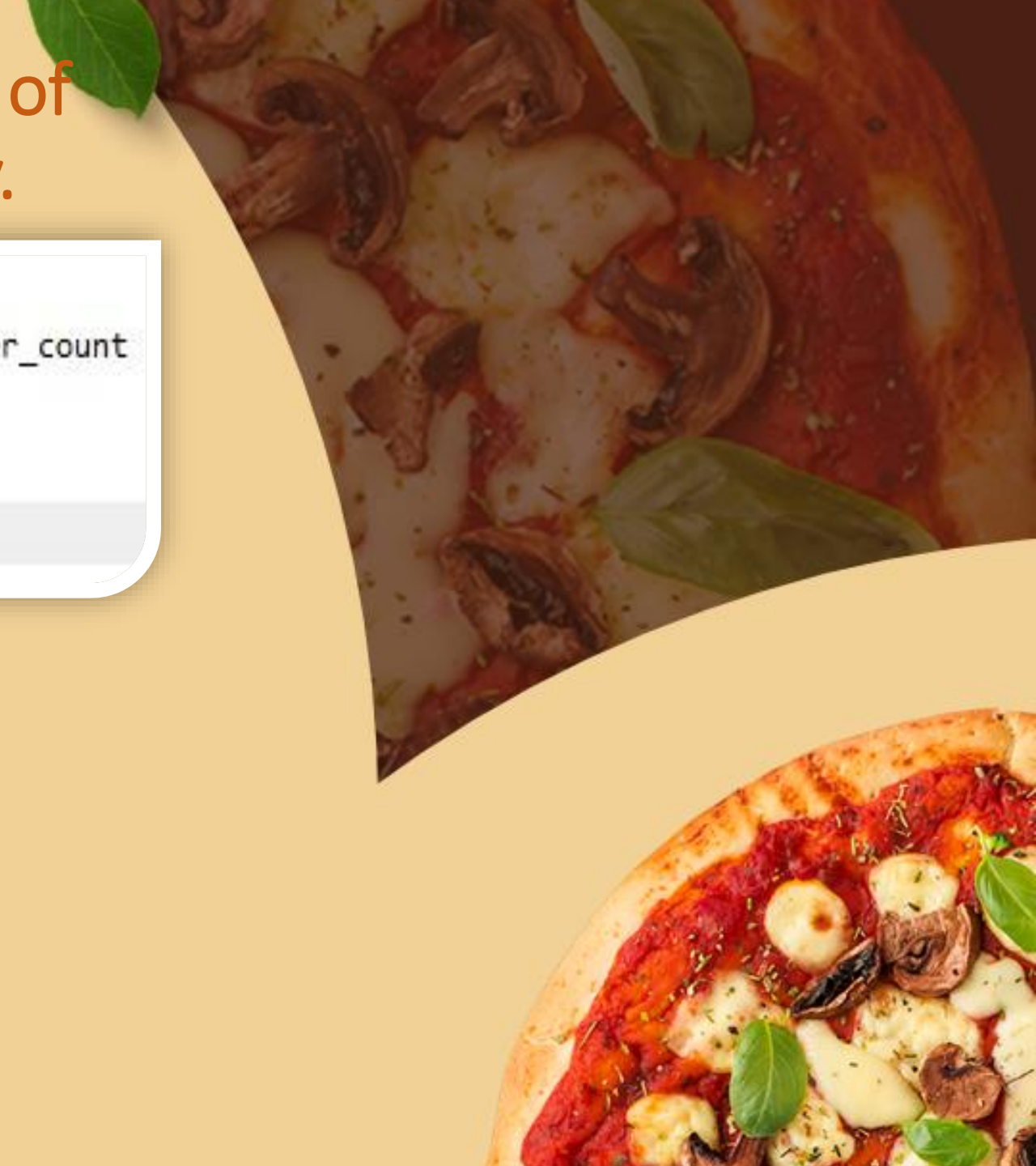


# Determine the distribution of orders by hour of the day.

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

	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	1
10	8
53	58








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






Group the orders by date and calculate the Join relevant tables to find the category-wise distribution of pizzas.

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

	ROUND(AVG(quantity), 0)
▶	138





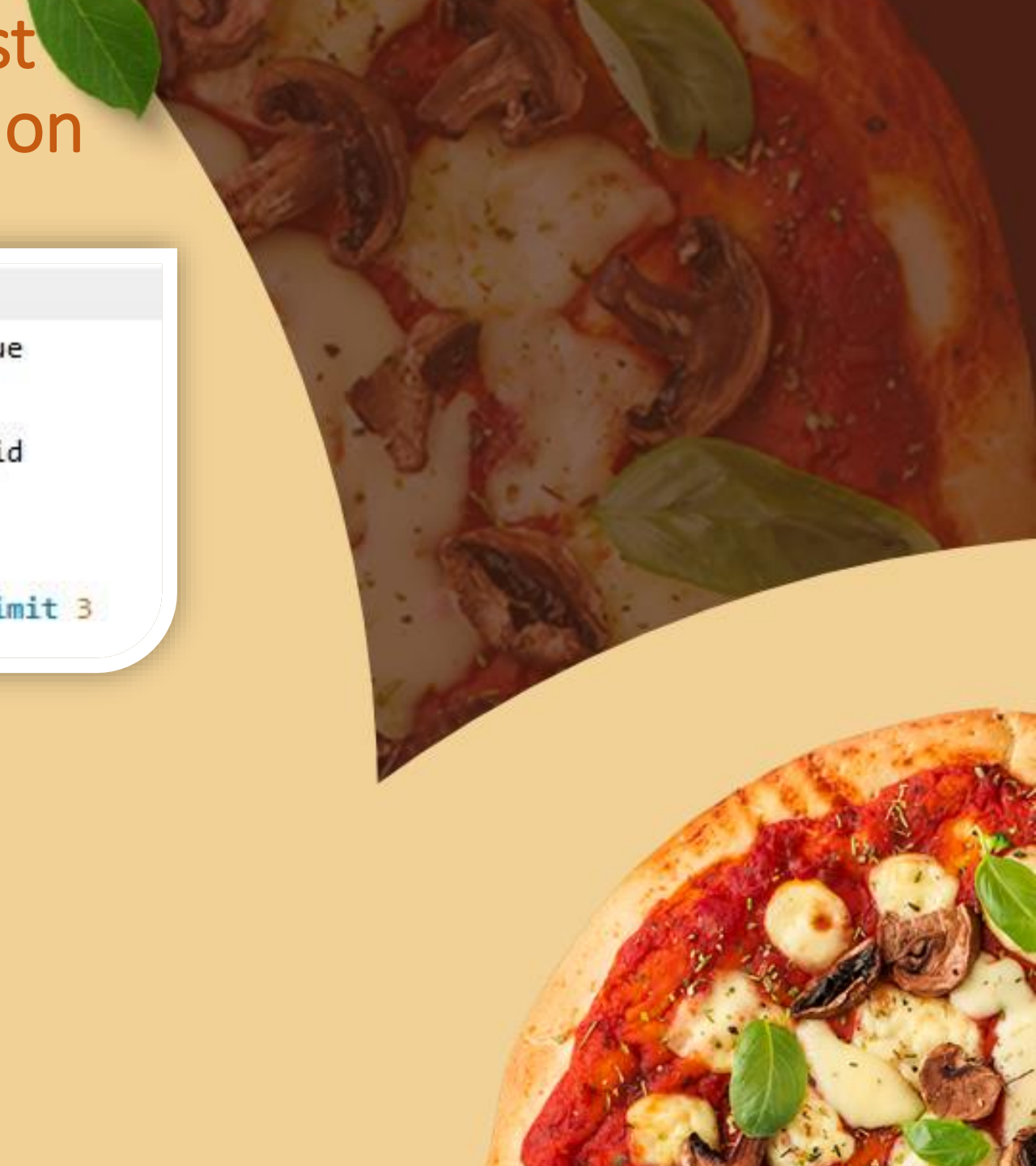


# Determine the top 3 most ordered pizza types based on revenue.

```
select pizza_types.name,  
sum(order_details.quantity*pizzas.price) as revenue  
from pizza_types join pizzas  
on pizzas.pizza_type_id = pizza_types.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

The California Chicken Pizza 41409.5



# 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(od.quantity * ps.price), 2) AS total_sales  
FROM  
  order_details AS od  
  JOIN  
  pizzas AS ps ON ps.pizza_id = od.pizza_id)*100,2) as revenue  
from pizza_types join pizzas  
on pizzas.pizza_type_id = pizza_types.pizza_type_id  
join order_details  
on order_details.pizza_id = pizzas.pizza_id  
group by pizza_types.category order by revenue desc;
```


	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68



# Analyze the cumulative revenue generated over time.

```
select date as order_date, sum(revenue) over(order by date) as total_revenue
from
(select orders.date,
round(sum(order_details.quantity*pizzas.price),2) 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.date) as t
```

order_date	total_revenue
2015-01-01	2713.85
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	21526.399999999998
2015-01-10	23990.35
2015-01-11	25862.649999999998
2015-01-12	27781.699999999997
2012-01-15	51181.000000000001
2012-01-11	52805.000000000001



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(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 t) as t2
where rn<=3;
```

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







## Conclusion:

This SQL project not only functions as a strong data management system but also acts as a strategic tool for business intelligence. By storing detailed data on every aspect of store operations, the database enables managers to make precise adjustments that enhance both customer experience and profitability. When shared on a blog, this project can offer practical insights into how structured SQL queries can be leveraged to utilize data for real-world business applications, making it a valuable resource for aspiring data analysts and business owners.