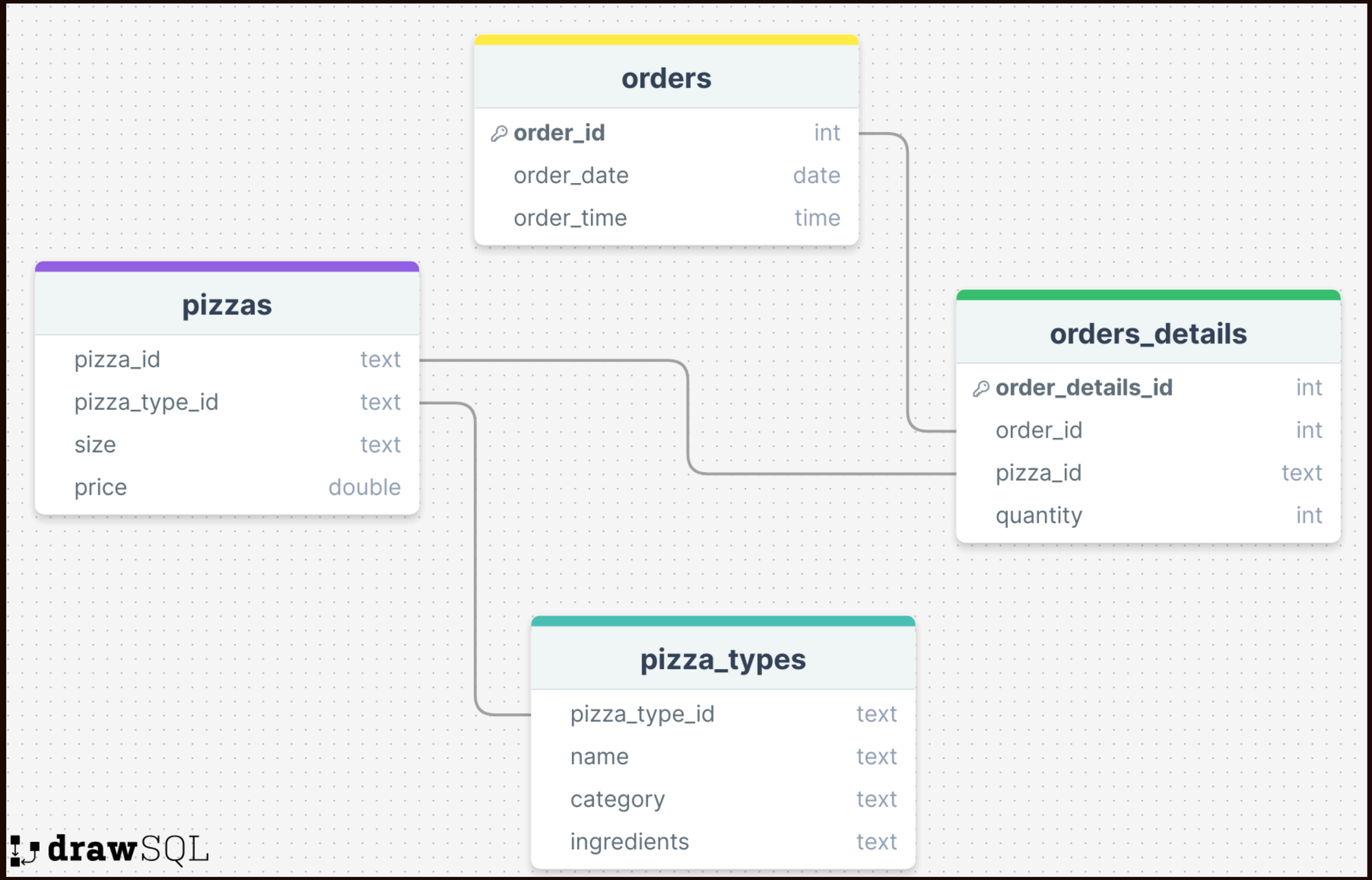
SQL PROJECT

PIZZA SALES ANALYSIS

SCHEMA


orders
order_id int
order_date date
order_time time
pizzas
pizza_id text
pizza_type_id text
size text
price double
orders_details
order_details_id int
order_id int
pizza_id text
quantity int
pizza_types
pizza_type_id text
name text
category text
ingredients text
drawSQL

EASY SQL QUERIES

1. Retrieve the total number of orders placed.

```
SELECT count(order_id) as total_orders FROM orders
```

Result Grid

	total_orders
▶	21350

2. Calculate the total revenue generated from pizza sales.

```sql
SELECT round(sum(od.quantity * p.price),2)as total_revenue FROM pizzas p
INNER JOIN orders_details od ON p.pizza_id = od.pizza_id;
```

Result Grid

total_revenue
817860.05

3. Identify the highest-priced pizza.

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

Result Grid | Filter Rows:

name	price
The Greek Pizza	35.95

4. Identify the most common pizza size ordered.

```sql
SELECT p.size, count(od.order_details_id) as order_count FROM orders_details od
INNER JOIN pizzas p ON od.pizza_id = p.pizza_id
GROUP BY p.size
ORDER BY order_count DESC LIMIT 1;
```

Result Grid | Filter Rows:

	size	order_count
▶	L	18526

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

```sql
SELECT name, count(quantity) as total_quantity FROM pizzas p
INNER JOIN pizza_types pt  ON p.pizza_type_id = pt.pizza_type_id
INNER JOIN orders_details od ON p.pizza_id = od.pizza_id
GROUP BY name
ORDER BY total_quantity DESC LIMIT 5;
```

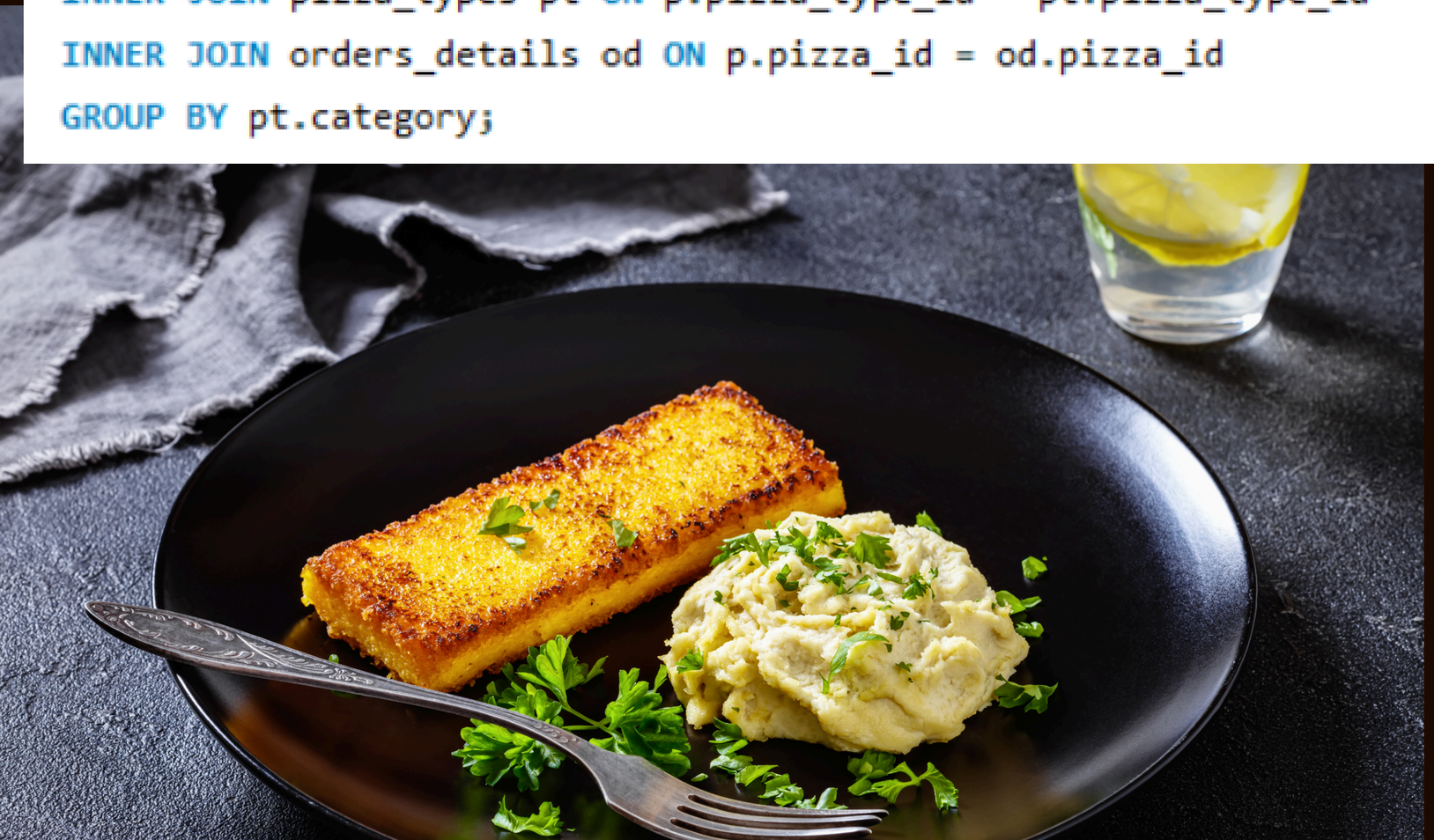
Result Grid | Filter Rows:

name	total_quantity
The Classic Deluxe Pizza	2416
The Barbecue Chicken Pizza	2372
The Hawaiian Pizza	2370
The Pepperoni Pizza	2369
The Thai Chicken Pizza	2315

INTERMEDIATE SQL QUERIES

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

```sql
SELECT pt.category, count(od.quantity) FROM pizzas p
INNER JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
INNER JOIN orders_details od ON p.pizza_id = od.pizza_id
GROUP BY pt.category;
```



Result Grid | Filter Rows:

category	count(od.quantity)
Classic	14579
Veggie	11449
Supreme	11777
Chicken	10815

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

```
SELECT  hour(order_time )as hours, count(order_id) as distribution FROM orders
GROUP BY hours;
```

Result Grid | Filter

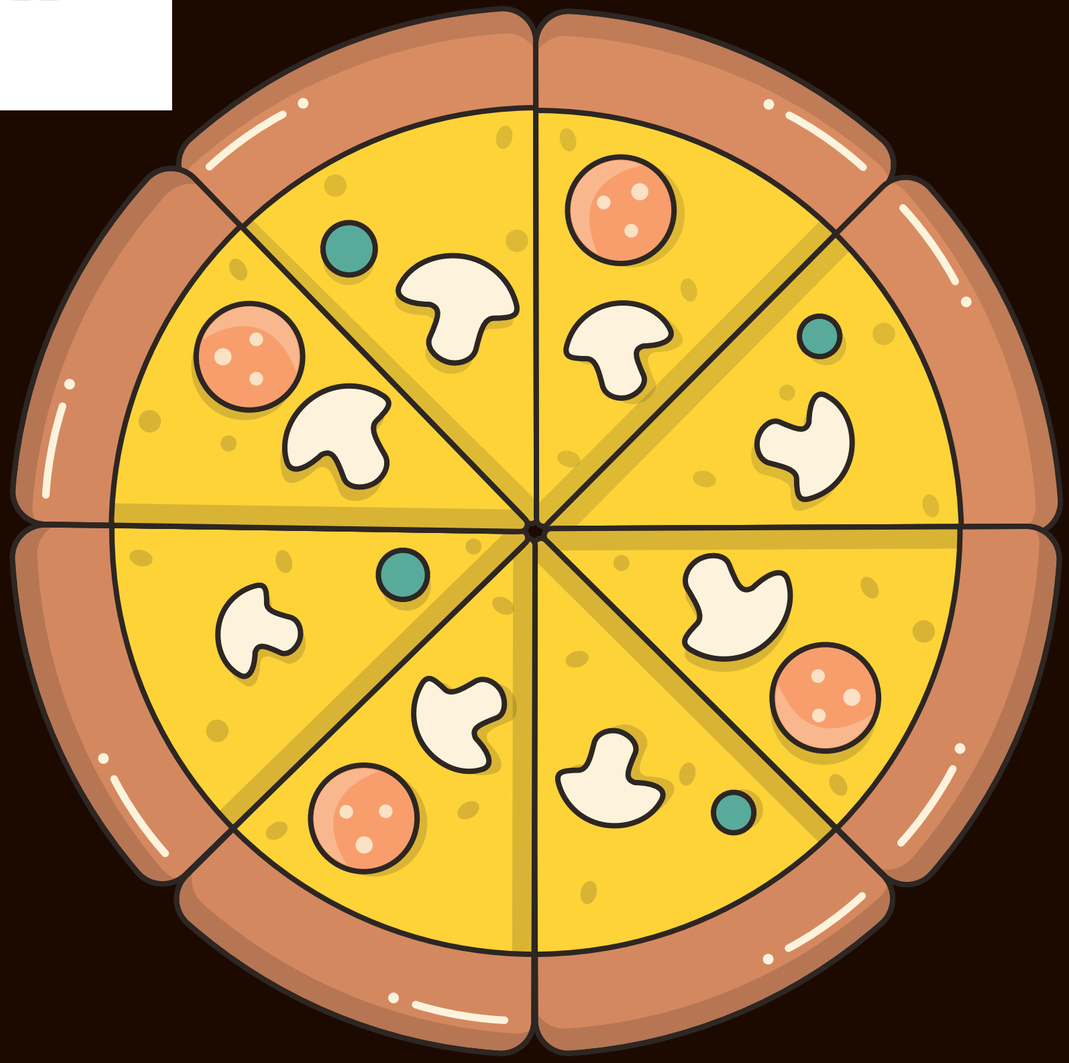
	hours	distribution
▶	11	1231
	12	2520
	13	2455
	14	1472
	15	1468
	16	1920
	17	2336
	18	2399
	19	2009
	20	1642
	21	1198
	22	663
	23	28
	10	8
	9	1

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

```
SELECT category, count(category) FROM pizza_types
GROUP BY category;
```

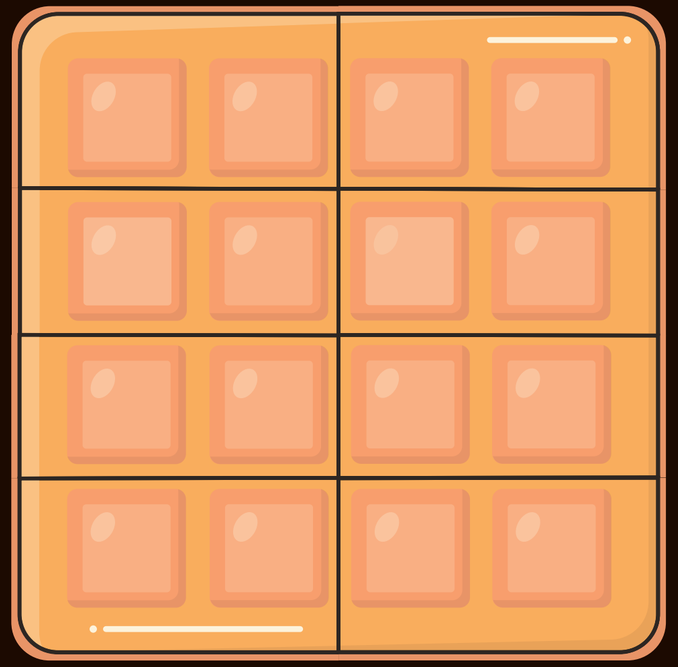
Result Grid | Filter Rows:

category	count(category)
Chicken	6
Classic	8
Supreme	9
Veggie	9



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

```sql
SELECT round(avg(order_quantity),0) as avg_no_of_pizzas_ordered_per_day FROM
(SELECT o.order_date, sum(od.quantity) as order_quantity FROM orders o
INNER JOIN orders_details od on o.order_id = od.order_id
GROUP BY order_date) as order_quantity;
```



Result Grid | Filter Rows:

	avg_no_of_pizzas_ordered_per_day
▶	138

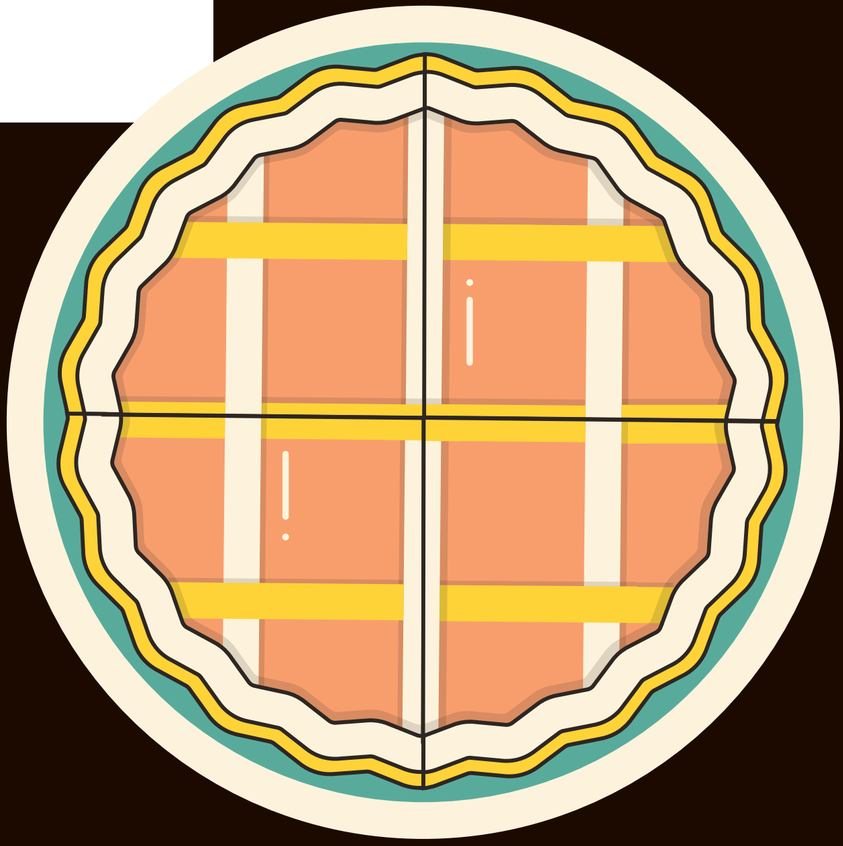
ADVANCE SQL QUERIES

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

```sql
SELECT pt.name , round(sum(p.price * od.quantity),2) as revenue FROM pizzas p
INNER JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
INNER JOIN orders_details od ON p.pizza_id = od.pizza_id
GROUP BY pt.name
ORDER BY revenue DESC LIMIT 3;
```

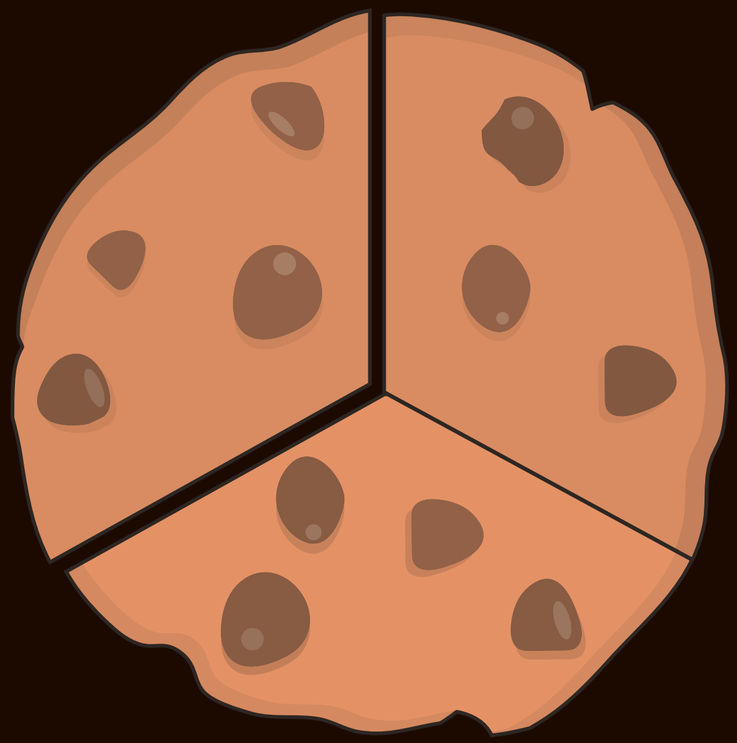
Result Grid | Filter Rows:

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



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

```sql
SELECT pt.category , round((sum(p.price * od.quantity) /(SELECT sum(o.quantity * p.price) as total_revenue FROM pizzas p
INNER JOIN orders_details o ON p.pizza_id = o.pizza_id))*100,2) as revenue FROM pizzas p
INNER JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
INNER JOIN orders_details od ON p.pizza_id = od.pizza_id
GROUP BY pt.category;
```

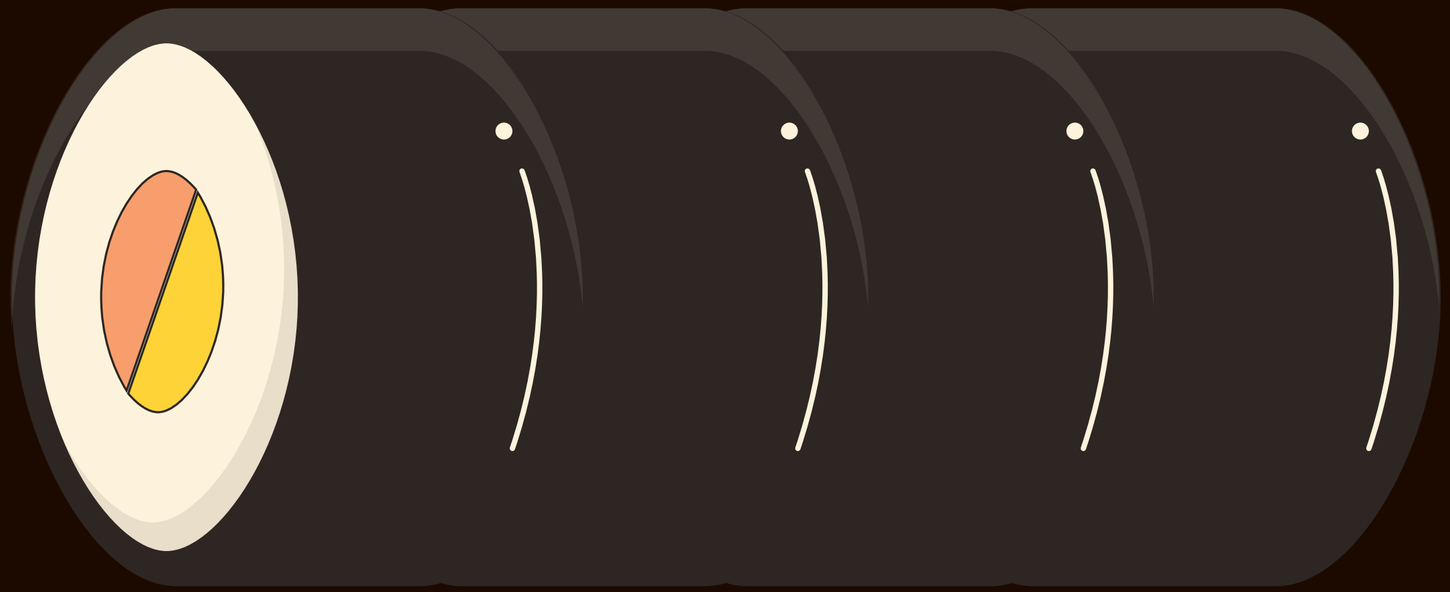


Result Grid | Filter

category	revenue
Classic	26.91
Veggie	23.68
Supreme	25.46
Chicken	23.96

12. Analyze the cumulative revenue generated over time.

```sql
SELECT order_date, round(sum(revenue) OVER(ORDER BY order_date),2) as cumulative_sales FROM
(SELECT  o.order_date , round(sum(p.price * od.quantity),2) as revenue FROM orders_details od
INNER JOIN orders o  ON od.order_id = o.order_id
INNER JOIN pizzas p ON od.pizza_id = p.pizza_id
GROUP BY o.order_date) as sales;
```



order_date	cumulative_sales
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	21526.4
2015-01-10	23990.35
2015-01-11	25862.65
2015-01-12	27781.7
2015-01-13	29831.3
2015-01-14	32358.7
2015-01-15	34343.5
2015-01-16	36937.65
2015-01-17	39001.75
2015-01-18	40978.6
2015-01-19	43365.75
2015-01-20	45763.65
2015-01-21	47804.2
2015-01-22	50300.9
2015-01-23	52724.6
2015-01-24	55013.85

Result 3

PIZZA
RESTAURANT

THANK YOU