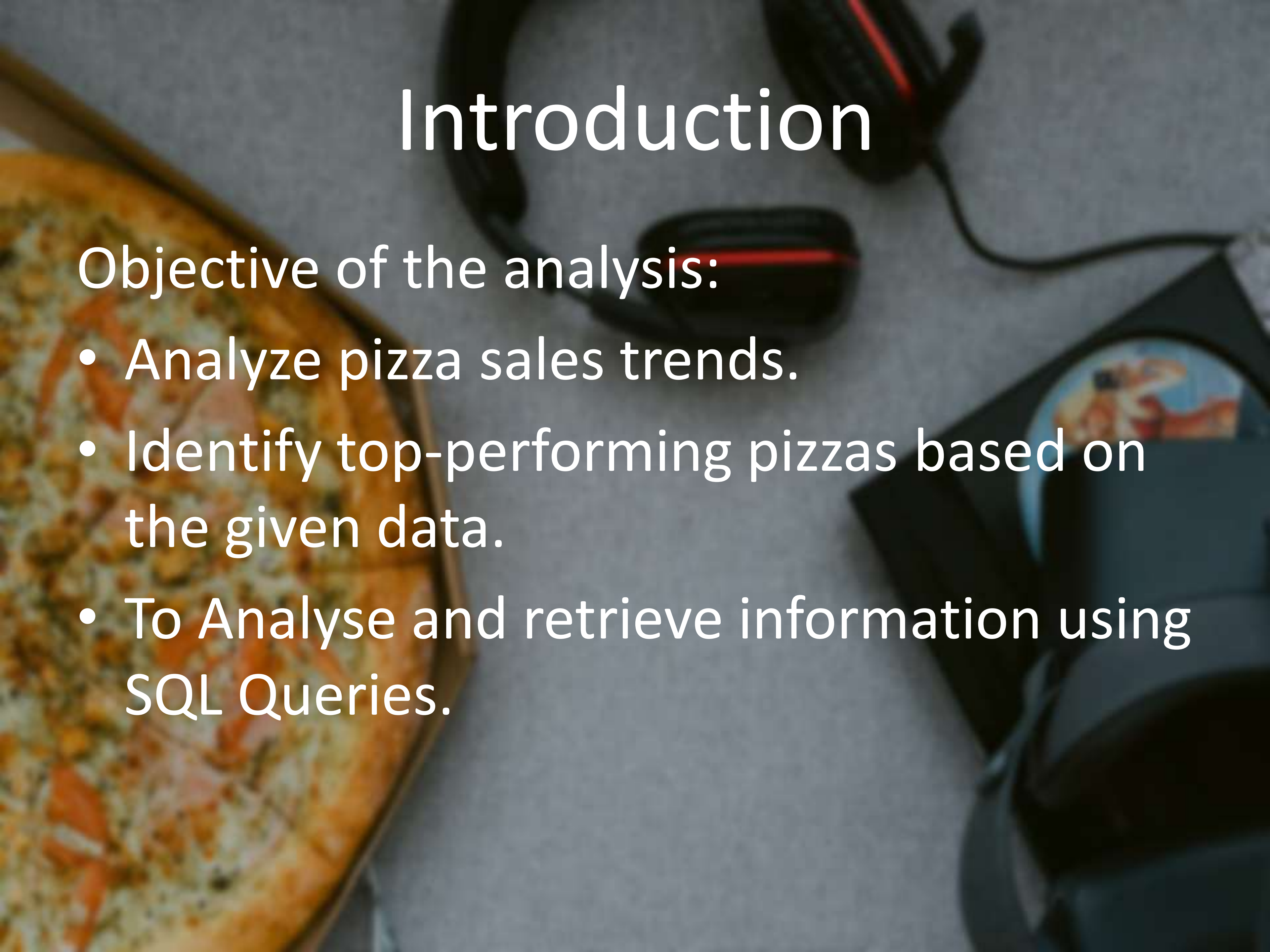
Pizza Sales Analysis

Hello

My name is Hasmath Ali ,

In this project I’ve utilized SQL queries to solve question related to pizza sales Analytics.

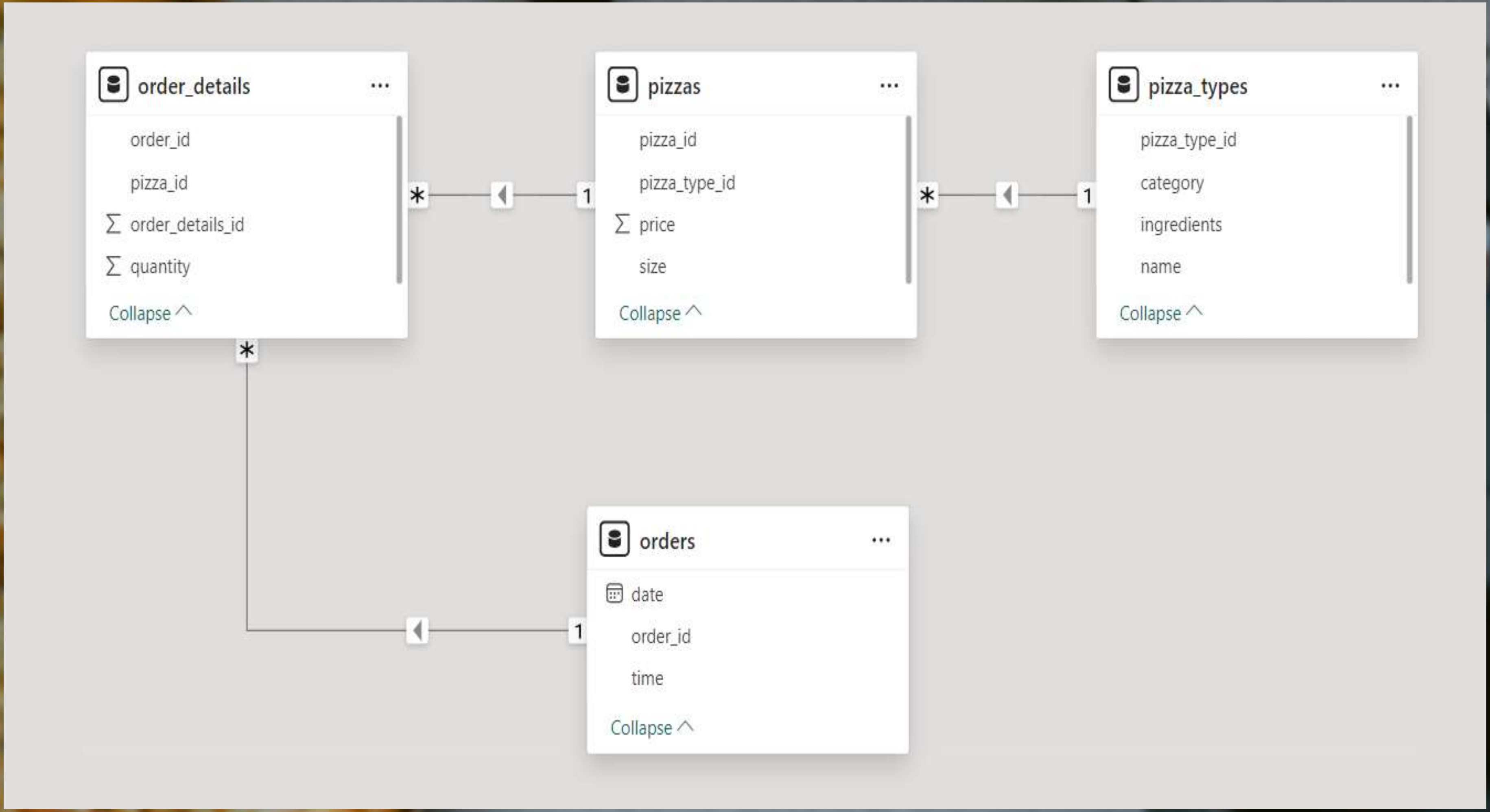


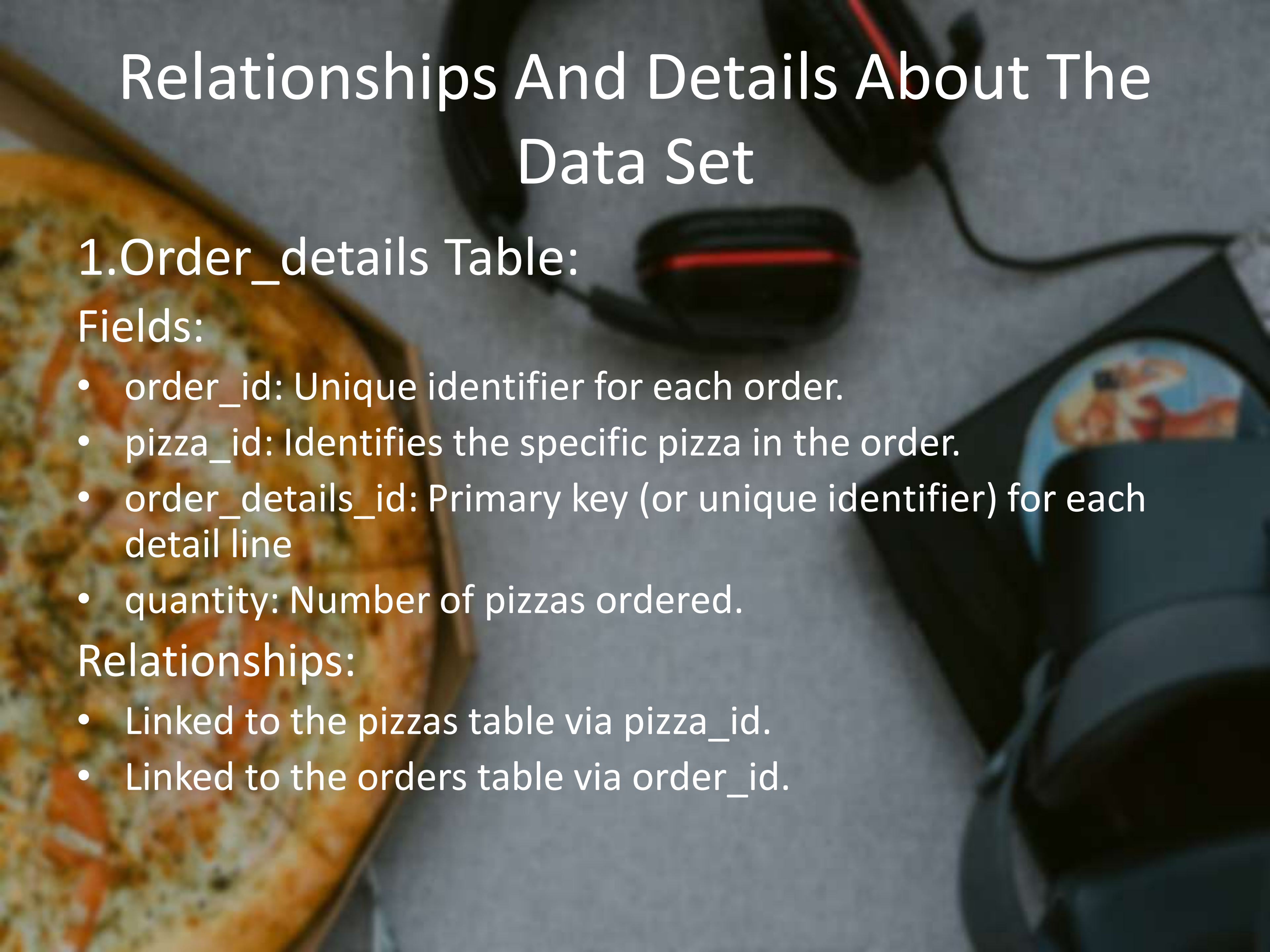
Introduction

Objective of the analysis:

- Analyze pizza sales trends.
- Identify top-performing pizzas based on the given data.
- To Analyse and retrieve information using SQL Queries.

Relationships And Details About The Data Set





Relationships And Details About The Data Set

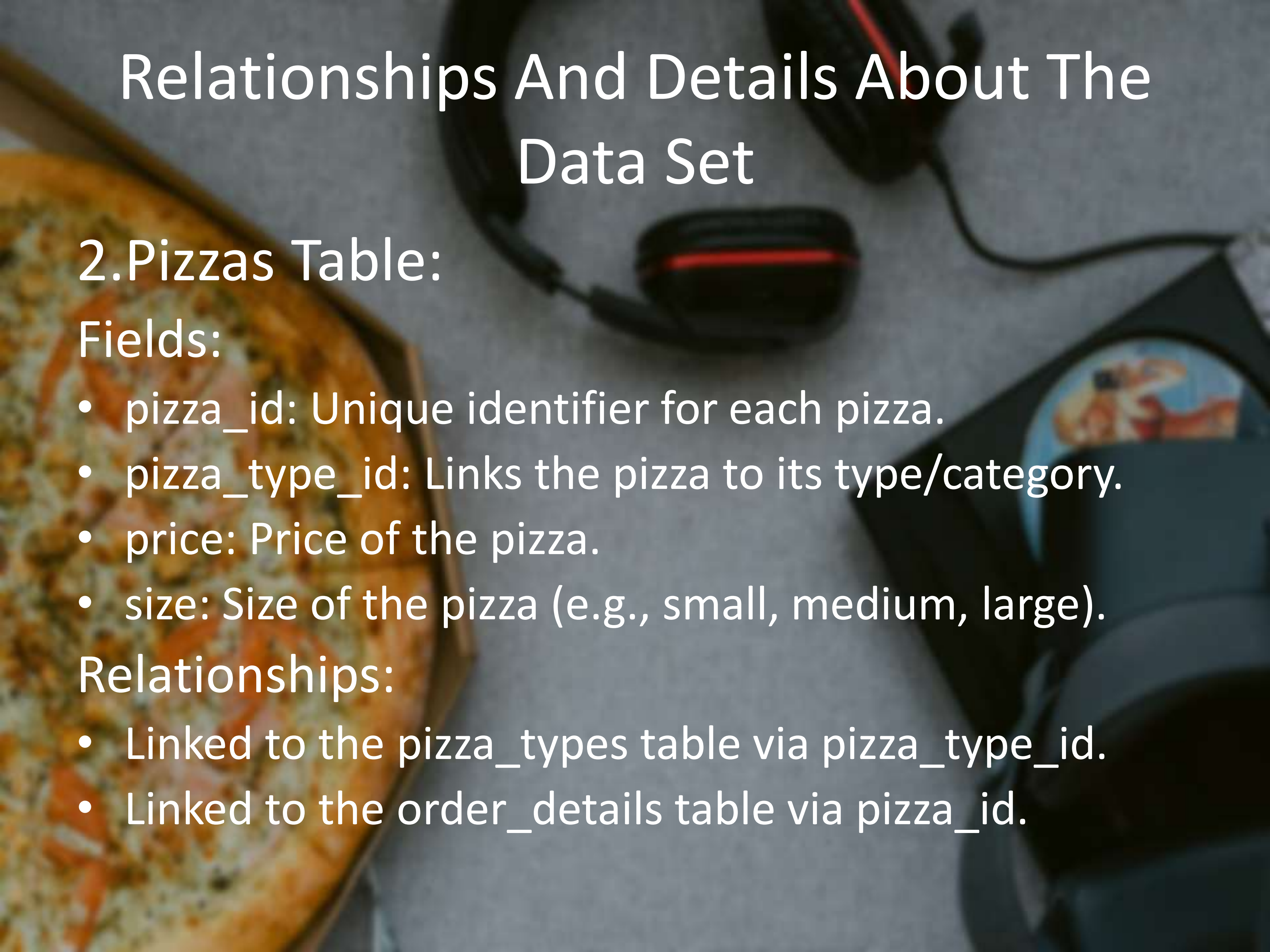
1.Order_details Table:

Fields:

- order_id: Unique identifier for each order.
- pizza_id: Identifies the specific pizza in the order.
- order_details_id: Primary key (or unique identifier) for each detail line
- quantity: Number of pizzas ordered.

Relationships:

- Linked to the pizzas table via pizza_id.
- Linked to the orders table via order_id.



Relationships And Details About The Data Set

2.Pizzas Table:

Fields:

- pizza_id: Unique identifier for each pizza.
- pizza_type_id: Links the pizza to its type/category.
- price: Price of the pizza.
- size: Size of the pizza (e.g., small, medium, large).

Relationships:

- Linked to the pizza_types table via pizza_type_id.
- Linked to the order_details table via pizza_id.

Relationships And Details About The Data Set

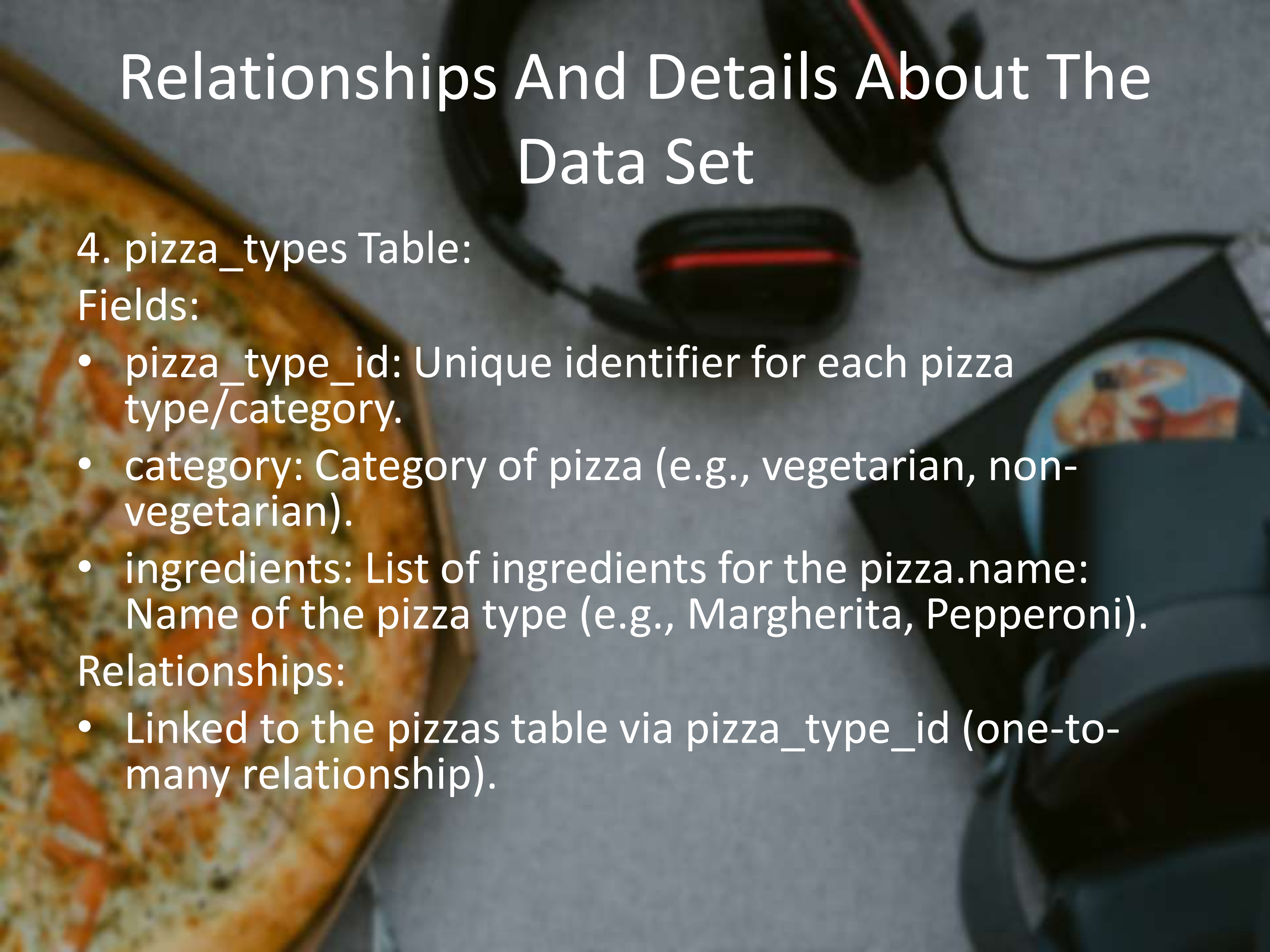
3.orders Table:

Fields:

- date: Date of the order.
- order_id: Unique identifier for each order.time: Time when the order was placed.

Relationships:

- Linked to the order_details table via order_id (one-to-many relationship)



Relationships And Details About The Data Set

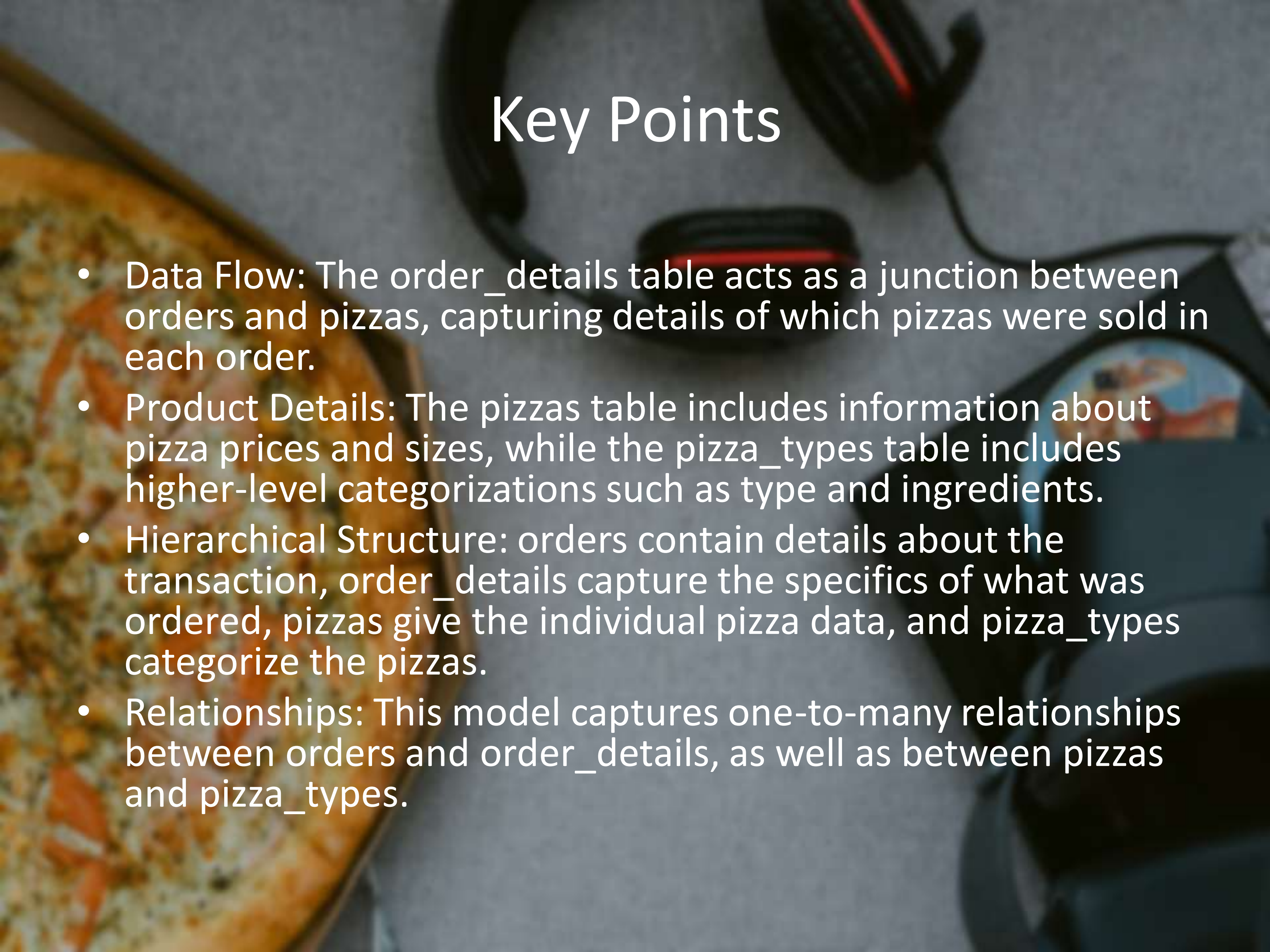
4. pizza_types Table:

Fields:

- pizza_type_id: Unique identifier for each pizza type/category.
- category: Category of pizza (e.g., vegetarian, non-vegetarian).
- ingredients: List of ingredients for the pizza.name: Name of the pizza type (e.g., Margherita, Pepperoni).

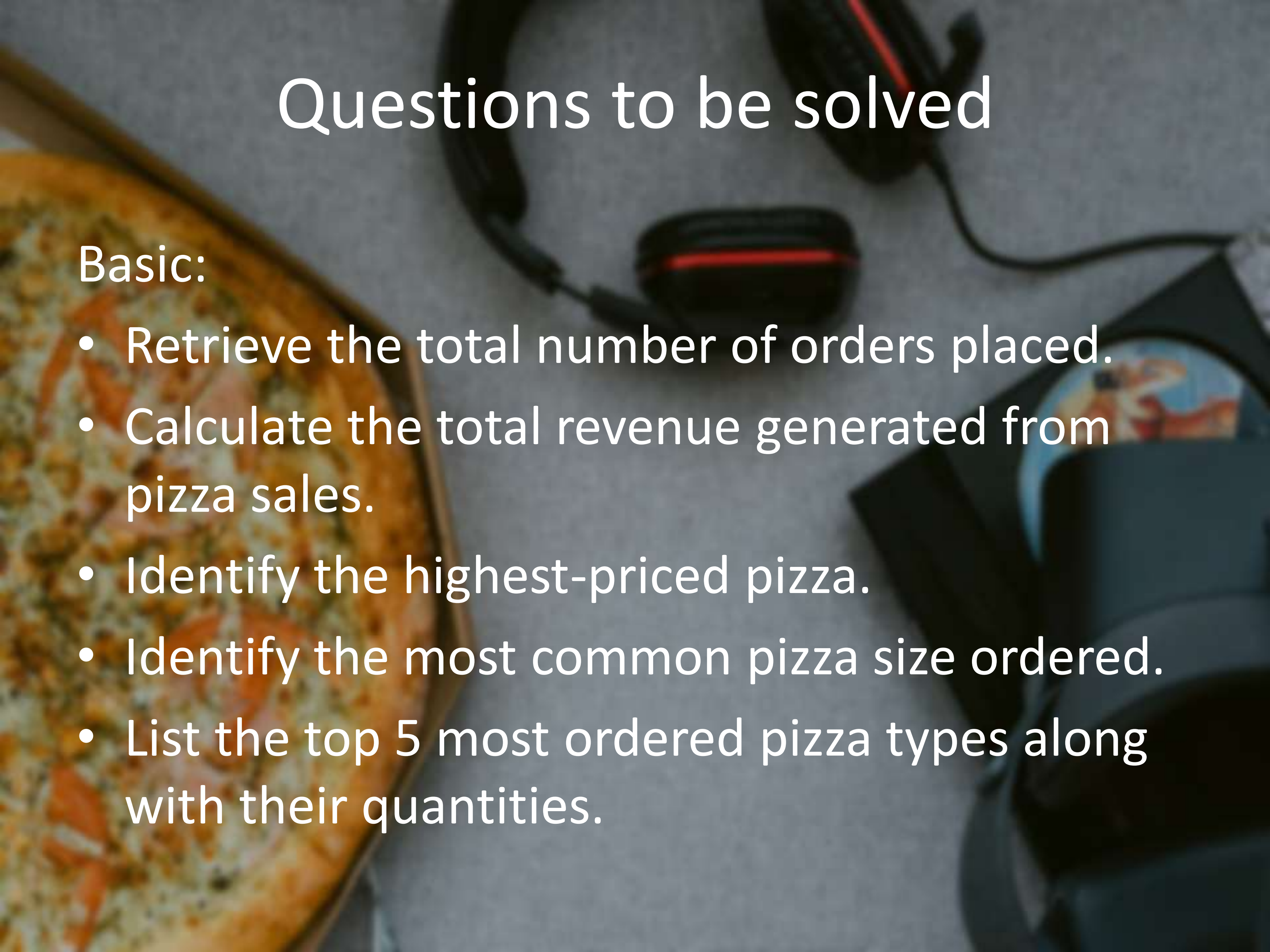
Relationships:

- Linked to the pizzas table via pizza_type_id (one-to-many relationship).



Key Points

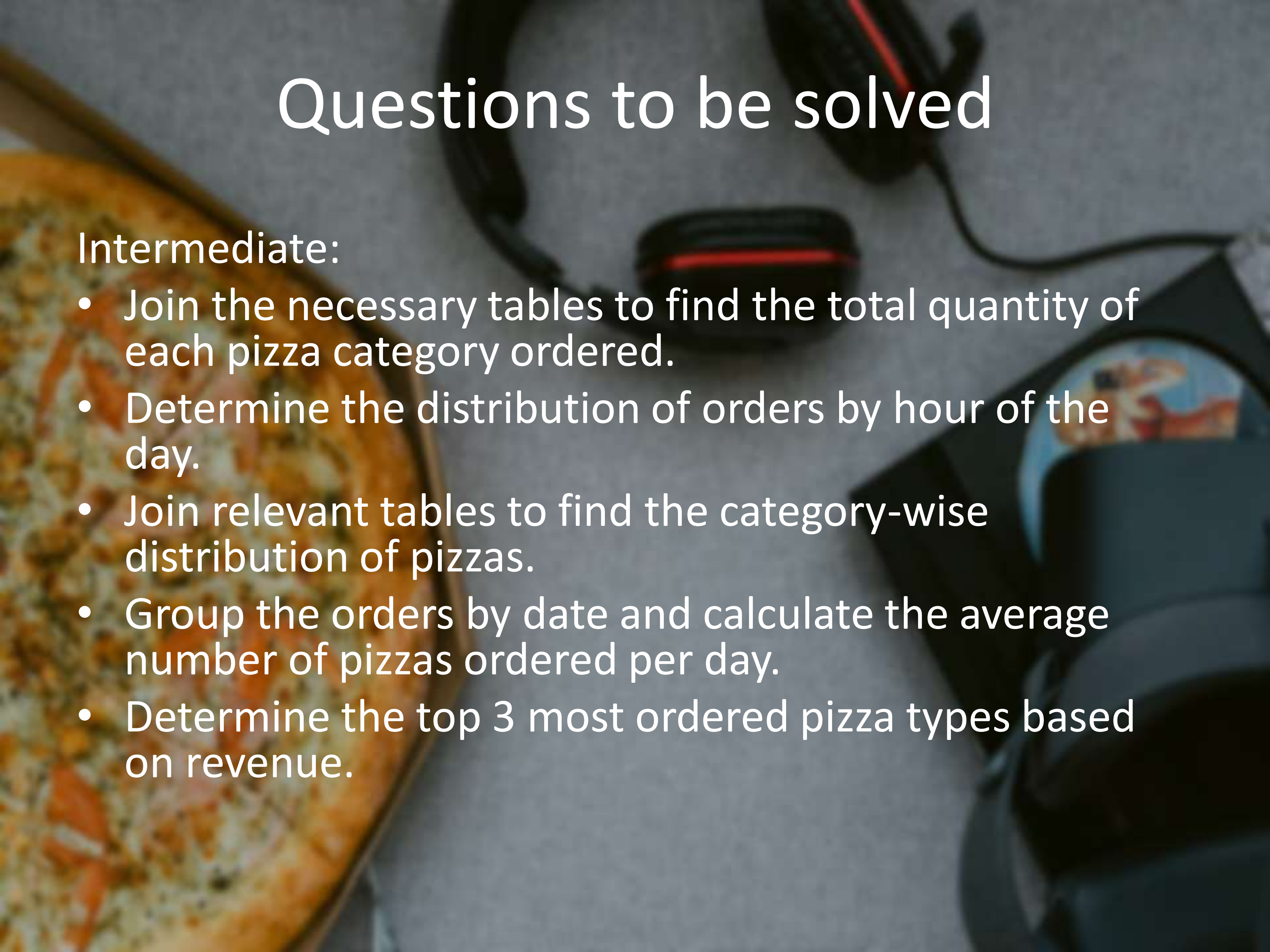
- Data Flow: The order_details table acts as a junction between orders and pizzas, capturing details of which pizzas were sold in each order.
- Product Details: The pizzas table includes information about pizza prices and sizes, while the pizza_types table includes higher-level categorizations such as type and ingredients.
- Hierarchical Structure: orders contain details about the transaction, order_details capture the specifics of what was ordered, pizzas give the individual pizza data, and pizza_types categorize the pizzas.
- Relationships: This model captures one-to-many relationships between orders and order_details, as well as between pizzas and pizza_types.



Questions to be solved

Basic:

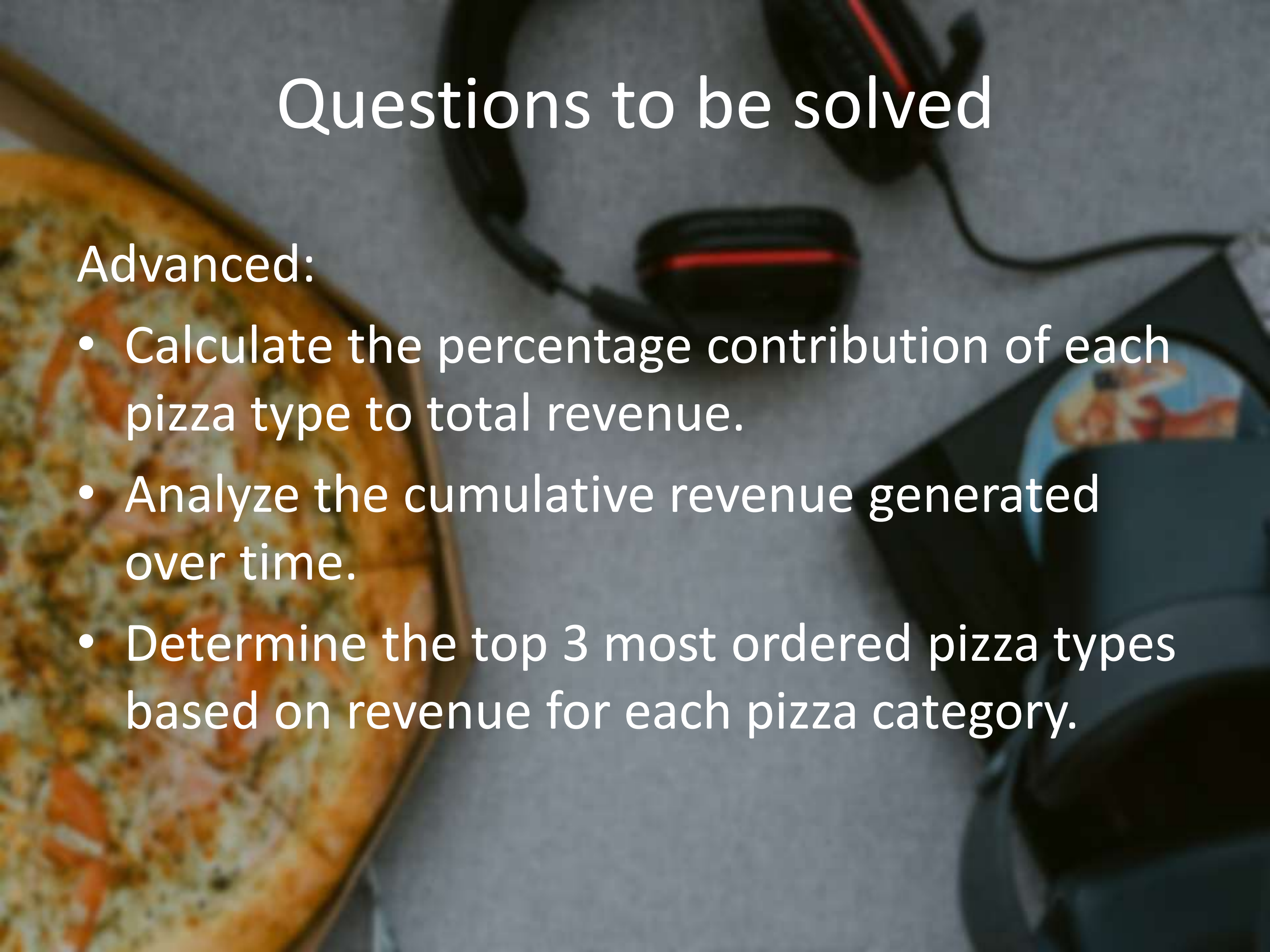
- Retrieve the total number of orders placed.
- Calculate the total revenue generated from pizza sales.
- Identify the highest-priced pizza.
- Identify the most common pizza size ordered.
- List the top 5 most ordered pizza types along with their quantities.



Questions to be solved

Intermediate:

- Join the necessary tables to find the total quantity of each pizza category ordered.
- Determine the distribution of orders by hour of the day.
- Join relevant tables to find the category-wise distribution of pizzas.
- Group the orders by date and calculate the average number of pizzas ordered per day.
- Determine the top 3 most ordered pizza types based on revenue.

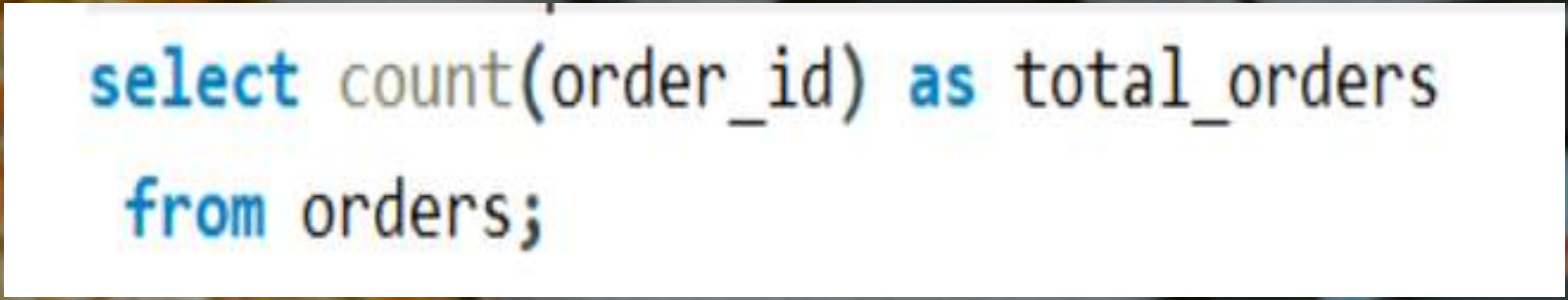


Questions to be solved

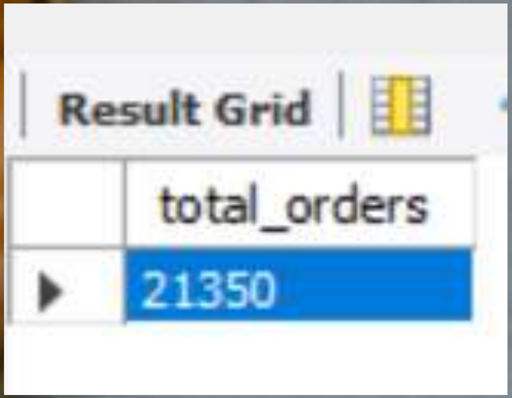
Advanced:

- Calculate the percentage contribution of each pizza type to total revenue.
- Analyze the cumulative revenue generated over time.
- Determine the top 3 most ordered pizza types based on revenue for each pizza category.

1.Retrieve the total number of orders placed.



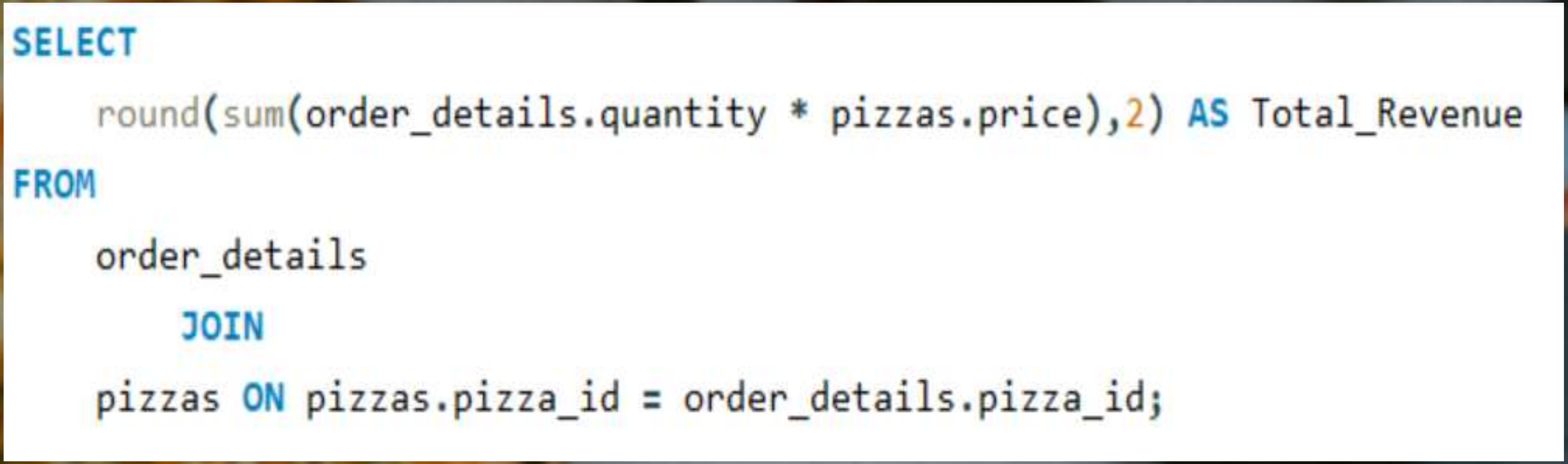
```
select count(order_id) as total_orders
 from orders;
```



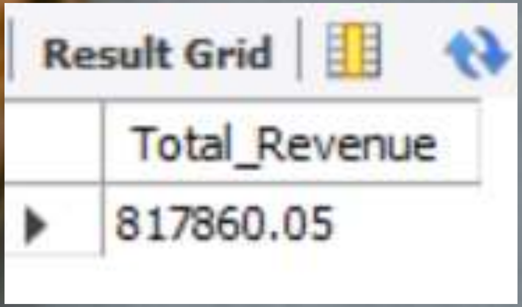
Result Grid

```
total_orders
21350
```

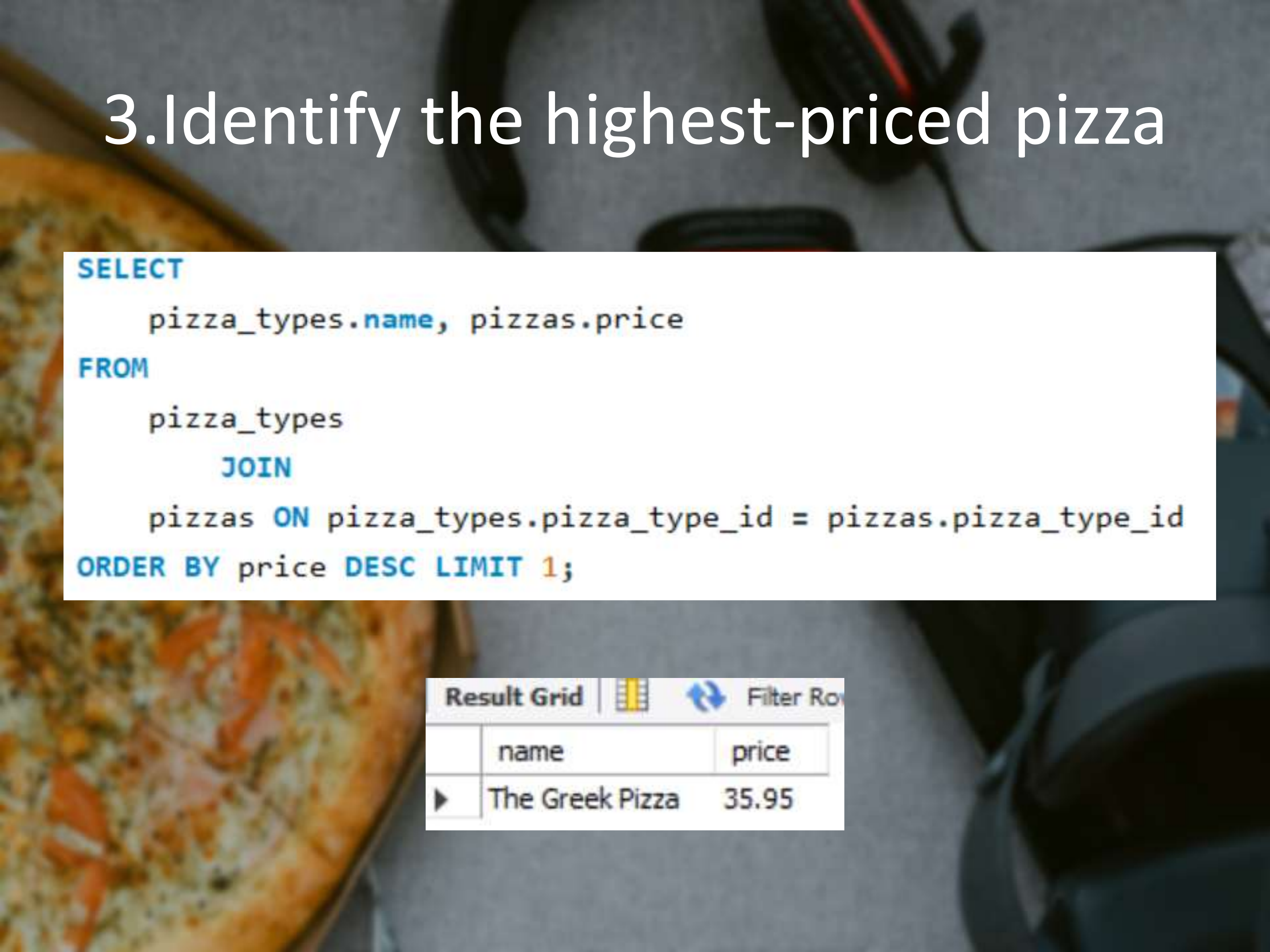
2.Calculate the total revenue generated from pizza sales



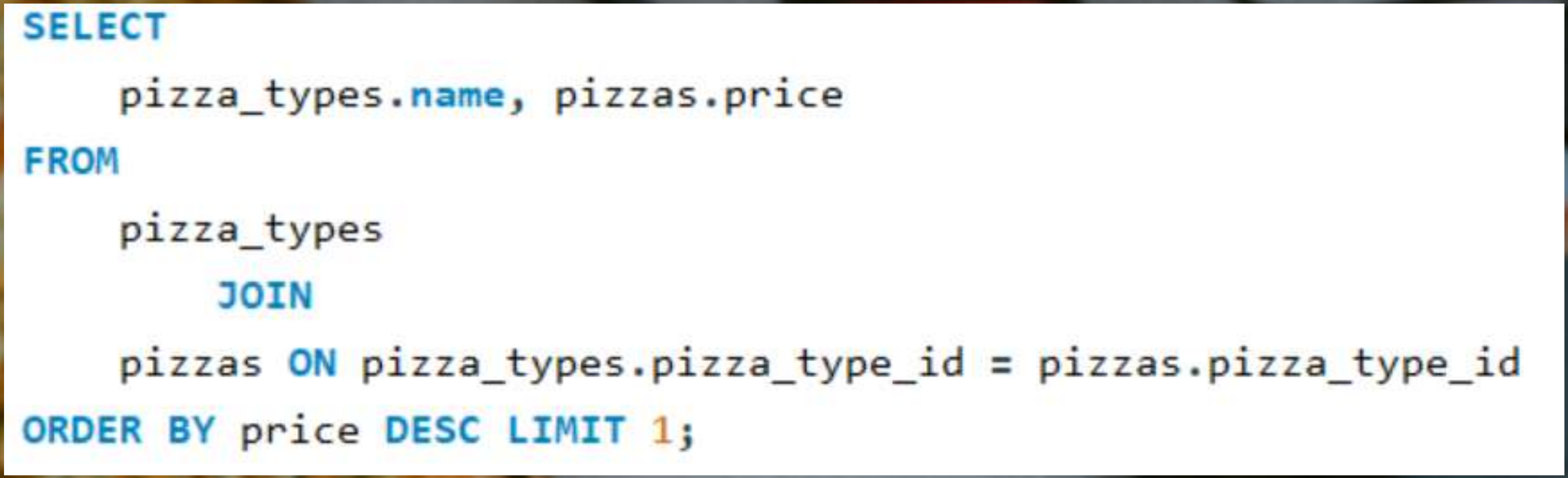
```
SELECT
    round(sum(order_details.quantity * pizzas.price),2) AS Total_Revenue
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id;
```



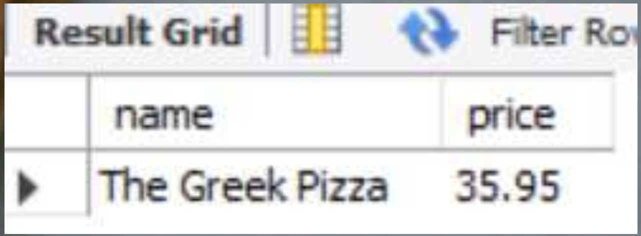
```
Total_Revenue
817860.05
```



3.Identify the highest-priced pizza

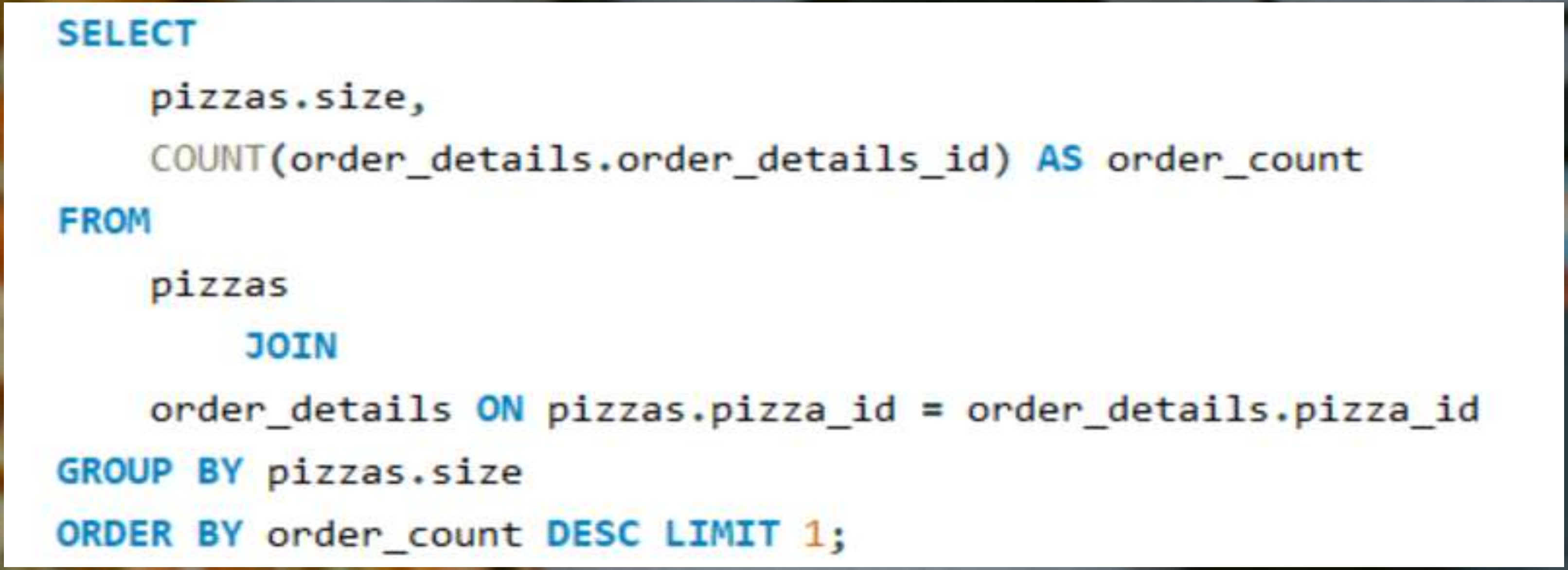


```
SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY price DESC LIMIT 1;
```

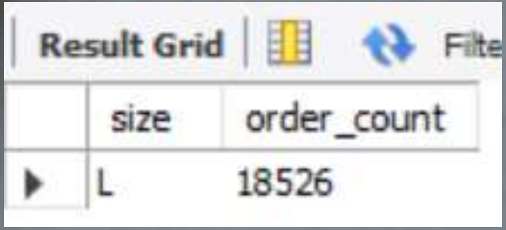


```
  name              price
▶ The Greek Pizza   35.95
```

4.Identify the most common pizza size ordered.

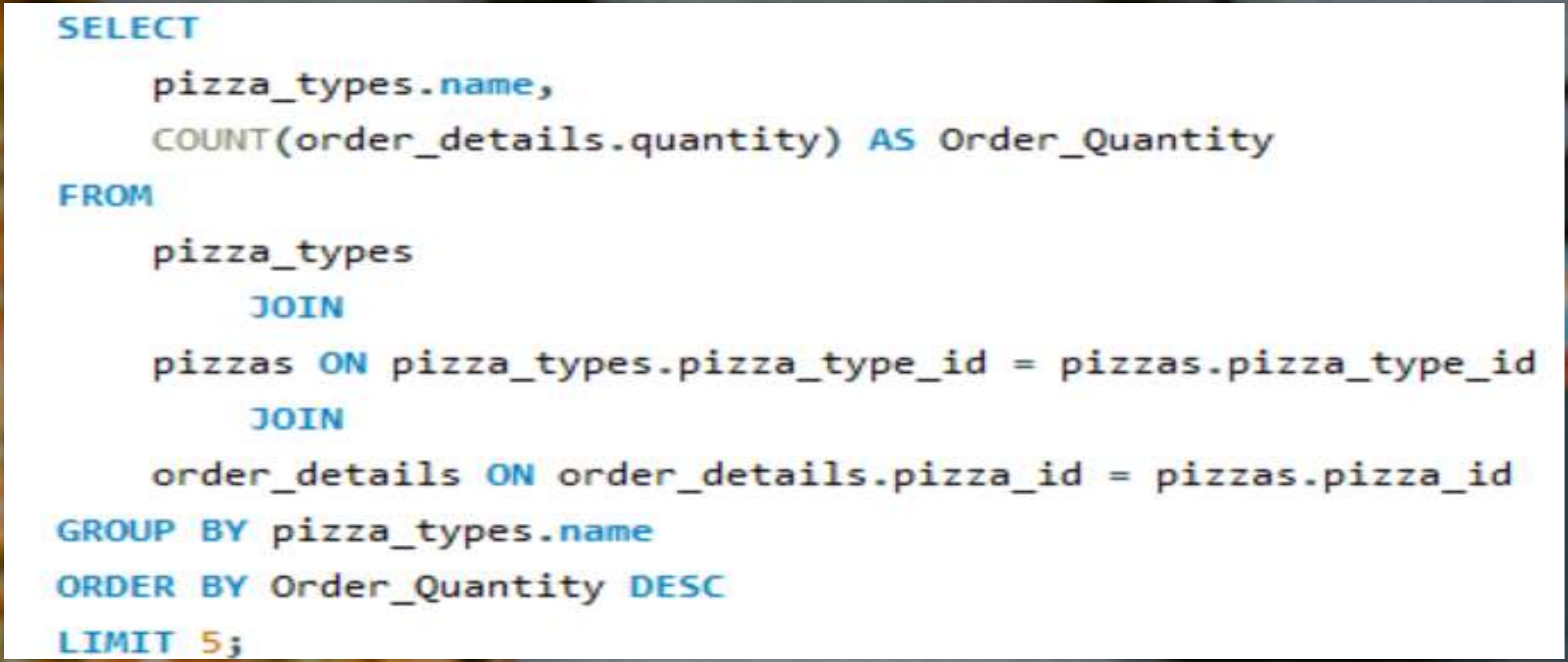


```
SELECT
    pizzas.size,
    COUNT(order_details.order_details_id) AS order_count
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC LIMIT 1;
```

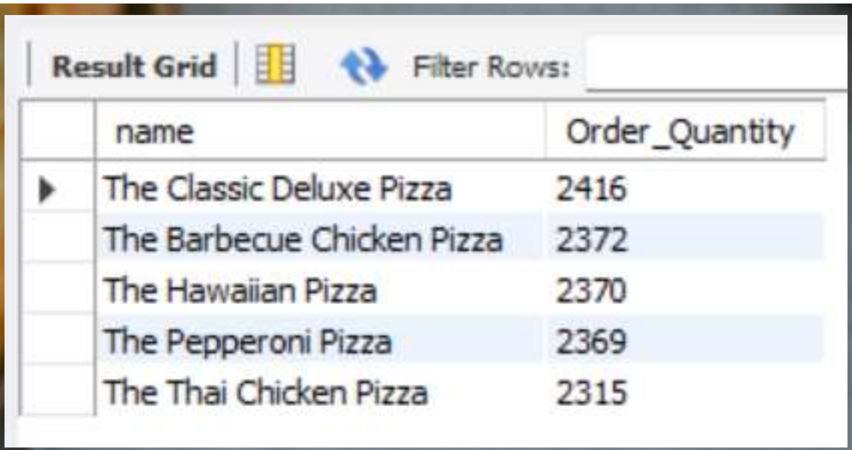


	size	order_count
▶	L	18526

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

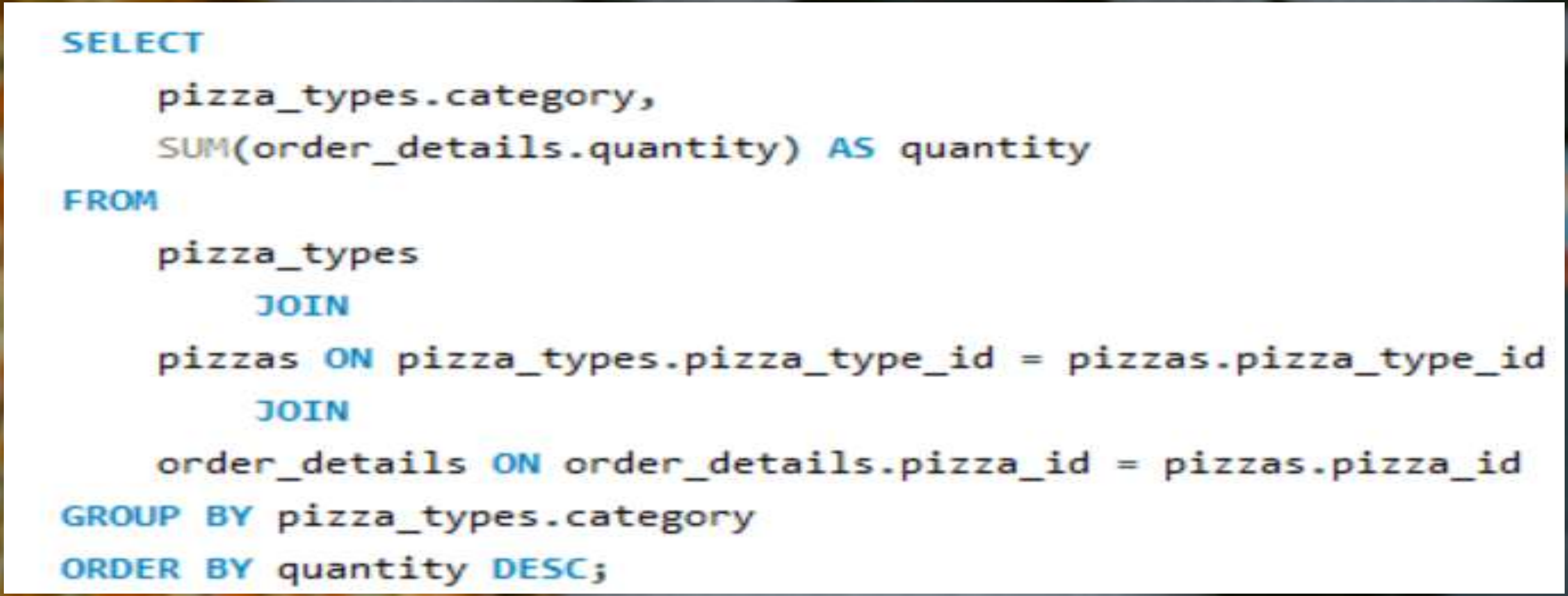


```sql
SELECT
    pizza_types.name,
    COUNT(order_details.quantity) AS Order_Quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY Order_Quantity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

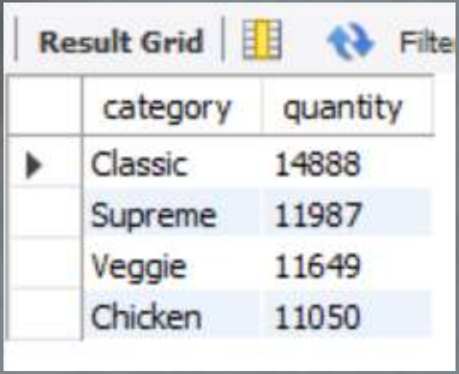


name	Order_Quantity
The Classic Deluxe Pizza	2416
The Barbecue Chicken Pizza	2372
The Hawaiian Pizza	2370
The Pepperoni Pizza	2369
The Thai Chicken Pizza	2315

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

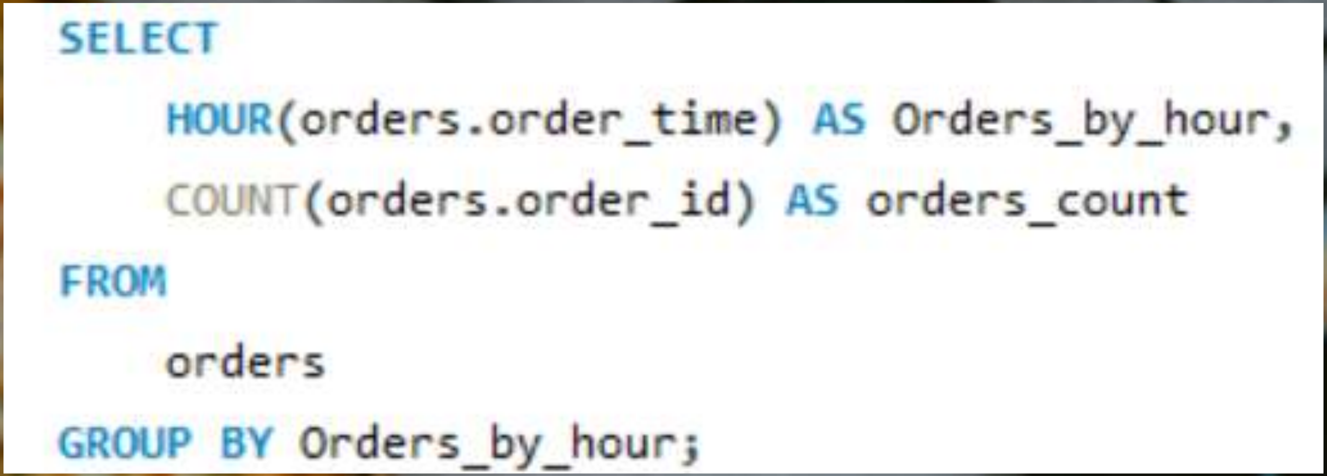


```
SELECT
    pizza_types.category,
    SUM(order_details.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity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

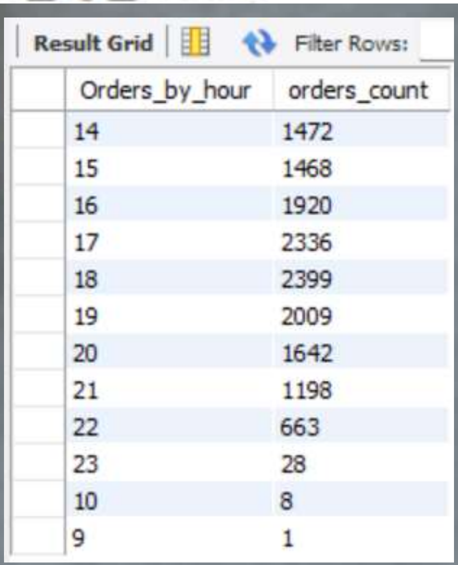


category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

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

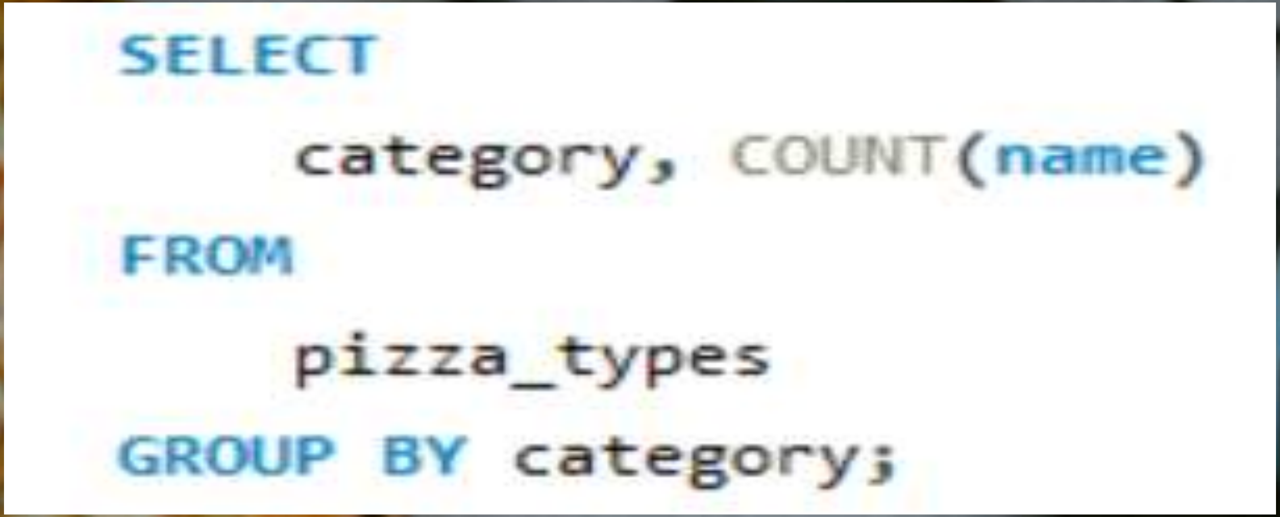


```
SELECT
    HOUR(orders.order_time) AS Orders_by_hour,
    COUNT(orders.order_id) AS orders_count
FROM
    orders
GROUP BY Orders_by_hour;
```

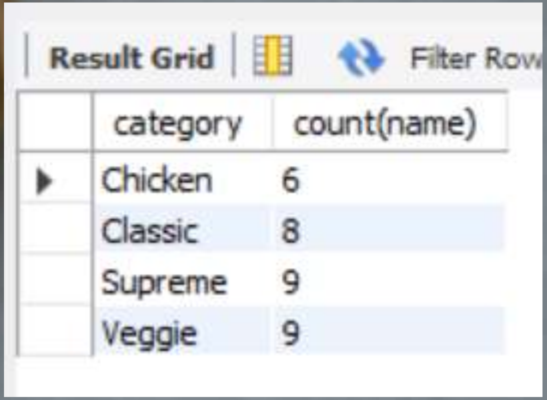


Orders_by_hour	orders_count
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009
20	1642
21	1198
22	663
23	28
10	8
9	1

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

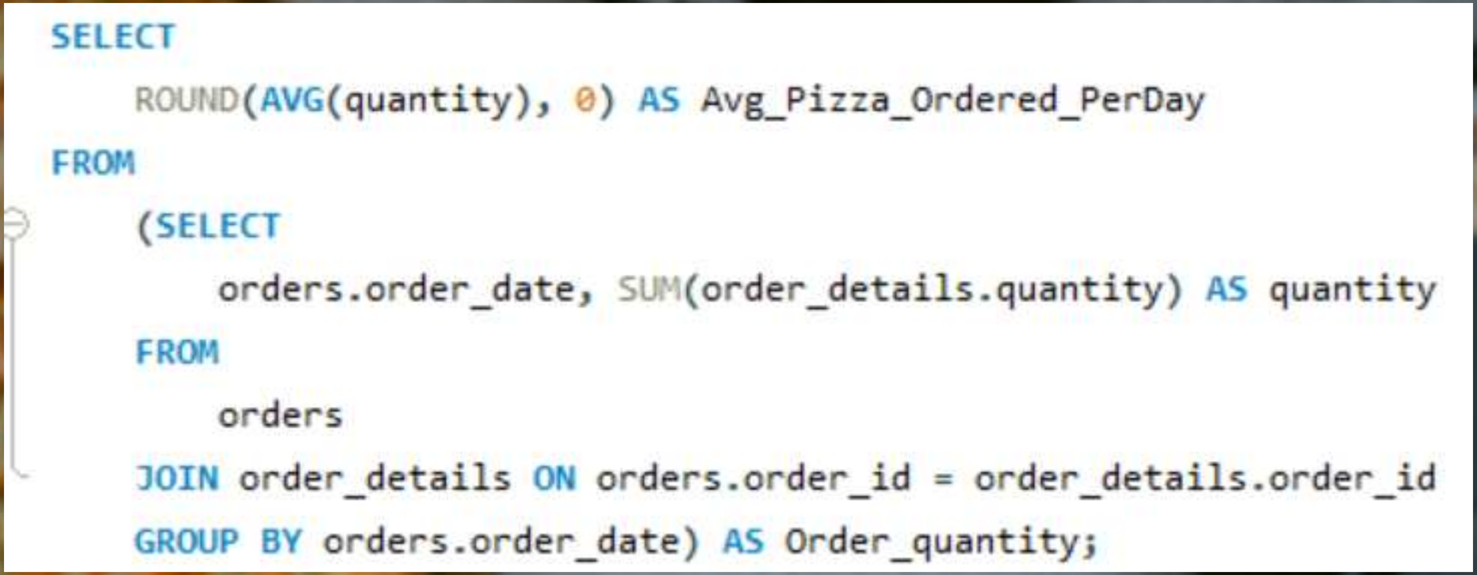


```
SELECT
    category, COUNT(name)
FROM
    pizza_types
GROUP BY category;
```

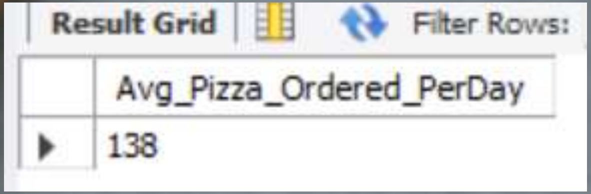


category	count(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

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

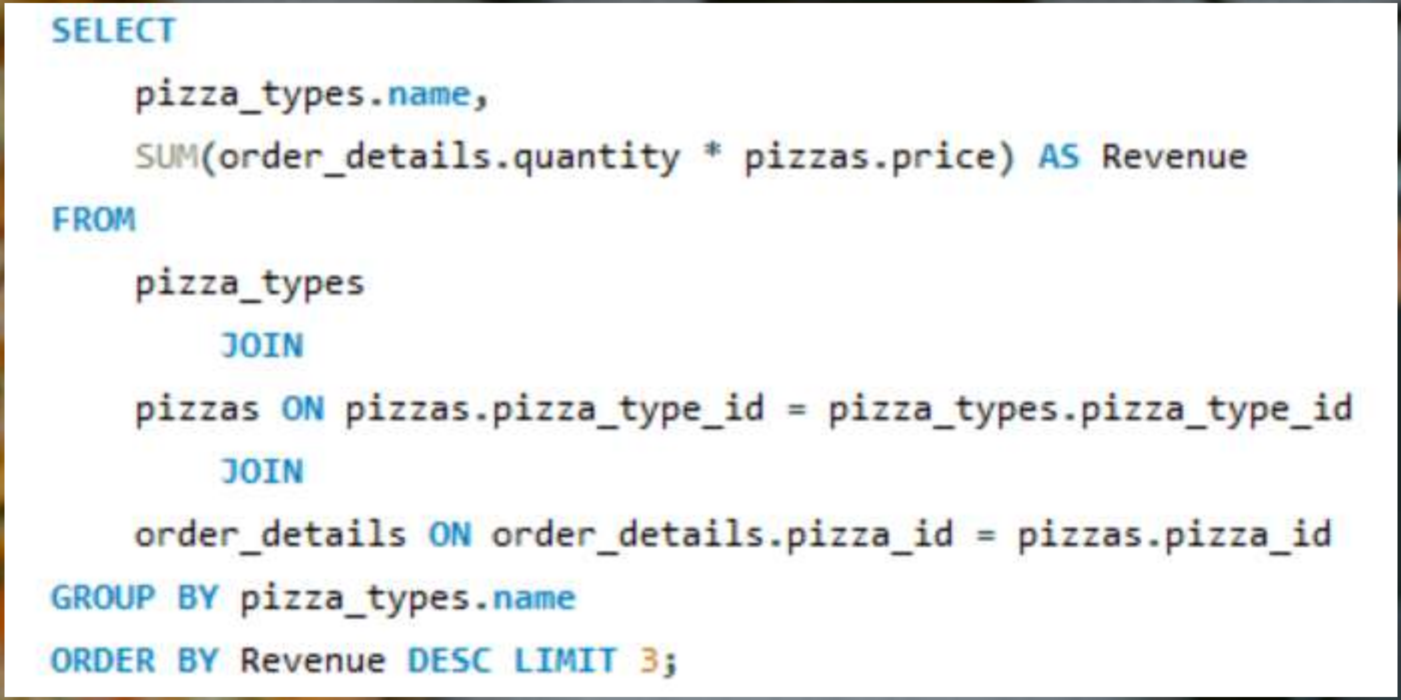


```
SELECT
    ROUND(AVG(quantity), 0) AS Avg_Pizza_Ordered_PerDay
FROM
    (SELECT
        orders.order_date, SUM(order_details.quantity) AS quantity
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY orders.order_date) AS Order_quantity;
```

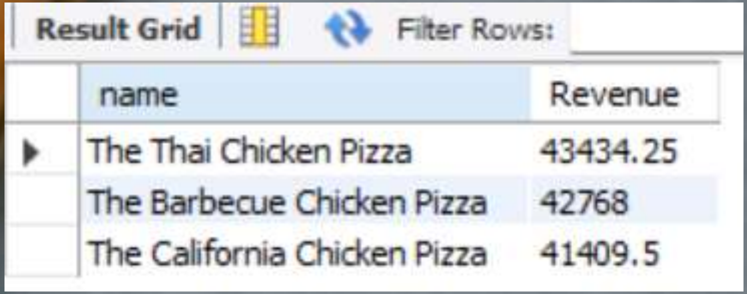


```
Avg_Pizza_Ordered_PerDay
138
```

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

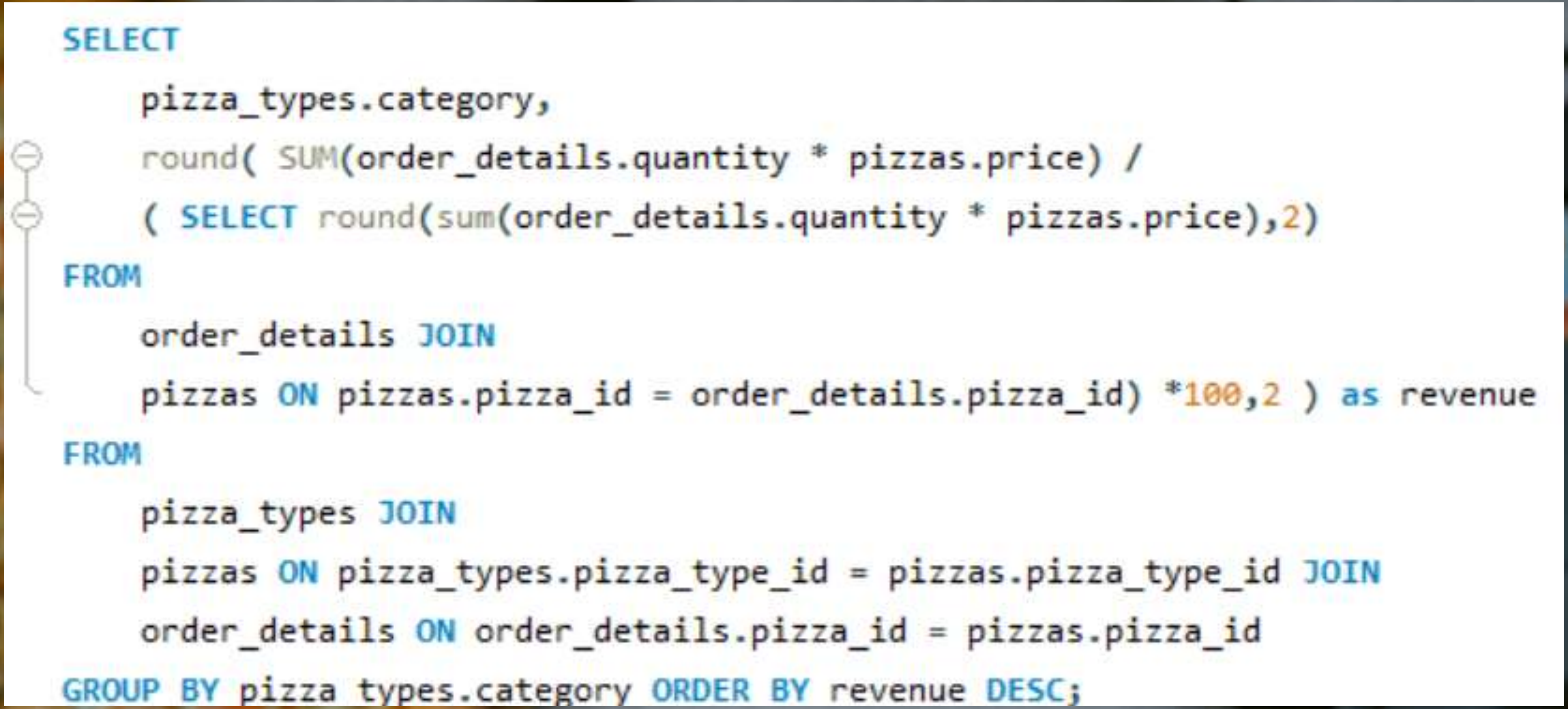


```
SELECT
    pizza_types.name,
    SUM(order_details.quantity * pizzas.price) AS Revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY Revenue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

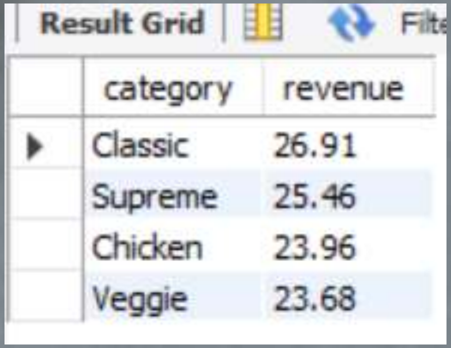


name	Revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

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

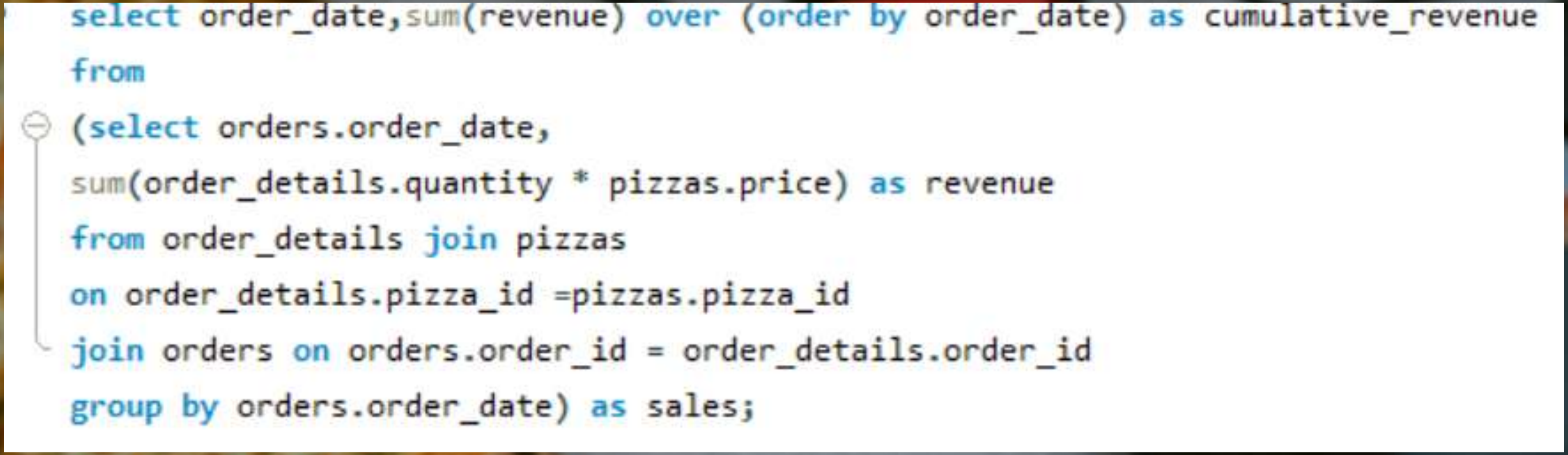


```
SELECT
    pizza_types.category,
    round( SUM(order_details.quantity * pizzas.price) /
    ( SELECT round(sum(order_details.quantity * pizzas.price),2)
FROM
    order_details JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id) *100,2 ) as revenue
FROM
    pizza_types JOIN
    pizzas ON 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 ORDER BY revenue DESC;
```

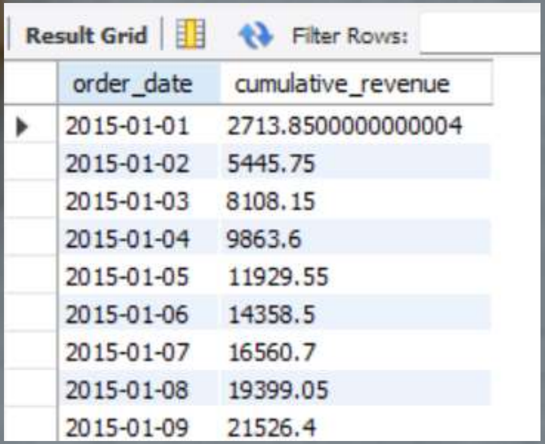


category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

12. Analyze the cumulative revenue generated over time.



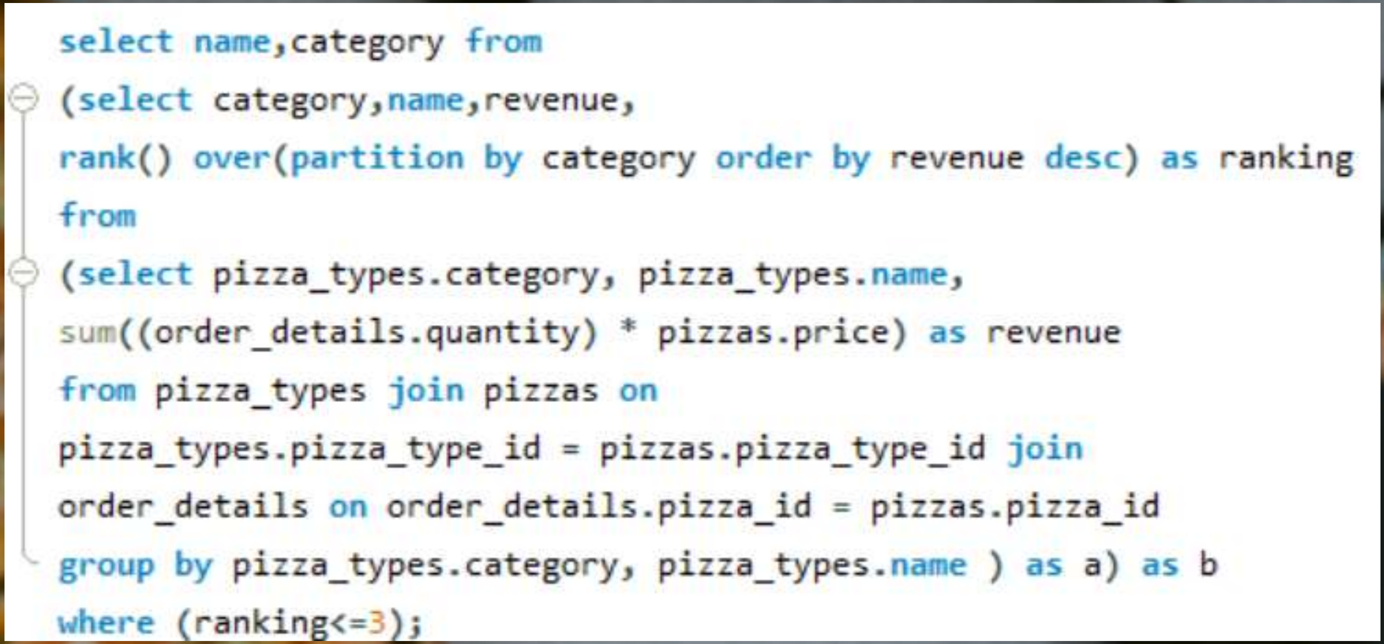
```sql
select order_date,sum(revenue) over (order by order_date) as cumulative_revenue
from
(select orders.order_date,
sum(order_details.quantity * pizzas.price) as revenue
from order_details join pizzas
on order_details.pizza_id =pizzas.pizza_id
join orders on orders.order_id = order_details.order_id
group by orders.order_date) as sales;
```



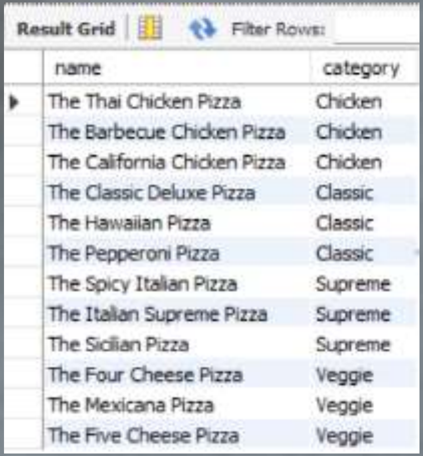
Result Grid | Filter Rows:

order_date	cumulative_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

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



```sql
select name,category from
(select category,name,revenue,
rank() over(partition by category order by revenue desc) as ranking
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 (ranking<=3);
```



Result Grid | Filter Rows:

name	category
The Thai Chicken Pizza	Chicken
The Barbecue Chicken Pizza	Chicken
The California Chicken Pizza	Chicken
The Classic Deluxe Pizza	Classic
The Hawaiian Pizza	Classic
The Pepperoni Pizza	Classic
The Spicy Italian Pizza	Supreme
The Italian Supreme Pizza	Supreme
The Sicilian Pizza	Supreme
The Four Cheese Pizza	Veggie
The Mexicana Pizza	Veggie
The Five Cheese Pizza	Veggie

Thank you