SQL Project: Pizzahut Sales Analysis

Welcome to the presentation! Let's explore the power of SQL in analyzing pizza sales data. We'll uncover insights into trends, customer behavior, and operational efficiency.

by Madhur Dhavre

Introduction: Why Data Analysis Matters for Pizza Business

Data-driven decisions

Data analysis helps pizza businesses make informed decisions. This allows for better menu planning, marketing strategies, and operational optimization.

Competitive edge

Understanding customer preferences and sales trends gives pizza businesses a competitive edge. This enables them to cater to customers' needs and attract new customers.

Pizza Salesfure
Our opian a dn thiy parta fied. Onlos your your Maner eay for and bates ror and tale.
Wovir Stecy for the Pizza Slale Conylets witns thdr cheesi Is your free.
Pizza Sales Data

Data Collection: Sourcing and Preparing the Pizza Sales Data

Source Identification

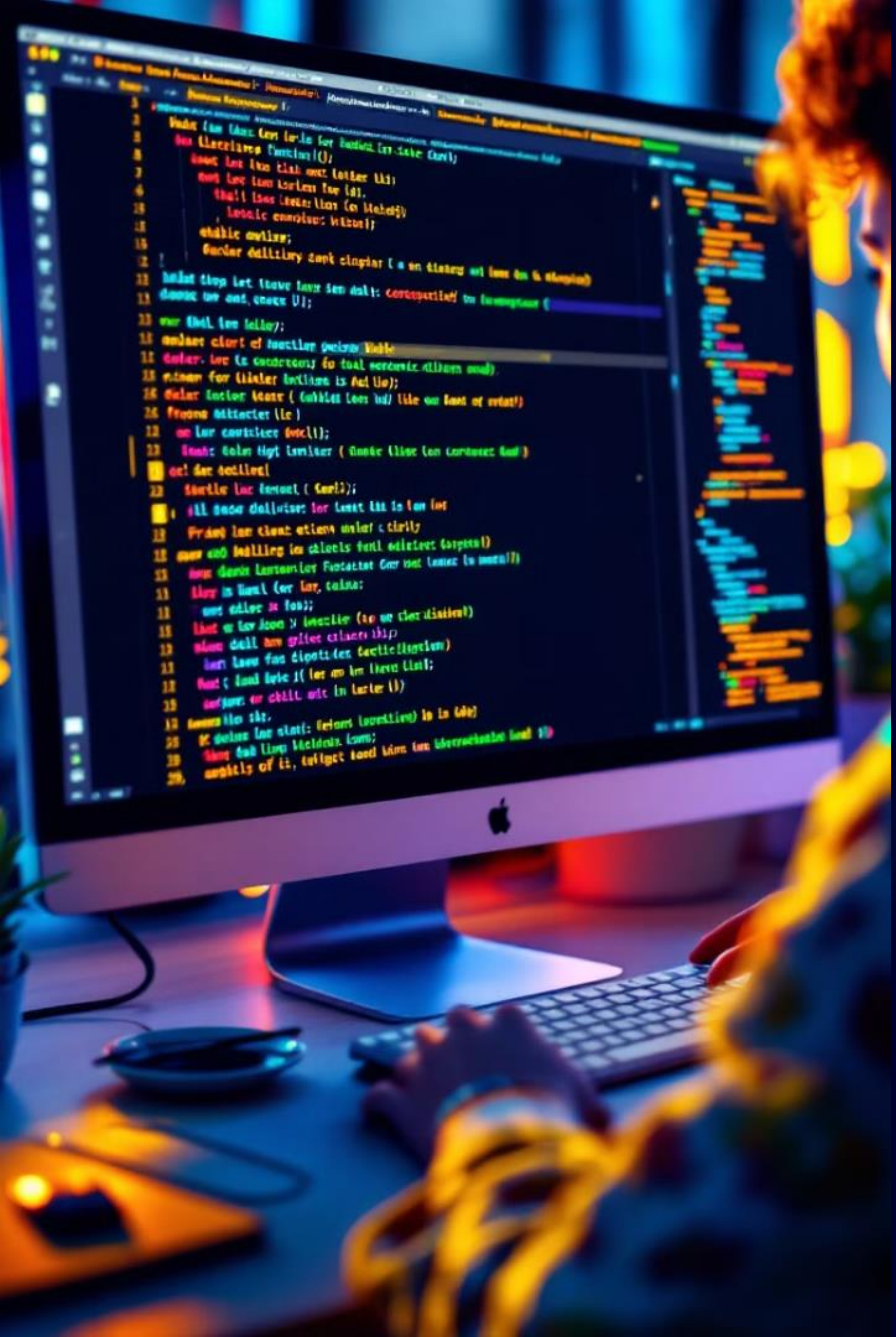
Identify the sources of pizza sales data. This could be POS systems, online ordering platforms, or customer relationship management systems.

Data Extraction

Extract the relevant data from these sources into a suitable format. This could be CSV files, SQL databases, or other data formats.

Data Cleaning

Clean and prepare the data for analysis. This includes handling missing values, inconsistent data, and formatting issues.



Data Exploration: Uncovering Insights with SQL Queries

1 **Basic Queries**

Start with basic queries to understand the structure of the data. This includes selecting, filtering, and grouping the data.

2 **Aggregate Functions**

Utilize aggregate functions like COUNT, SUM, AVG, and MAX to calculate key metrics and insights from the data.

3 **Joins and Subqueries**

Combine data from different tables using joins and subqueries to gain deeper insights into relationships between sales, customers, and menu items.

Customer Behavior: Understanding Ordering Patterns and Preferences

1

Customer Segmentation

Segment customers based on order frequency, average order value, and preferred pizza types. This can help target marketing campaigns.

2

Order Frequency

Analyze the frequency of orders placed by customers. This can help identify loyal customers and identify those who need more engagement.

3

Order Value

Examine the average order value of customers. This can help understand which customer segments contribute most to revenue.

Sales Trends: Identifying Peak Seasons and Popular Menu Items

1 Seasonal Trends

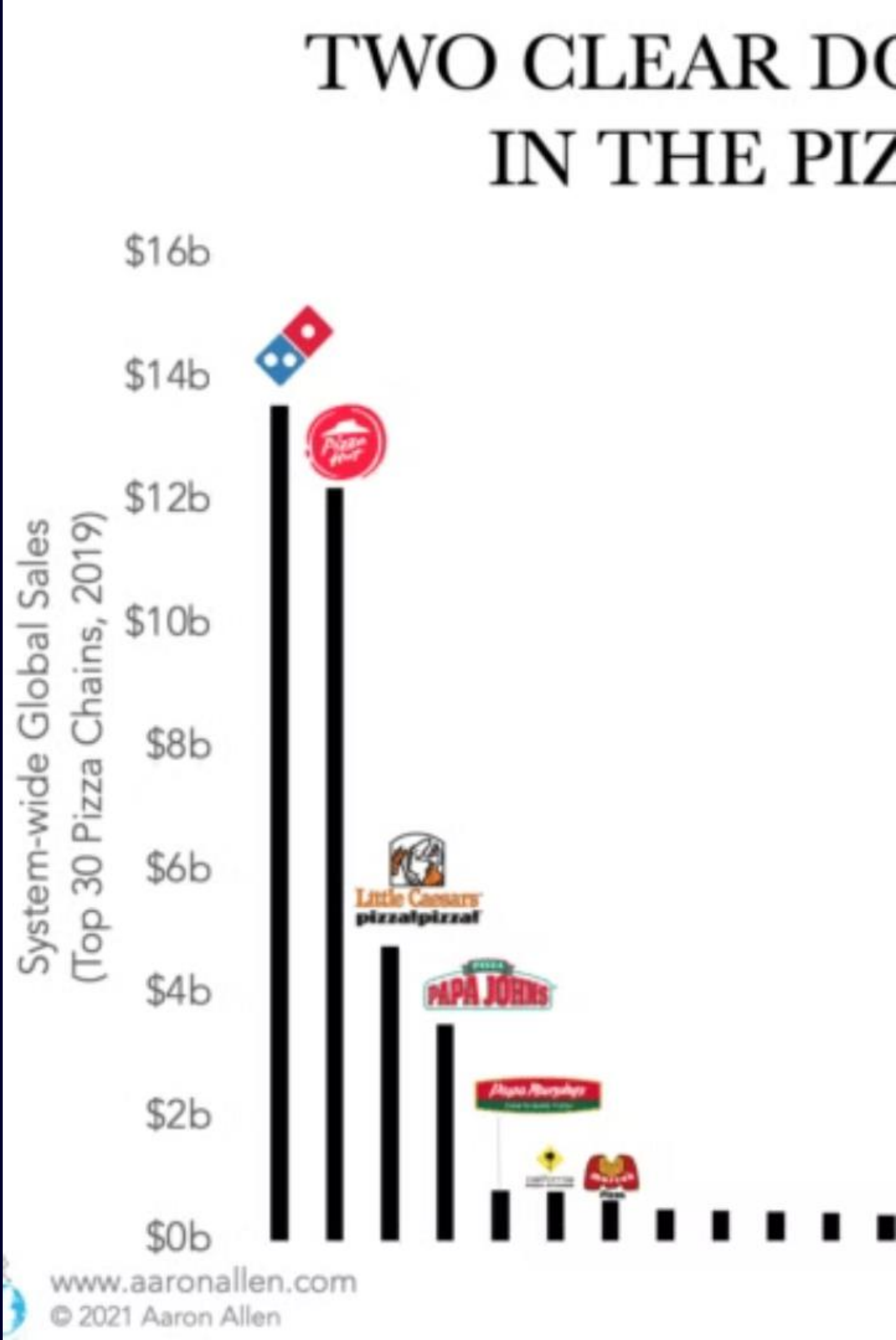
Analyze sales data to identify peak seasons. This helps optimize staffing, inventory, and marketing efforts during high demand periods.

2 Popular Menu Items

Identify the most popular pizza toppings, crust types, and sizes. This information can be used to optimize the menu and focus on popular items.

3 Sales by Day and Hour

Understand sales patterns throughout the day and week. This can help optimize staffing schedules and improve operational efficiency.



Operational Efficiency: Optimizing Inventory and Supply Chain

Delivery Optimization

Analyze delivery times and routes to identify areas for improvement. This can help reduce delivery costs and improve customer satisfaction.

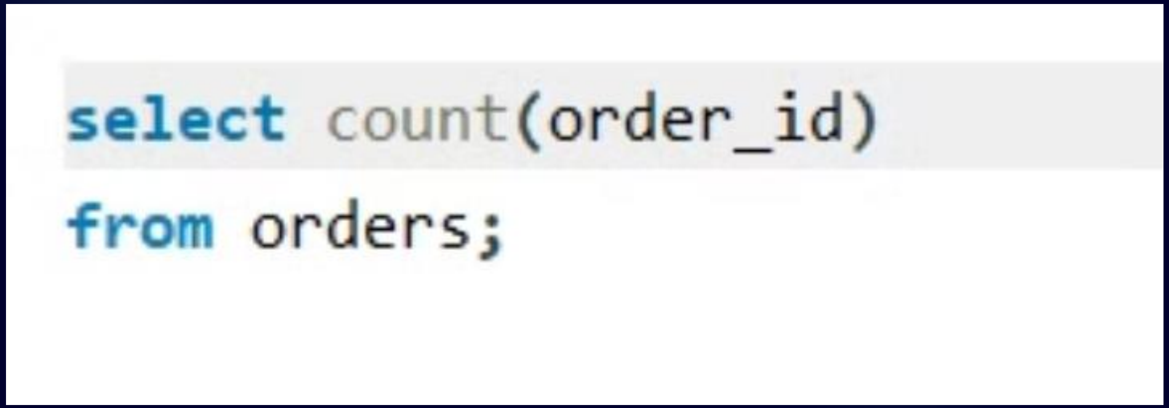
Inventory Management

Monitor inventory levels of pizza ingredients and supplies. This can help avoid stockouts and reduce waste, ensuring cost-effectiveness.

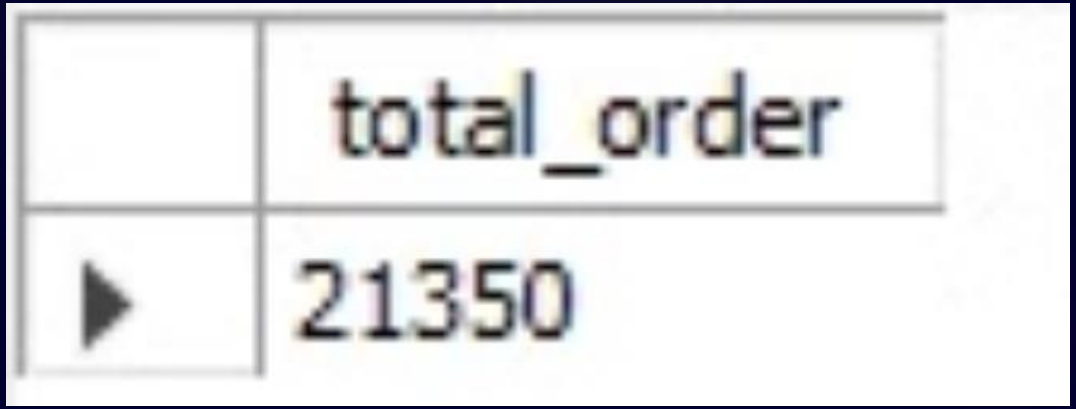
Cost Analysis

Track costs associated with ingredients, labor, and other operational expenses. This can help identify areas for cost savings and increase profitability.

Retrieve the total number of orders placed.

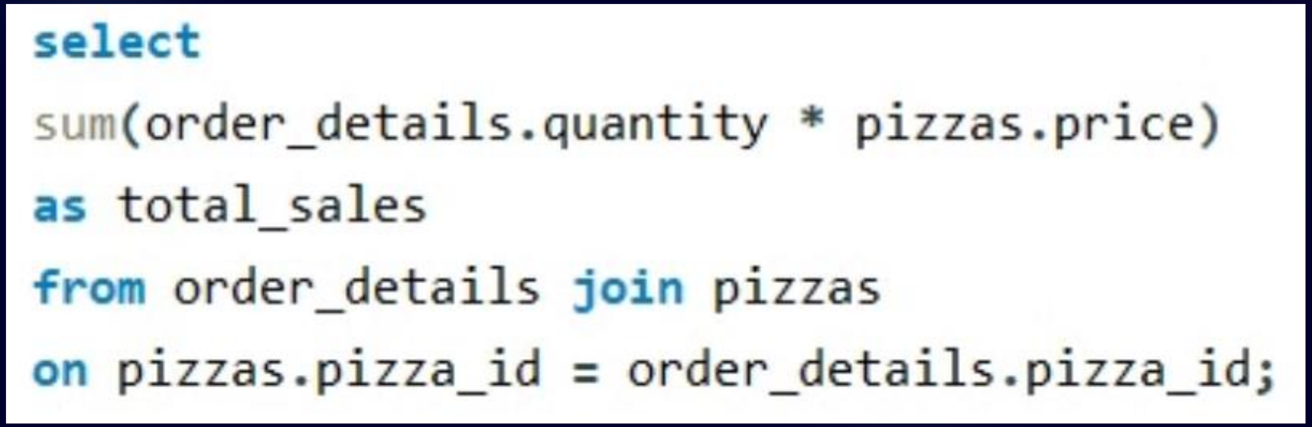


```sql
select count(order_id)
from orders;
```



```
total_order
21350
```

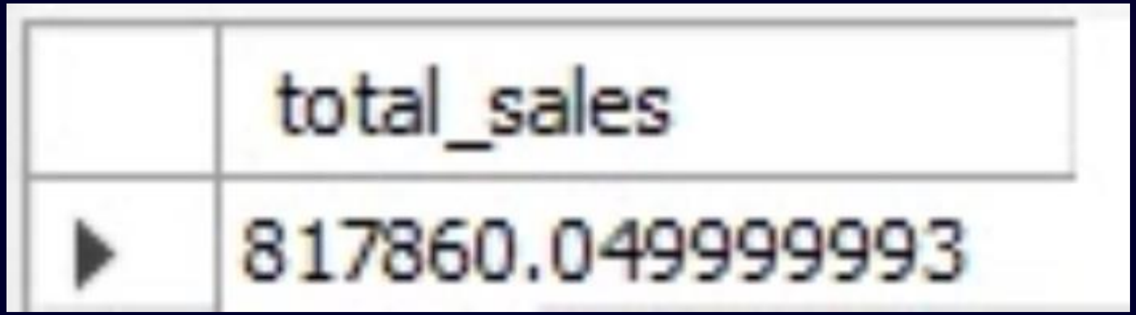
Calculate the total revenue generated from pizza sales.



```
select
sum(order_details.quantity * pizzas.price)
as total_sales
from order_details join pizzas
on pizzas.pizza_id = order_details.pizza_id;
```

Since the revenue-related data is provided in two distinct tables, we have used joins to determine the overall revenue.

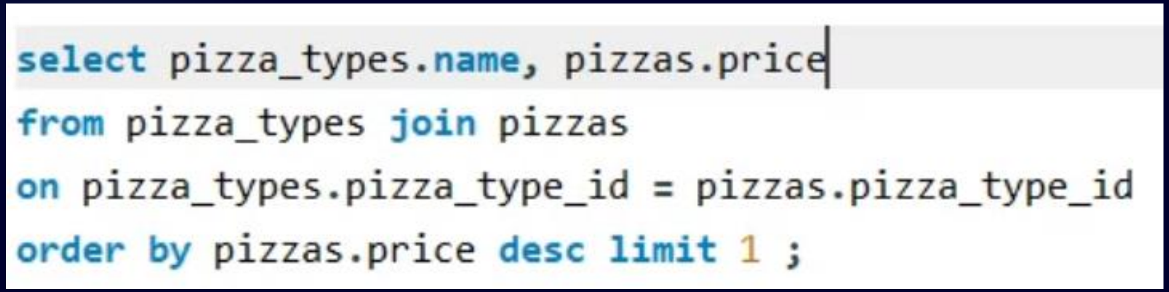
To calculate the revenue, we have to do sum of the quantity and price;



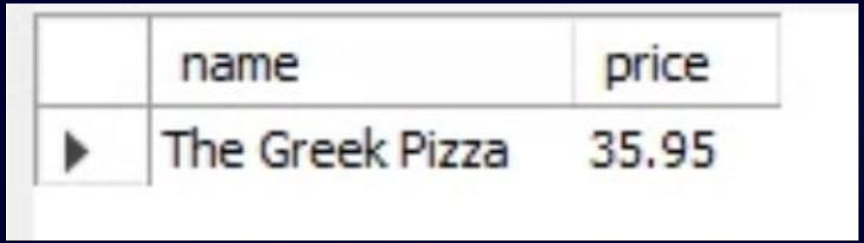
	total_sales
▶	817860.049999993

Sum(quantity * price) = total sales

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 ;
```

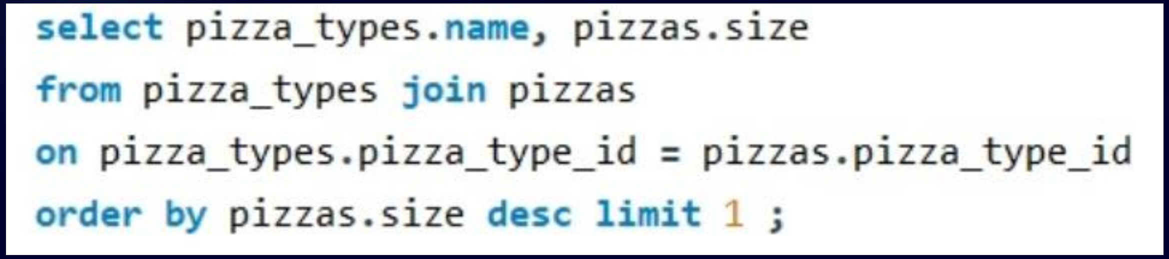


name	price
The Greek Pizza	35.95

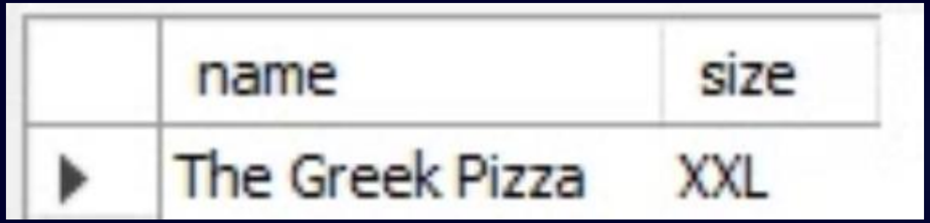
To determine the most expensive pizza, we combined two distinct tables, one for pizza varieties and another for pizzas themselves.

We used the order by clause in descending order and set the limit to 1 because we needed the highest-priced pizza value.

Identify the most common pizza size ordered.



```
select pizza_types.name, pizzas.size
from pizza_types join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
order by pizzas.size desc limit 1 ;
```



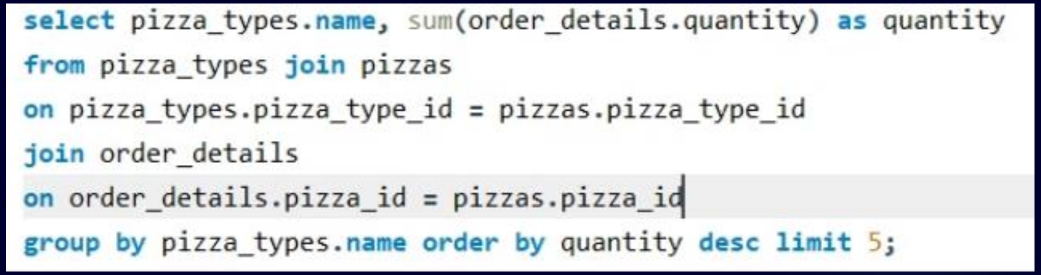
	name	size
▶	The Greek Pizza	XXL

To find the most common pizza, we connected two separate tables, such as pizza types and pizzas.

Because we require the largest value of the pizza, we used order by clause in descending order and set the limit to one to obtain the most popular pizza.

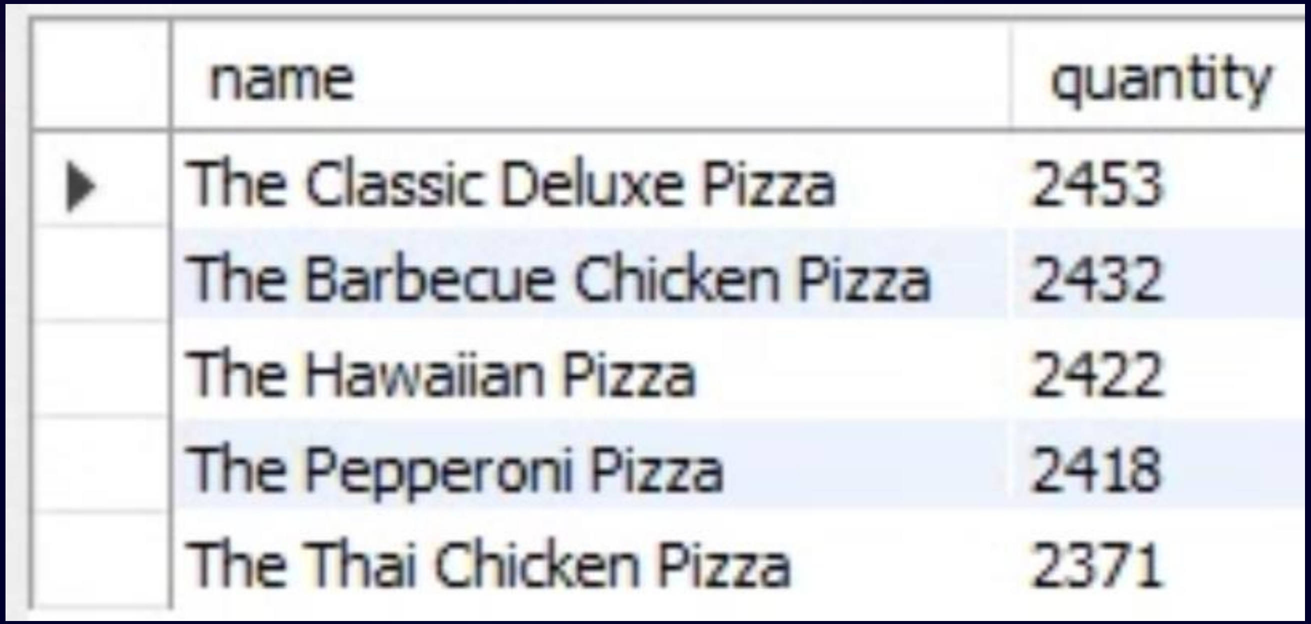
List the top 5 most ordered pizza types along with their quantities.

In the above query, I calculated the total quantity from the order information.



```
select pizza_types.name, 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.name order by quantity desc limit 5;
```

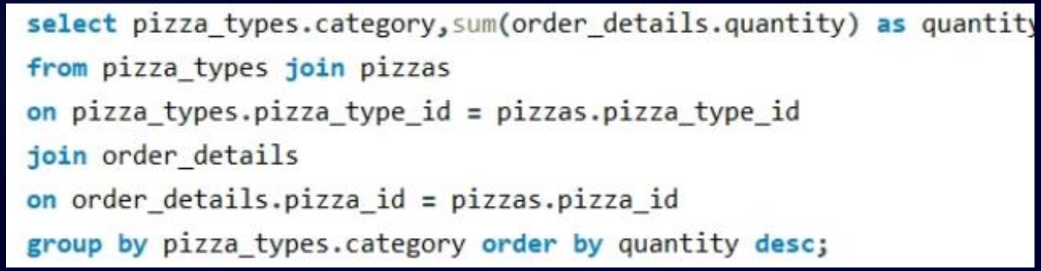
Because there is nothing in common between the two required tables, I combined the order details from the third table.



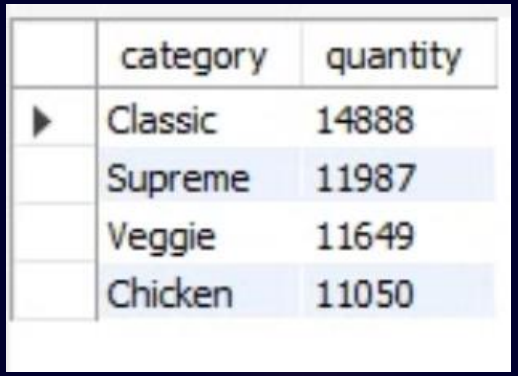
name	quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

The end result is obtained by inserting a group and order-by clause on the pizza type and quantity.

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;
```

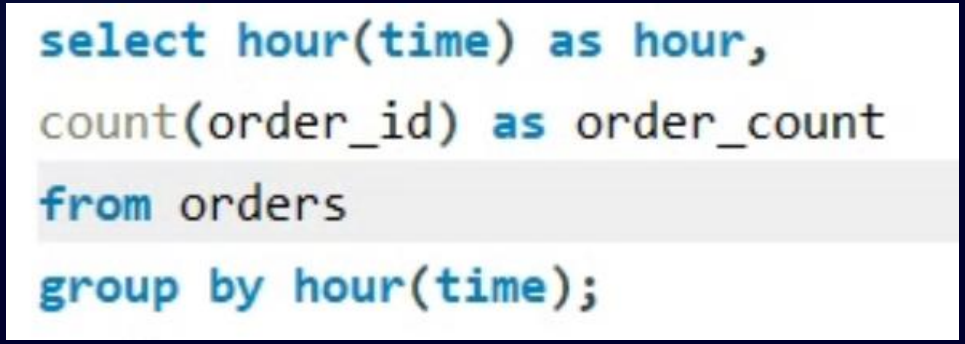


category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

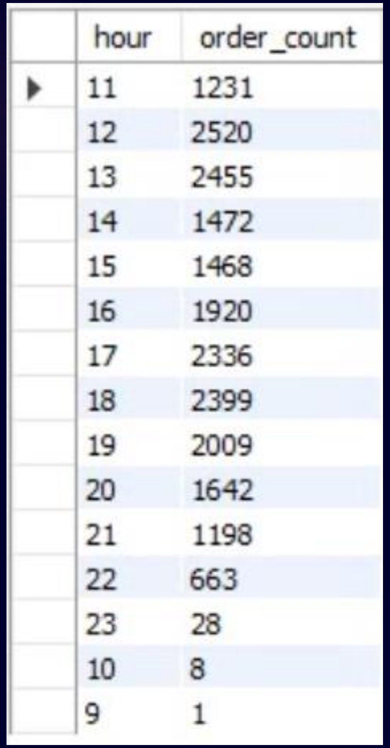
To get the overall quantity of each pizza category, we must merge three tables.

First, we must combine pizza types and pizzas; after that, we must combine them with order information because the pizza ID is shared by both. Finally, I used a group and order-by clause on the pizza type category and quantity to determine the total quantity of each pizza category.

Determine the distribution of orders by hour of the day.



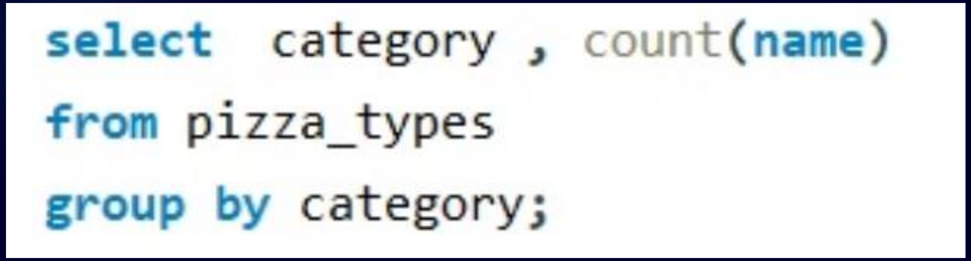
```
select hour(time) as hour,
count(order_id) as order_count
from orders
group by hour(time);
```



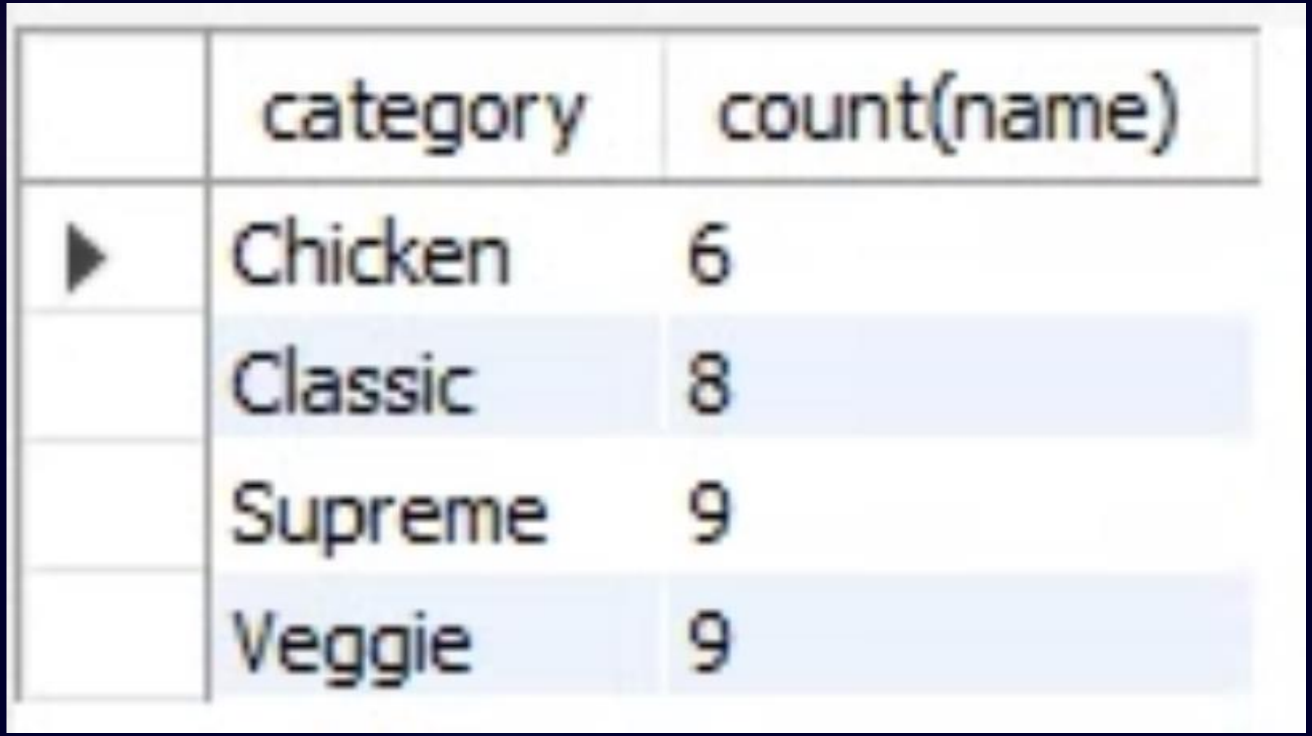
hour	order_count
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009
20	1642
21	1198
22	663
23	28
10	8
9	1

This query uses the hour operator and the count function to determine the distribution of orders by hour of the day.

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

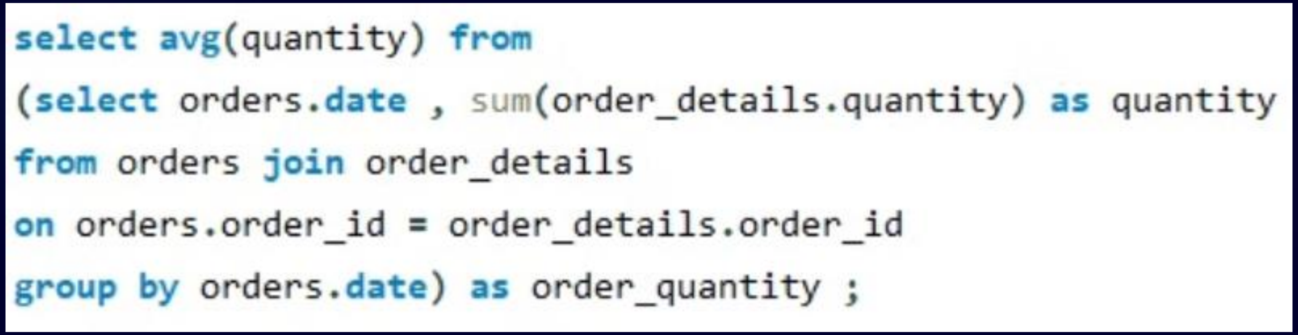


```
select  category , count(name)
from pizza_types
group by category;
```

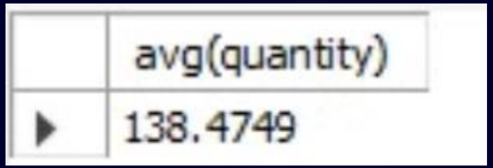


	category	count(name)
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

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



```
select avg(quantity) 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 ;
```



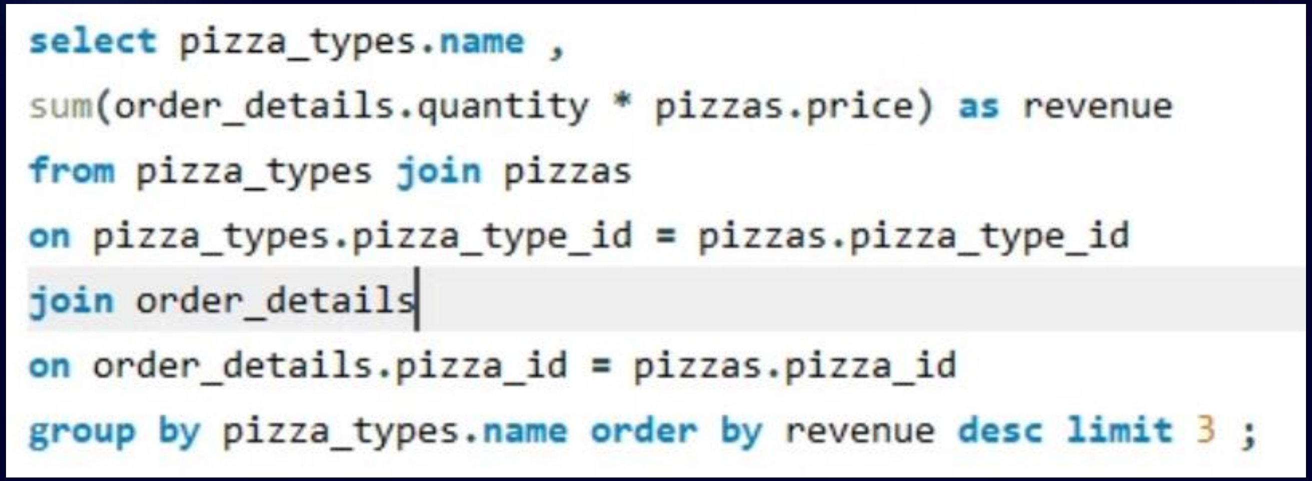
	avg(quantity)
▶	138.4749

When determining the average number of pizzas ordered per day, we must create a subquery to determine the average of the total quantity.

Subquery allows us to design sophisticated questions based on the results of another query.

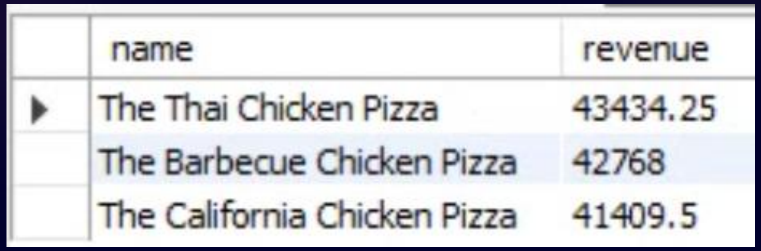
The given syntax consists of two choose statements, the first for the average and the second for the sum of the quantities.

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 ;
```

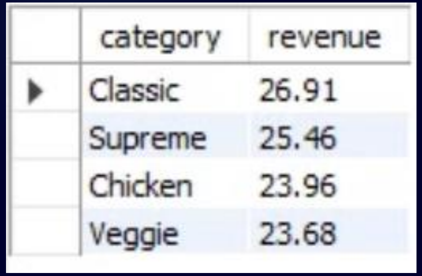
We must add together the quantity and pizza pricing to find the top three most ordered pizzas in terms of income.



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

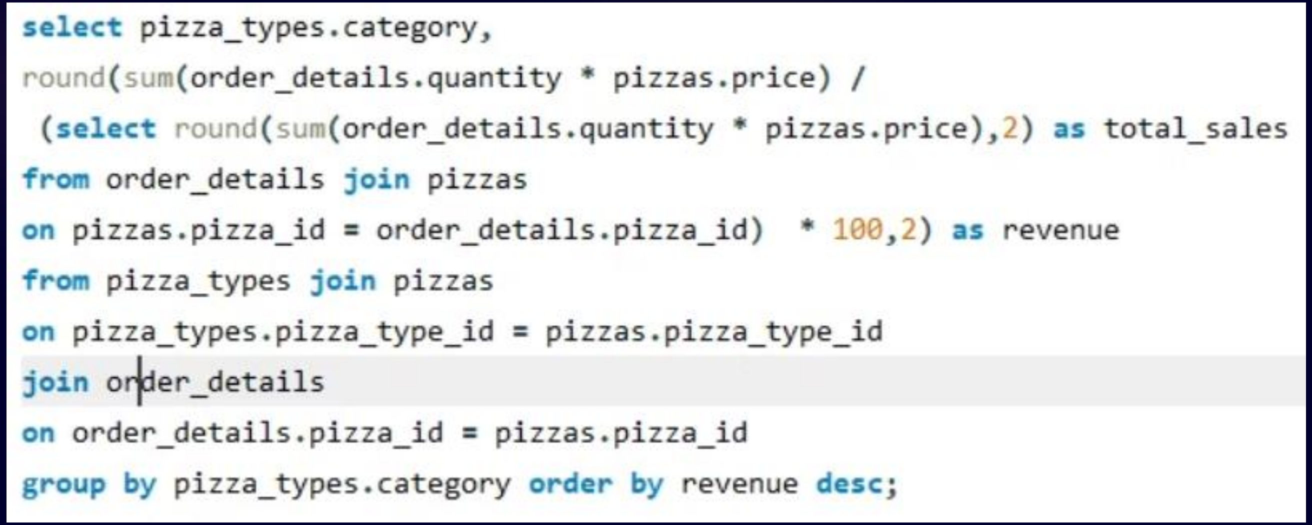
Finally, we must group and sort by revenue in descending order, and because we only need the top three results, we must limit the total revenue to three.

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



	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

The query connects three tables: pizza types, pizzas, and order details, determining the revenue per pizza category. The SUM() method aggregates revenue per pizza for each category, while the total revenue from all orders and pizzas is determined by multiplying quantity and price for each pizza

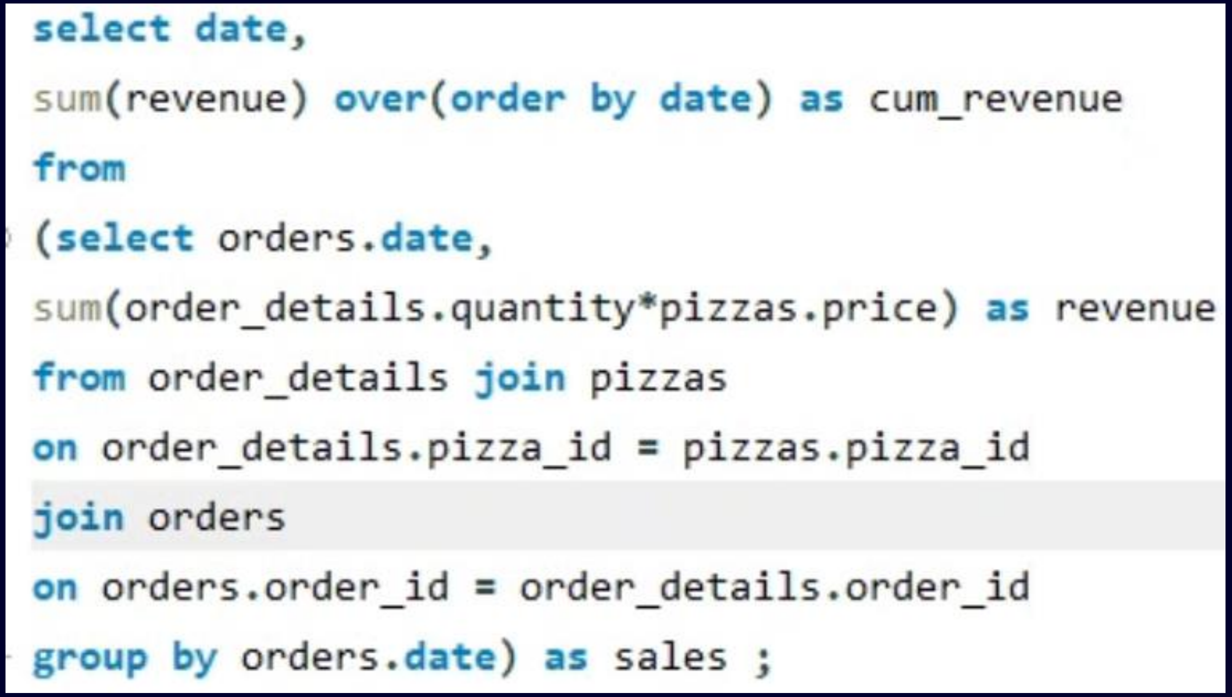


```
select pizza_types.category,
round(sum(order_details.quantity * pizzas.price) /
 (select round(sum(order_details.quantity * pizzas.price),2) as total_sales
from order_details join pizzas
on pizzas.pizza_id = order_details.pizza_id)  * 100,2) as revenue
from pizza_types join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join order_details
on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.category order by revenue desc;
```

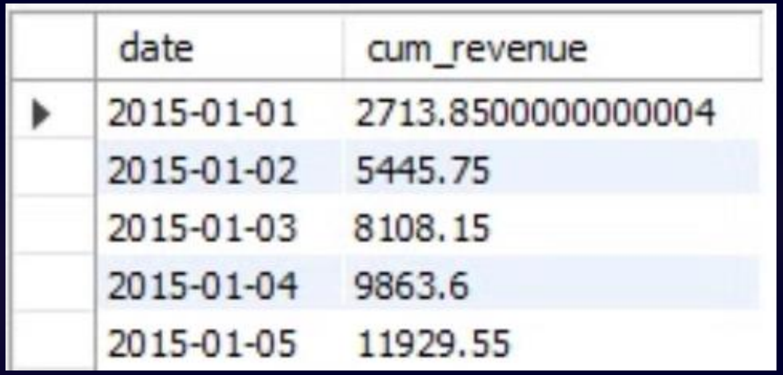
The total sales are rounded to two decimal places using the ROUND(..., 2) method. The GROUP BY pizza types ,category ensures each category is calculated independently.

Analyze the cumulative revenue generated over time.

The query calculates the total income for each date by joining the order details and pizzas tables, linking each order id to its date, and multiplying the number of pizzas in the order details table by the pizza price from the pizzas table



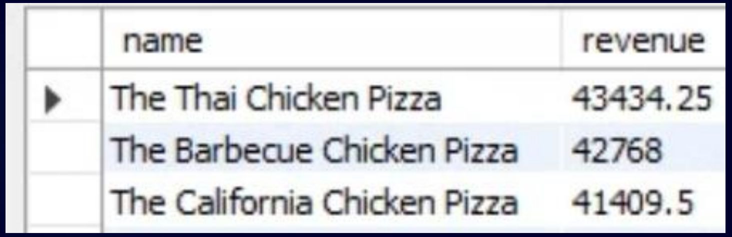
```
select date,
sum(revenue) over(order by date) as cum_revenue
from
(select orders.date,
sum(order_details.quantity*pizzas.price) as revenue
from order_details join pizzas
on order_details.pizza_id = pizzas.pizza_id
join orders
on orders.order_id = order_details.order_id
group by orders.date) as sales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



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

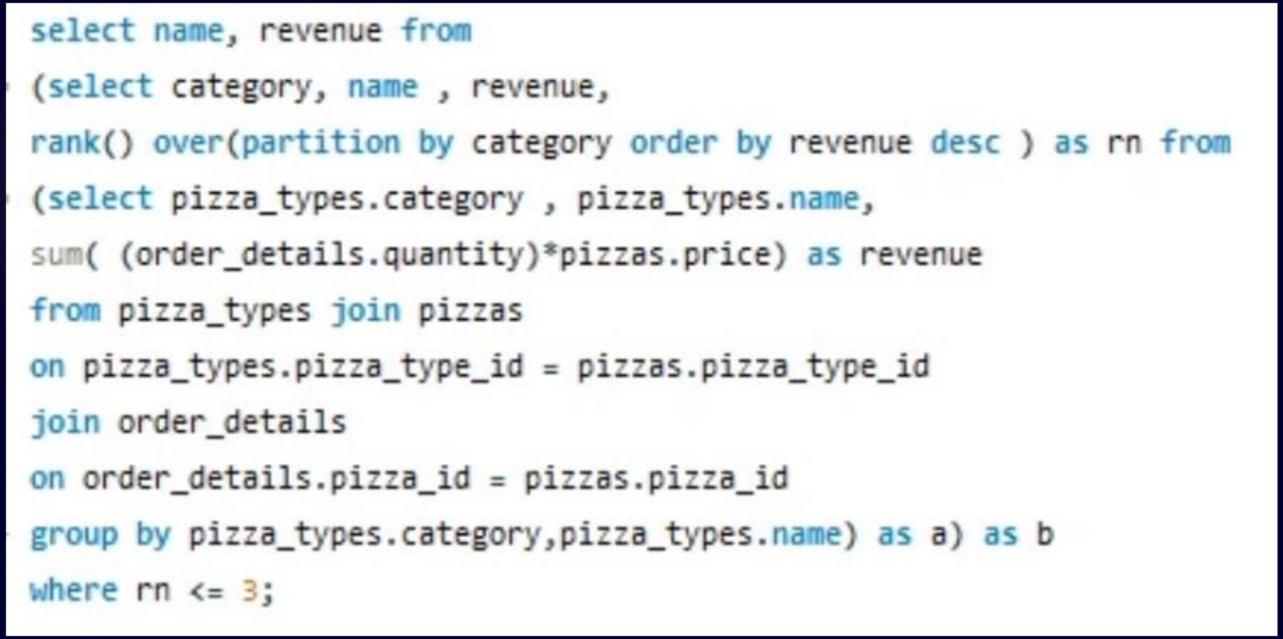
The total revenue for each date is calculated by aggregating the data and grouping it by order date. The daily revenue is derived from the inner query's sales results, which are expanded upon by the outer query. The query uses a window function to determine the cumulative revenue, which is then grouped by date. The query also includes subqueries for easier processing of raw data.

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



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

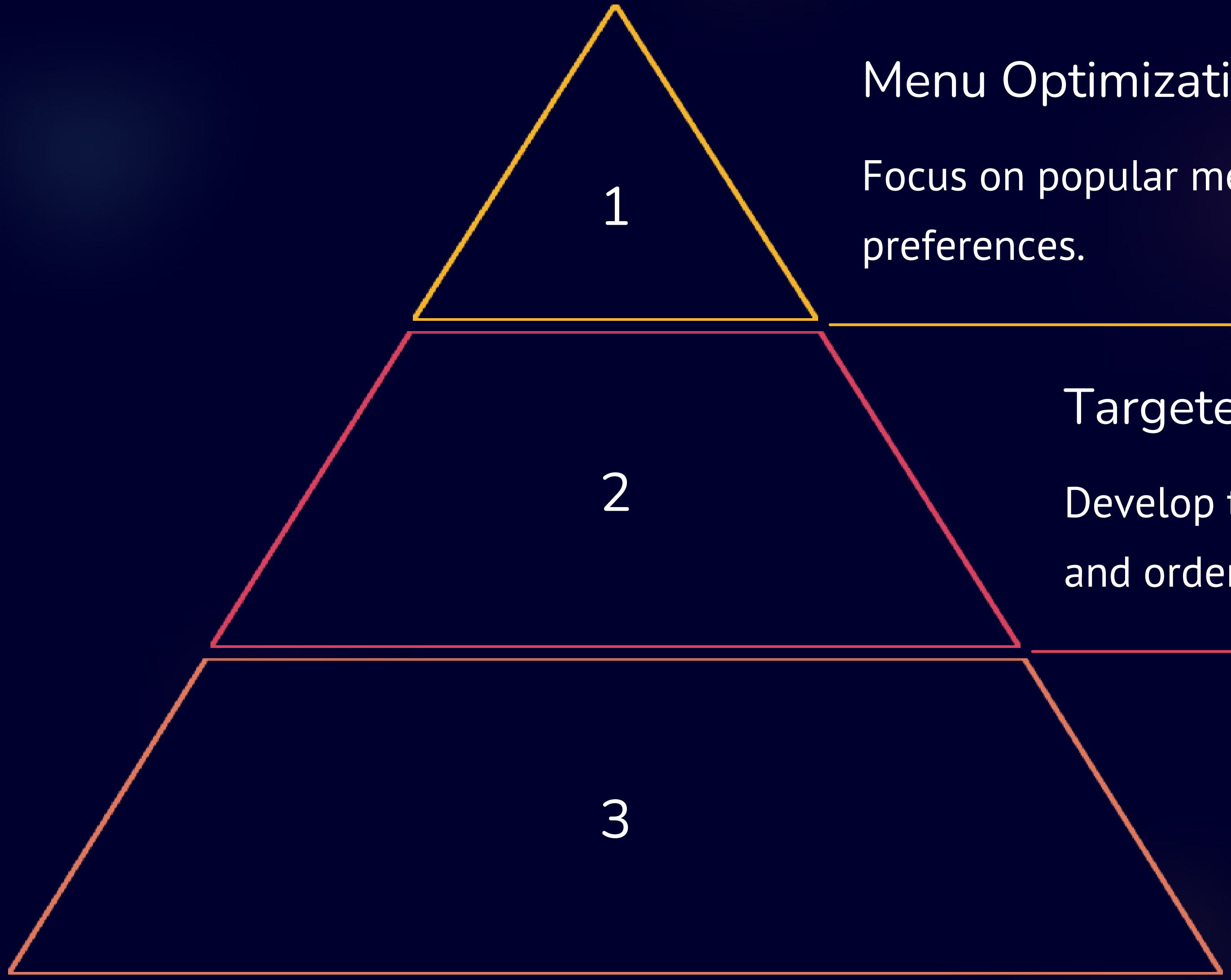
This subquery aims to determine the overall income for every pizza in every category using tables like pizza_types, pizzas, and order_details. The revenue is calculated using the formula Order_details.quantity * pizzas.price. The results are grouped by pizza name and category, and ranked according to revenue within each category. The Rank Function is used to rank each pizza in its category, descending order.



```
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 a) as b
where rn <= 3;
```

he outer query narrows down the results to show only the top three pizzas in each category, ensuring only the top three pizzas are displayed. The final product includes the names and earnings of the top three pizzas by revenue.

Recommendations: Actionable Insights to Grow the Pizza Business



Menu Optimization

Focus on popular menu items and introduce new items based on customer preferences.

Targeted Marketing

Develop targeted marketing campaigns based on customer segments and order history.

Improved Operations

Optimize delivery routes, inventory management, and staffing based on data insights.

Conclusion and Next Steps

SQL analysis is a powerful tool for pizza businesses to gain valuable insights and make data-driven decisions. By leveraging this information, businesses can optimize operations, improve customer satisfaction, and achieve greater success. Continuous analysis and monitoring is key to remain competitive and stay ahead of the curve.