

super delicious
PIZZA

Canva

ORDER NOW



Hello!



- I'm Neha Reddy,
- I've leveraged SQL queries to conduct an in-depth analysis of pizza sales data.

DATABASE BRIEF

DATABASE NAME: DOMINOS

ORDER DETAILS TABLE

ORDERS TABLE

PIZZA TYPES

PIZZAS

CASE STUDY

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.

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.

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.

SQL File 8* × order_details orders pizza_types pizzas

8 • ⏷ create table order_details (

9 order_details_id int Primary key,

10 order_id int not null,

11 pizza_id text not null,

12 quantity int not null

13);

14

15 -- Retrieve the total number of orders placed.

16 • select

17 count(order_id) as Total_orders

18 from

19 dominos.orders;

20

Result Grid | Filter Rows: Export: Wrap Cell Content:

Total_orders
21350

SQL File 8*

order_details

pizzas

orders

pizza_types

SQL File 9*

```
1   --- Calculate the total revenue generated from pizza sales.  
2 * select  
3   round (sum((order_details.quantity*pizzas.price)) ,2) as total_Revenue  
4   from  
5   order_details  
6   join pizzas  
7   on  
8   order_details.pizza_id=pizzas.pizza_id;  
9
```

Result Grid



Filter Rows

Export



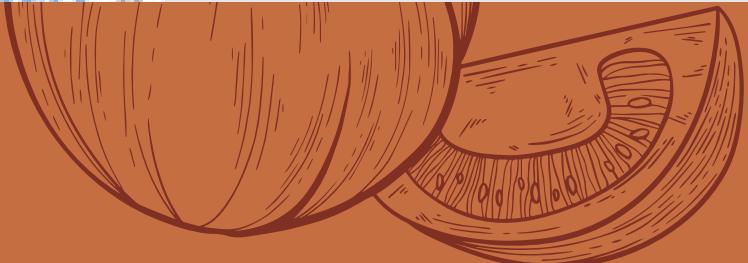
Wrap Cell Contents



total_Revenue

817860.05

Result 6





SQL File 8*

order_details

pizzas

orders

pizza_types

SQL File 9*

SQL File 10*

```
1 -- Identify the highest-priced pizza.--
2
3 * select
4   pizzas.price,
5   pizza_types.name
6   from pizzas
7   join
8   pizza_types
9   on
10  pizzas.pizza_type_id=pizza_types.pizza_type_id
11  order by pizzas.price desc limit 1 ;
12
13
```

Result Grid Filter Rows

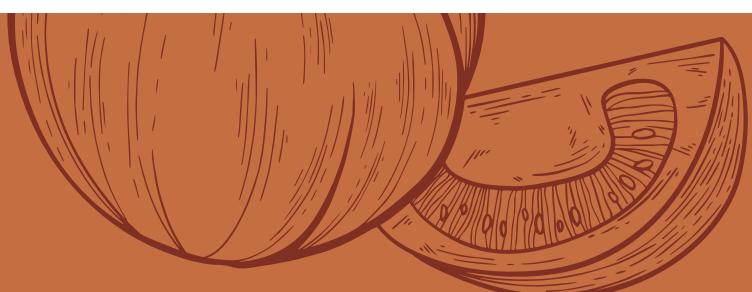
Export

Wrap Cell Content

Fetch rows



price	name
35.95	The Greek Pizza



```
1 -- Identify the most common pizza size ordered.  
2  
3 * select  
4 COUNT( order_details.order_id) AS order_count,  
5 pizzas.size  
6 from order_details  
7 join pizzas  
8 on  
9 order_details.pizza_id=pizzas.pizza_id  
10 GROUP BY pizzas.size;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Contents: |

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



```
1 -- List the top 5 most ordered pizza types along with their quantities.
2 • select
3     sum((order_details.quantity)) as quantity,
4     pizza_types.name
5     from pizza_types
6     join pizzas
7     on pizza_types.pizza_type_id=pizzas.pizza_type_id
8     join order_details
9     on order_details.pizza_id=pizzas.pizza_id
10    group by pizza_types.name
11    order by quantity desc limit 5 ;
```

Result Grid | Filter Rows: Export: Wrap Cell Contents: Fetch rows:

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

Result 12



order_details pizza_types pizzas orders SQL File 8* SQL File 9* SQL File 10* SQL File 11* SQL File 12*

Limit to 1000 rows

```
1 -- Join the necessary tables to find the total quantity of each pizza category ordered.
2 * select
3   sum(order_details.quantity),
4   pizza_types.category
5   from pizza_types
6   join pizzas
7   on pizza_types.pizza_type_id=pizzas.pizza_type_id
8   join order_details
9   on order_details.pizza_id=pizzas.pizza_id
10  group by pizza_types.category;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	sum(order_details.quantity)	category
1	14888	Classic
2	11649	Veggie
3	11987	Supreme
4	11050	Chicken

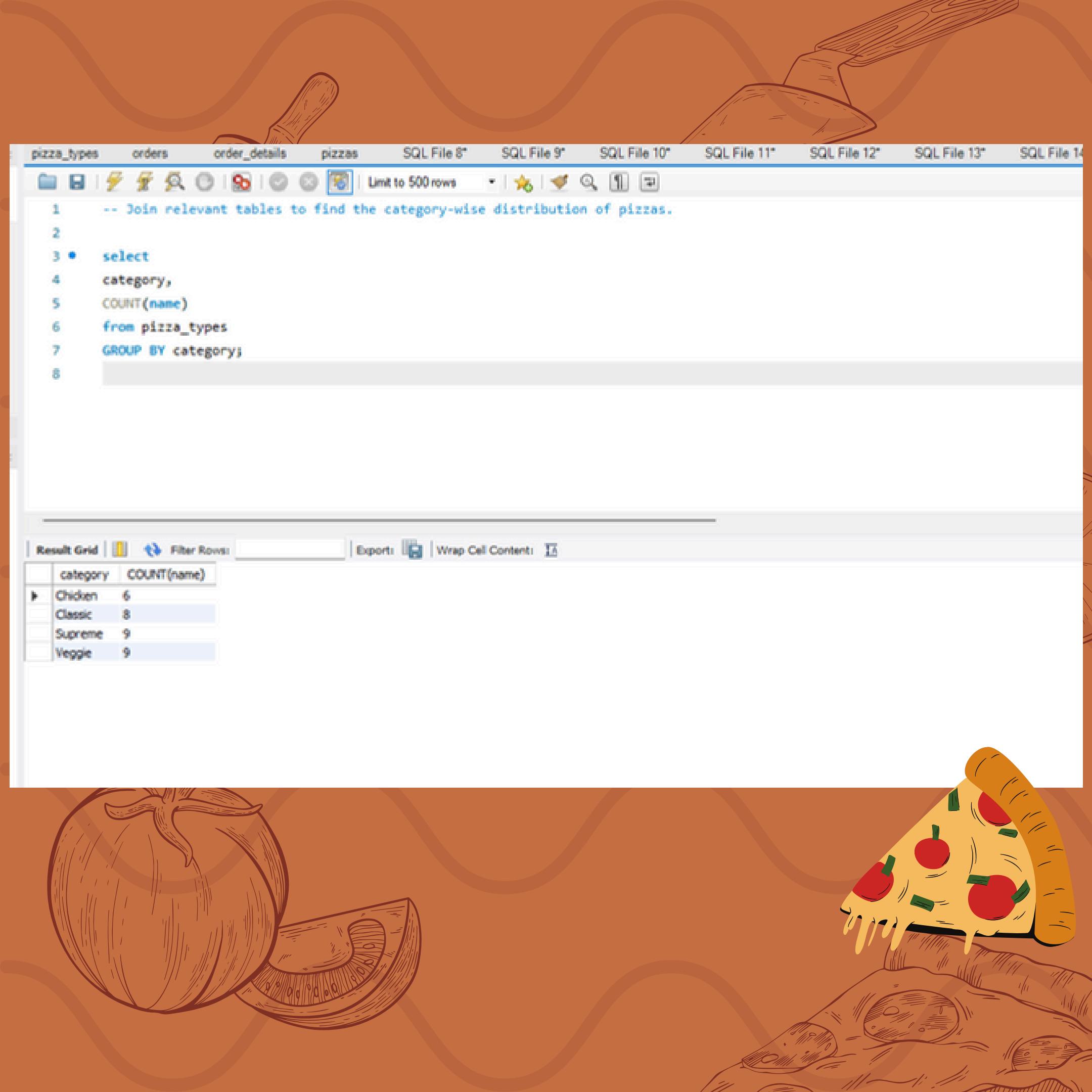
orders order_details pizzas pizza_types SQL File 8* SQL File 9* SQL File 10* SQL File 11* SQL File 12* S

1 -- Determine the distribution of orders by hour of the day
2
3 • select * from orders;
4
5 • select hour(order_time) as Hours,
6 count((order_id)) as order_count
7 from
8 orders
9 group by hour(order_time);

Result Grid | Filter Rows: Export: Wrap Cell Content:

Hours	order_count
9	1
10	8
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009

Result 4

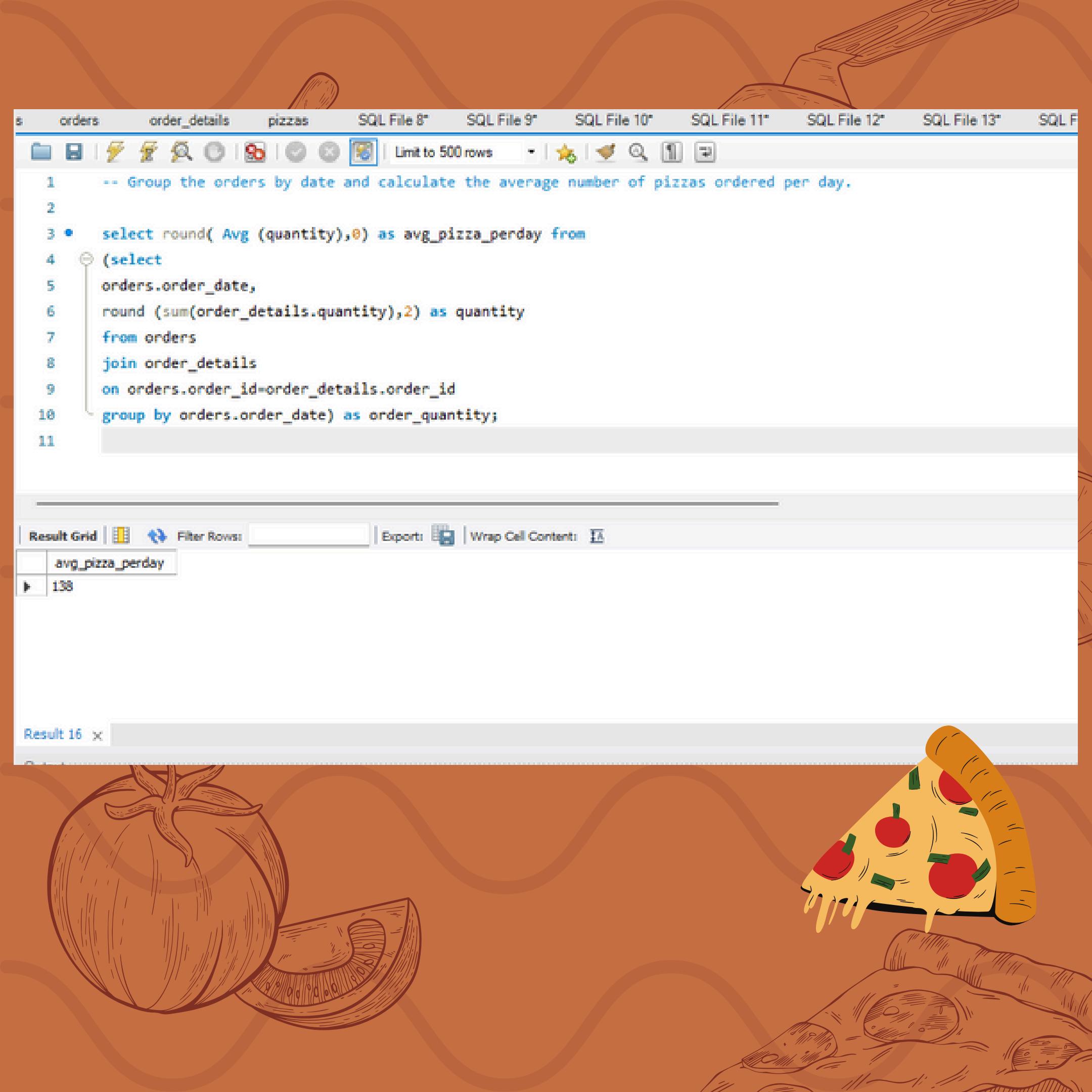


A screenshot of a MySQL Workbench interface showing a SQL query results window. The top navigation bar includes tabs for 'pizza_types', 'orders', 'order_details', 'pizzas', and several 'SQL File' tabs (8*, 9*, 10*, 11*, 12*, 13*, 14*). Below the tabs is a toolbar with icons for file operations, search, and refresh.

```
1 -- Join relevant tables to find the category-wise distribution of pizzas.  
2  
3 * select  
4   category,  
5   COUNT(name)  
6   from pizza_types  
7   GROUP BY category;  
8
```

The results grid shows the following data:

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



S orders order_details pizzas SQL File 8* SQL File 9* SQL File 10* SQL File 11* SQL File 12* SQL File 13* SQL F

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

2

3 • select round(Avg (quantity),0) as avg_pizza_perday from

4 (select

5 orders.order_date,

6 round (sum(order_details.quantity),2) as quantity

7 from orders

8 join order_details

9 on orders.order_id=order_details.order_id

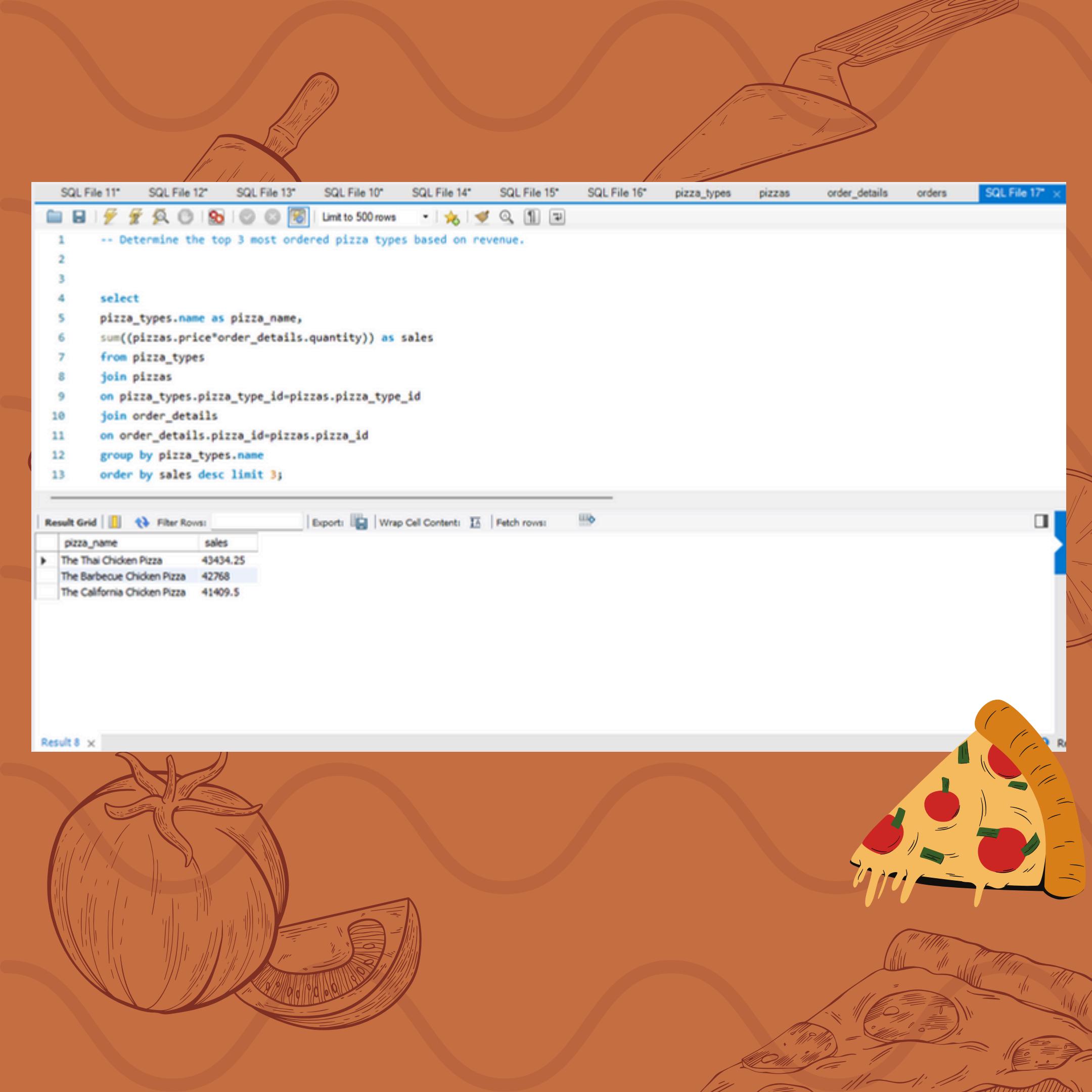
10 group by orders.order_date) as order_quantity;

11

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

avg_pizza_perday
138

Result 16 X

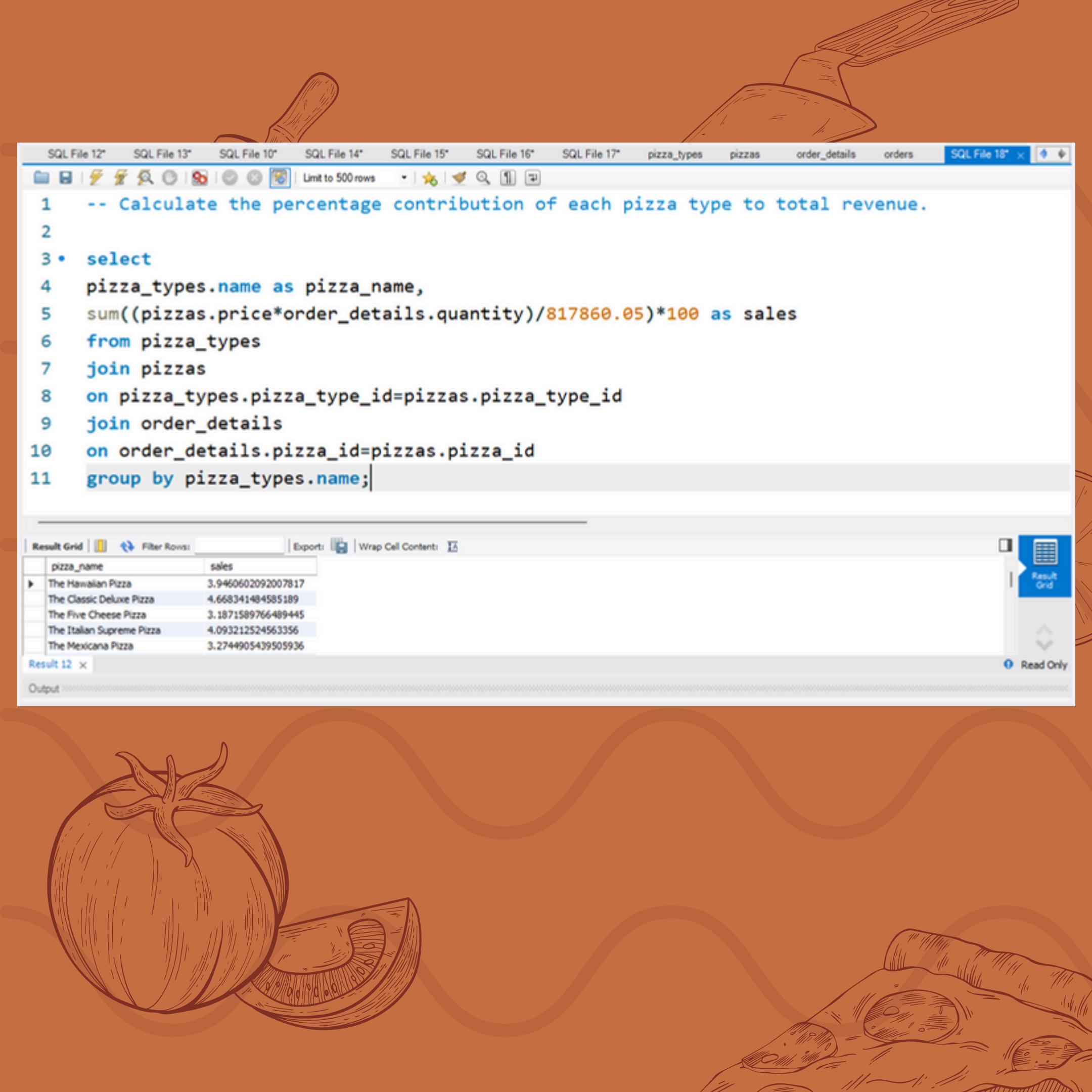


A screenshot of a SQL query editor window. The title bar shows multiple tabs: SQL File 11*, SQL File 12*, SQL File 13*, SQL File 10*, SQL File 14*, SQL File 15*, SQL File 16*, pizza_types, pizzas, order_details, orders, and SQL File 17*. Below the tabs is a toolbar with icons for file operations, search, and refresh. A dropdown menu says "Limit to 500 rows". The main area contains a SQL query:

```
1  -- Determine the top 3 most ordered pizza types based on revenue.  
2  
3  
4  select  
5    pizza_types.name as pizza_name,  
6    sum((pizzas.price*order_details.quantity)) as sales  
7  from pizza_types  
8  join pizzas  
9  on pizza_types.pizza_type_id=pizzas.pizza_type_id  
10 join order_details  
11 on order_details.pizza_id=pizzas.pizza_id  
12 group by pizza_types.name  
13 order by sales desc limit 3;
```

The result grid shows the output of the query:

pizza_name	sales
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5



SQL File 12* SQL File 13* SQL File 10* SQL File 14* SQL File 15* SQL File 16* SQL File 17* pizza_types pizzas order_details orders SQL File 18* X

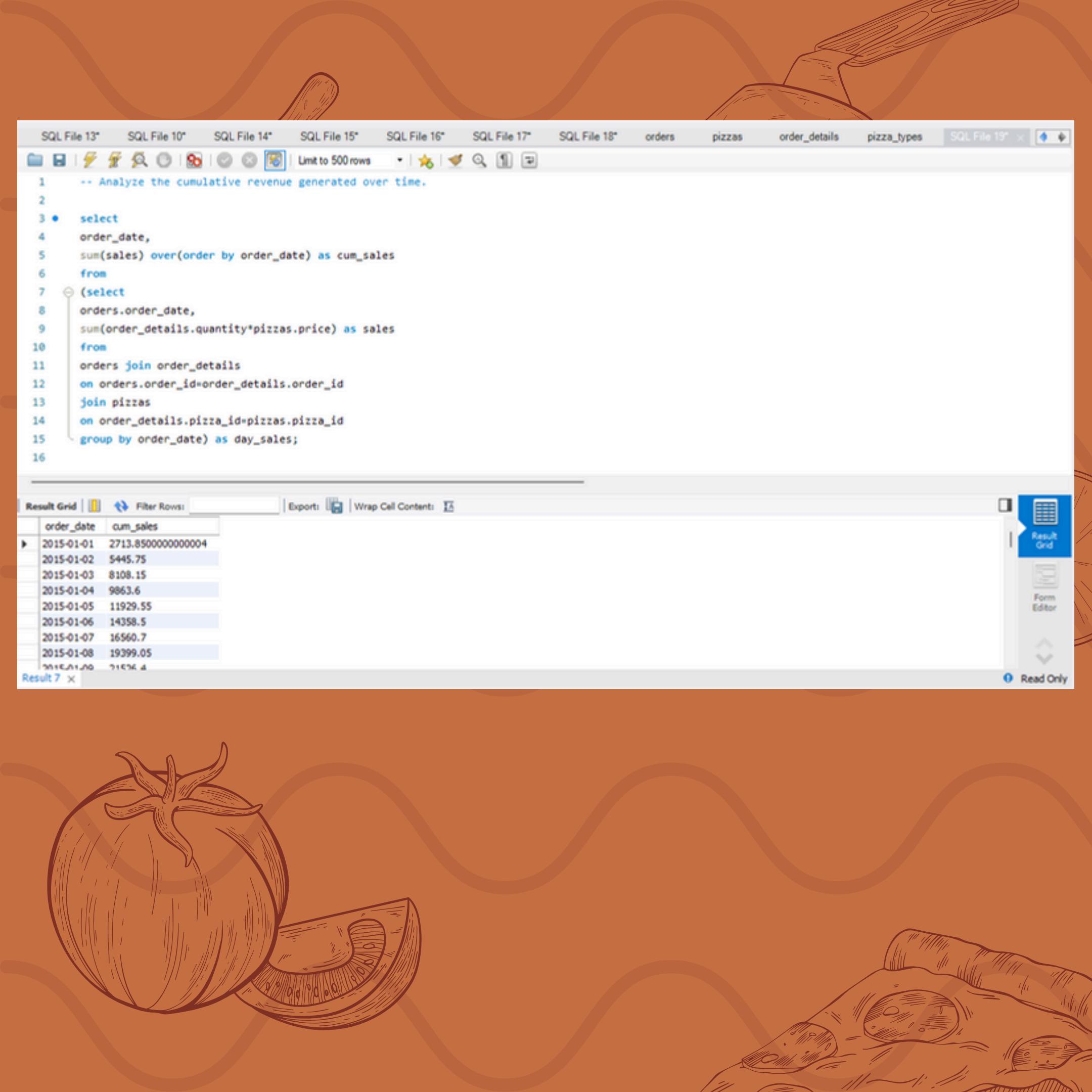
Limit to 500 rows

```
1 -- Calculate the percentage contribution of each pizza type to total revenue.
2
3 • select
4   pizza_types.name as pizza_name,
5   sum((pizzas.price*order_details.quantity)/817860.05)*100 as sales
6   from pizza_types
7   join pizzas
8   on pizza_types.pizza_type_id=pizzas.pizza_type_id
9   join order_details
10  on order_details.pizza_id=pizzas.pizza_id
11  group by pizza_types.name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Result 12 | Read Only

pizza_name	sales
The Hawaiian Pizza	3.9460602092007817
The Classic Deluxe Pizza	4.668341484585189
The Five Cheese Pizza	3.1871589766489445
The Italian Supreme Pizza	4.093212524563356
The Mexicana Pizza	3.2744905439505936

Output



SQL File 13* SQL File 10* SQL File 14* SQL File 15* SQL File 16* SQL File 17* SQL File 18* orders pizzas order_details pizza_types SQL File 19* 

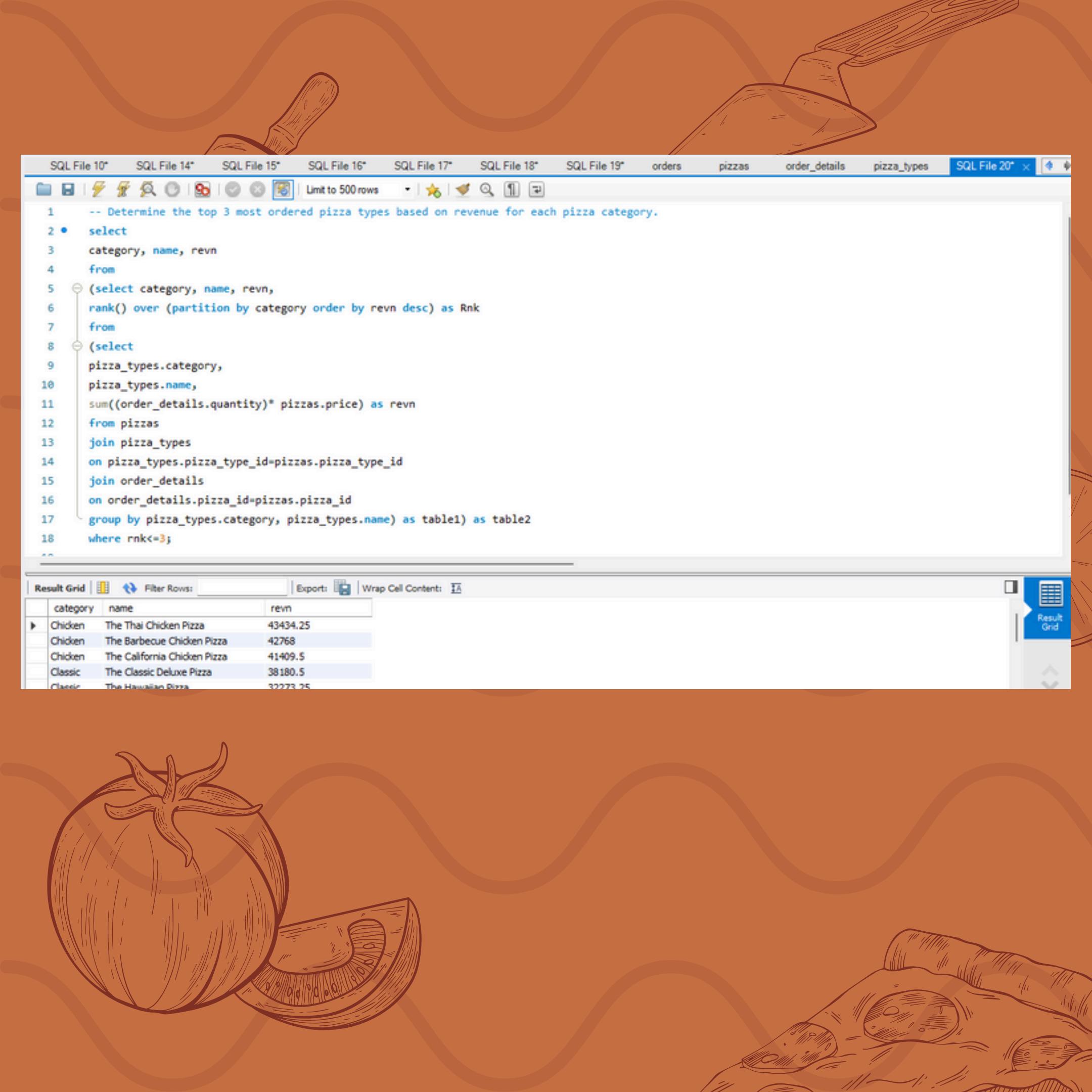
Limit to 500 rows   

```
1  -- Analyze the cumulative revenue generated over time.
2
3 • select
4   order_date,
5   sum(sales) over(order by order_date) as cum_sales
6   from
7   (select
8     orders.order_date,
9     sum(order_details.quantity*pizzas.price) as sales
10    from
11   orders join order_details
12   on orders.order_id=order_details.order_id
13   join pizzas
14   on order_details.pizza_id=pizzas.pizza_id
15   group by order_date) as day_sales;
16
```

Result Grid | Filter Rows! | Export | Wrap Cell Content:  

order_date	cum_sales
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	21594.4

Result 7 X Read Only



SQL File 10* SQL File 14* SQL File 15* SQL File 16* SQL File 17* SQL File 18* SQL File 19* orders pizzas order_details pizza_types SQL File 20* ×

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

2 * select

3 category, name, revn

4 from

5 (select category, name, revn,

6 rank() over (partition by category order by revn desc) as Rnk

7 from

8 (select

9 pizza_types.category,

10 pizza_types.name,

11 sum((order_details.quantity)* pizzas.price) as revn

12 from pizzas

13 join pizza_types

14 on pizza_types.pizza_type_id=pizzas.pizza_type_id

15 join order_details

16 on order_details.pizza_id=pizzas.pizza_id

17 group by pizza_types.category, pizza_types.name) as table1) as table2

18 where rnk<=3;

--

Result Grid | Filter Rows: _____ | Export: | Wrap Cell Content: | Result Grid

category	name	revn
Chicken	The Thai Chicken Pizza	43434.25
Chicken	The Barbecue Chicken Pizza	42768
Chicken	The California Chicken Pizza	41409.5
Classic	The Classic Deluxe Pizza	38180.5
Classic	The Hawaiian Pizza	32273.25



A detailed line drawing background features a rolling pin on the left, a pizza cutter on the right, a slice of pizza at the top, a tomato and a slice of cheese below it, and a hand holding a pizza at the bottom right.

THANK YOU

PLEASE LEAVE A FEEDBACK @
nehareddy1004@gmail.com

Why was the pizza happy? It was feeling cheesy!