



SQL PROJECT ON ***PIZZA*** ***SALES***



Contact Us:





HELLO

I'm subhash In this project,
I utilized SQL queries to solve the
questions that where related to
piza sales

CONTACT

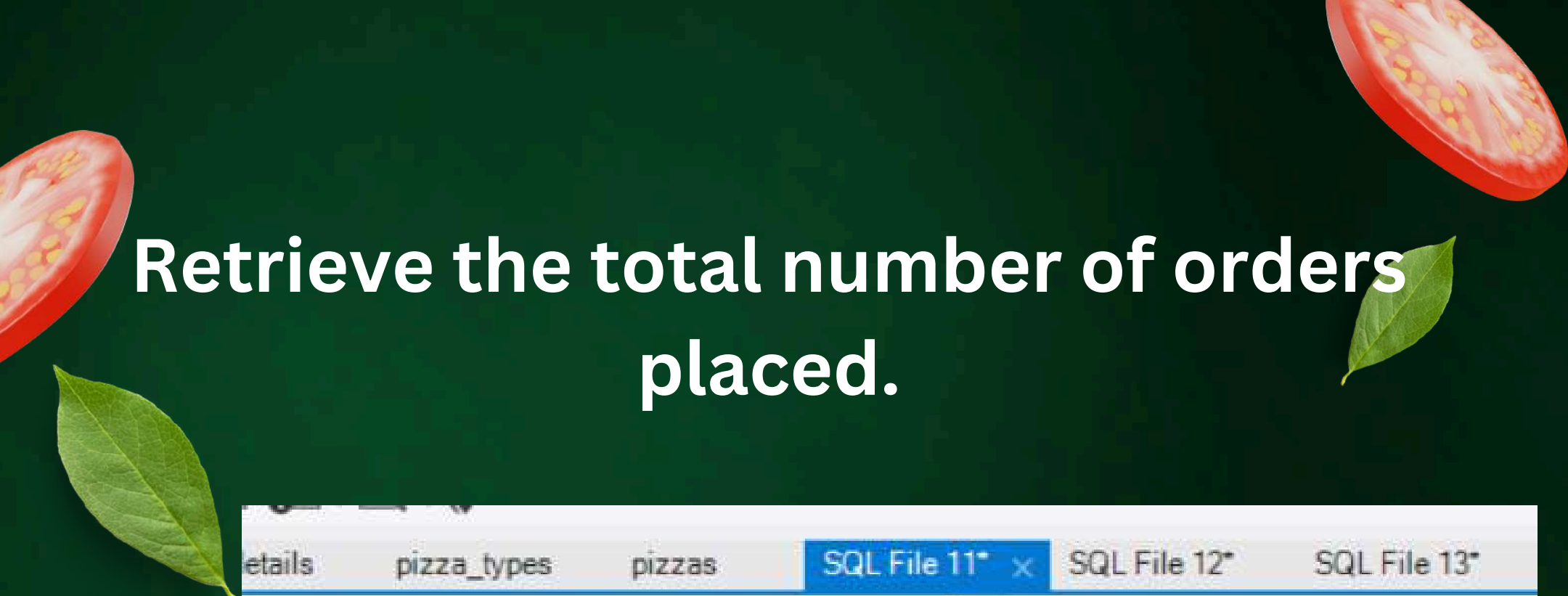


9481612592

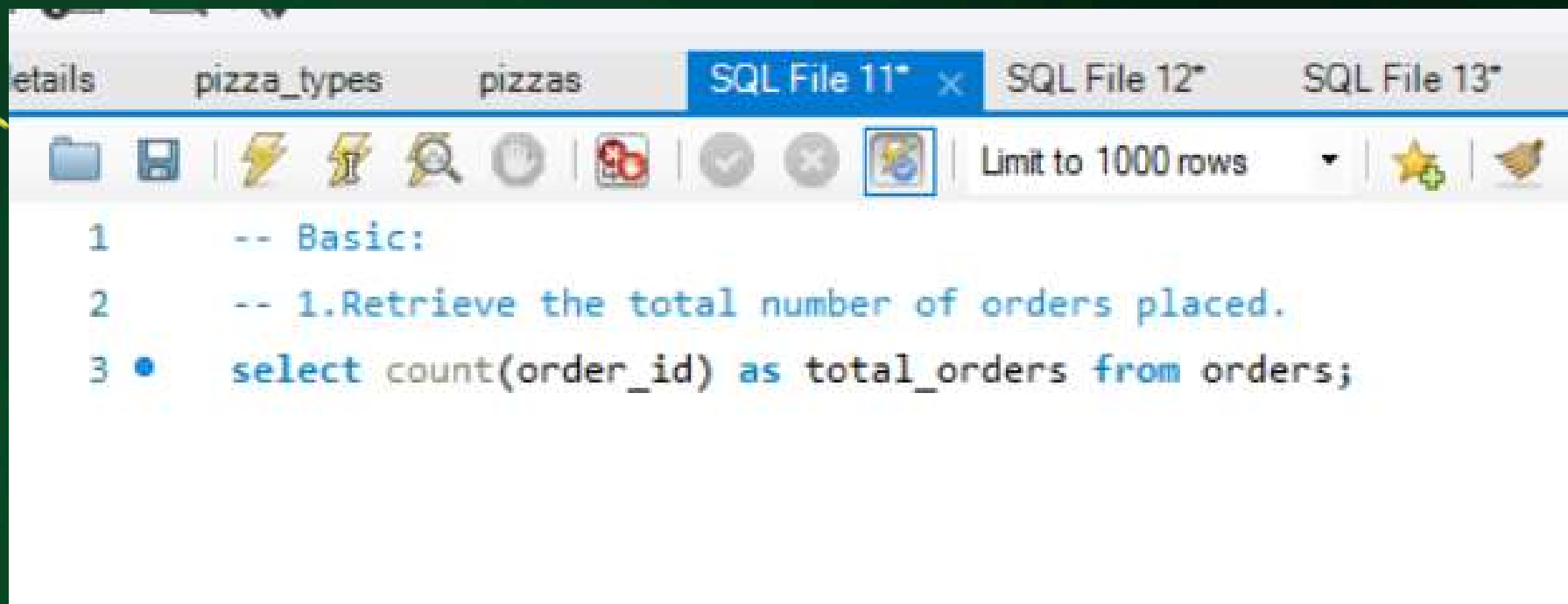


[https://www.linkedin.com/i
n/subhash-babu-vn-
235b6b236](https://www.linkedin.com/in/subhash-babu-vn-235b6b236)





Retrieve the total number of orders placed.

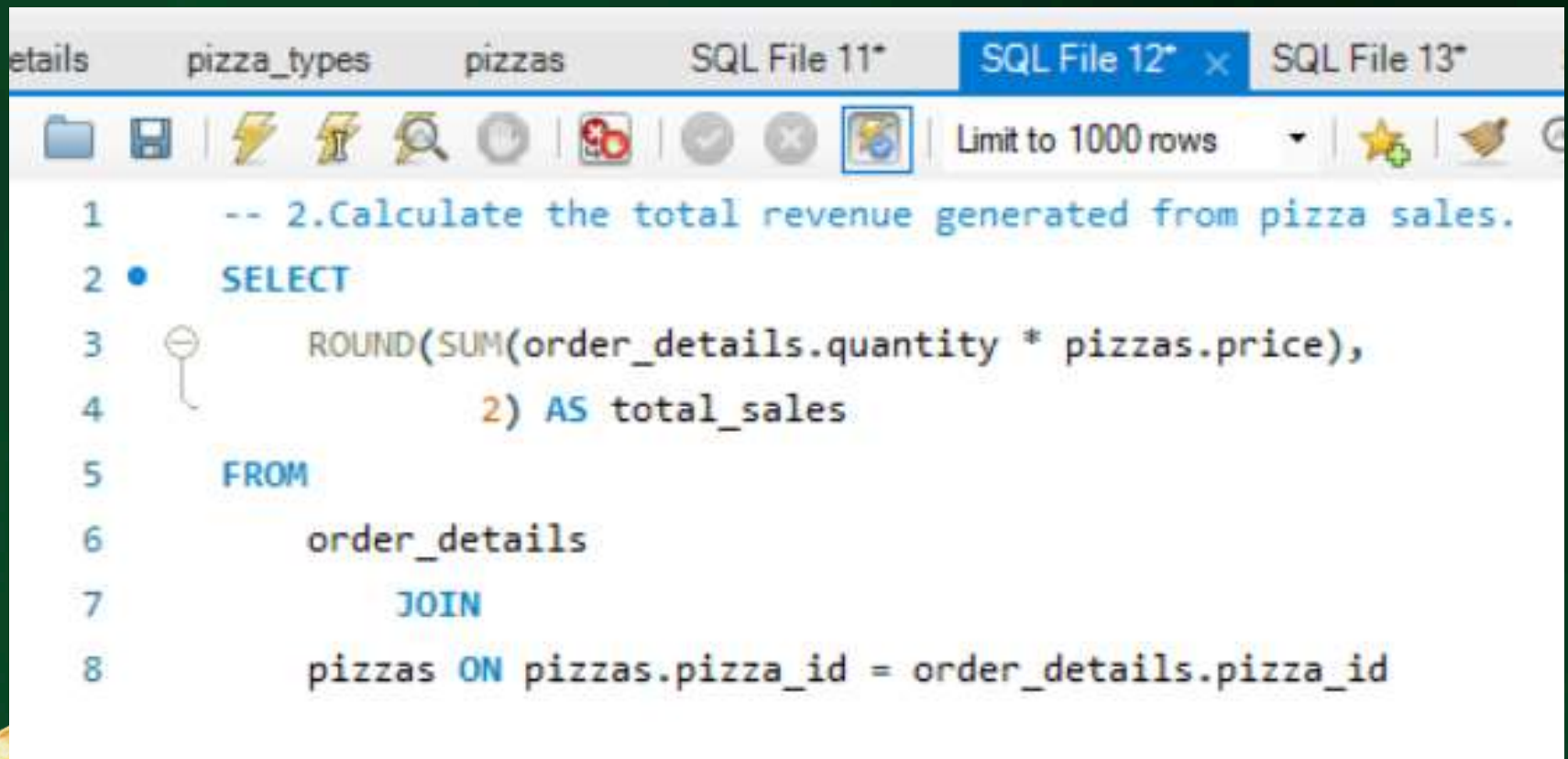


```
1  -- Basic:
2  -- 1.Retrieve the total number of orders placed.
3  • select count(order_id) as total_orders from orders;
```

The screenshot shows a SQL IDE interface with tabs for 'details', 'pizza_types', 'pizzas', 'SQL File 11*', 'SQL File 12*', and 'SQL File 13*'. The 'SQL File 11*' tab is active, displaying the SQL query. The toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown.



Calculate the total revenue generated from pizza sales.

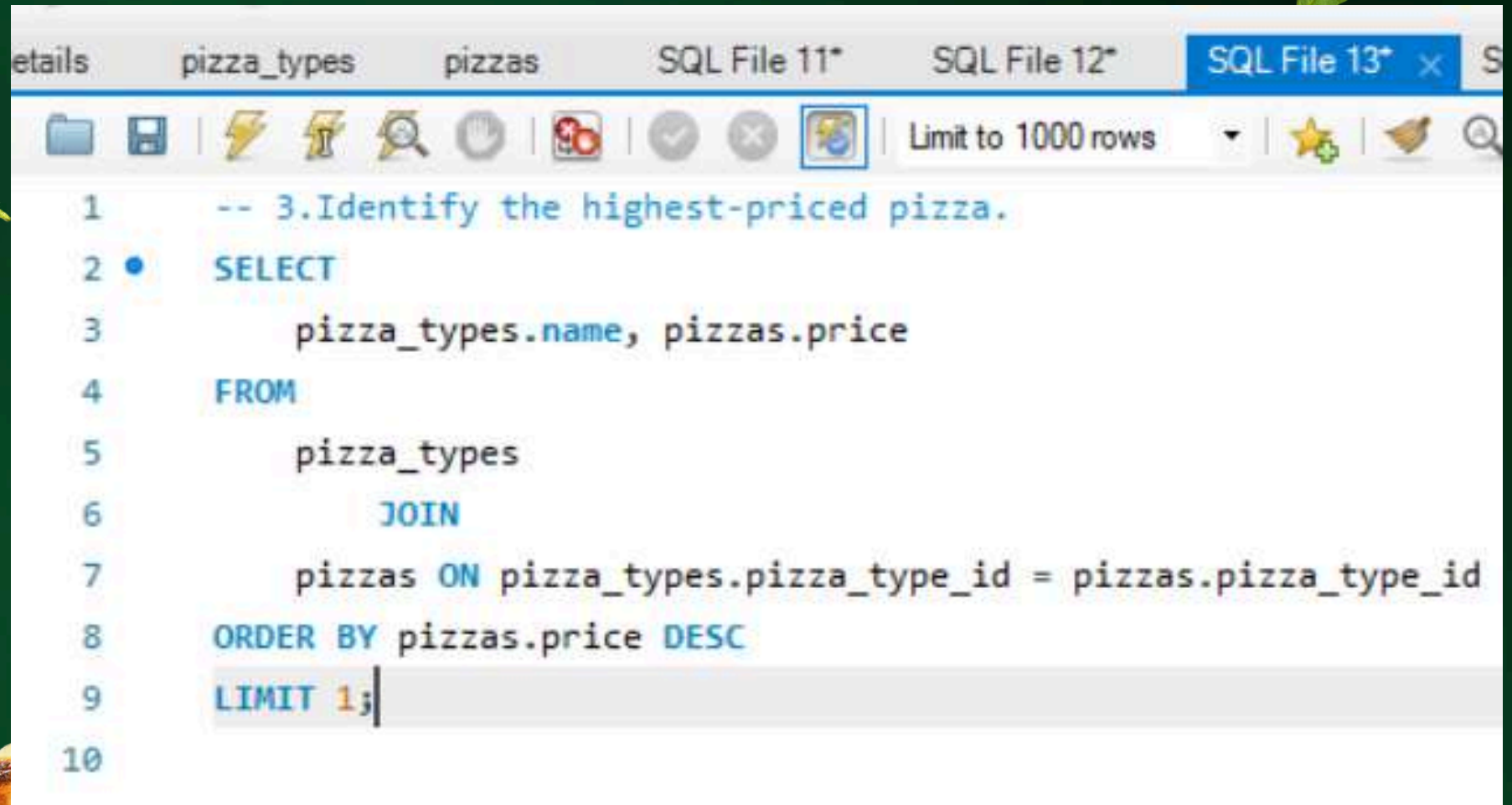


The screenshot shows a SQL IDE window with multiple tabs: 'details', 'pizza_types', 'pizzas', 'SQL File 11*', 'SQL File 12*' (selected), and 'SQL File 13*'. The toolbar includes icons for file operations, a search icon, and a 'Limit to 1000 rows' dropdown. The SQL query is as follows:

```
1  -- 2.Calculate the total revenue generated from pizza sales.
2  SELECT
3      ROUND(SUM(order_details.quantity * pizzas.price),
4             2) AS total_sales
5  FROM
6      order_details
7      JOIN
8      pizzas ON pizzas.pizza_id = order_details.pizza_id
```



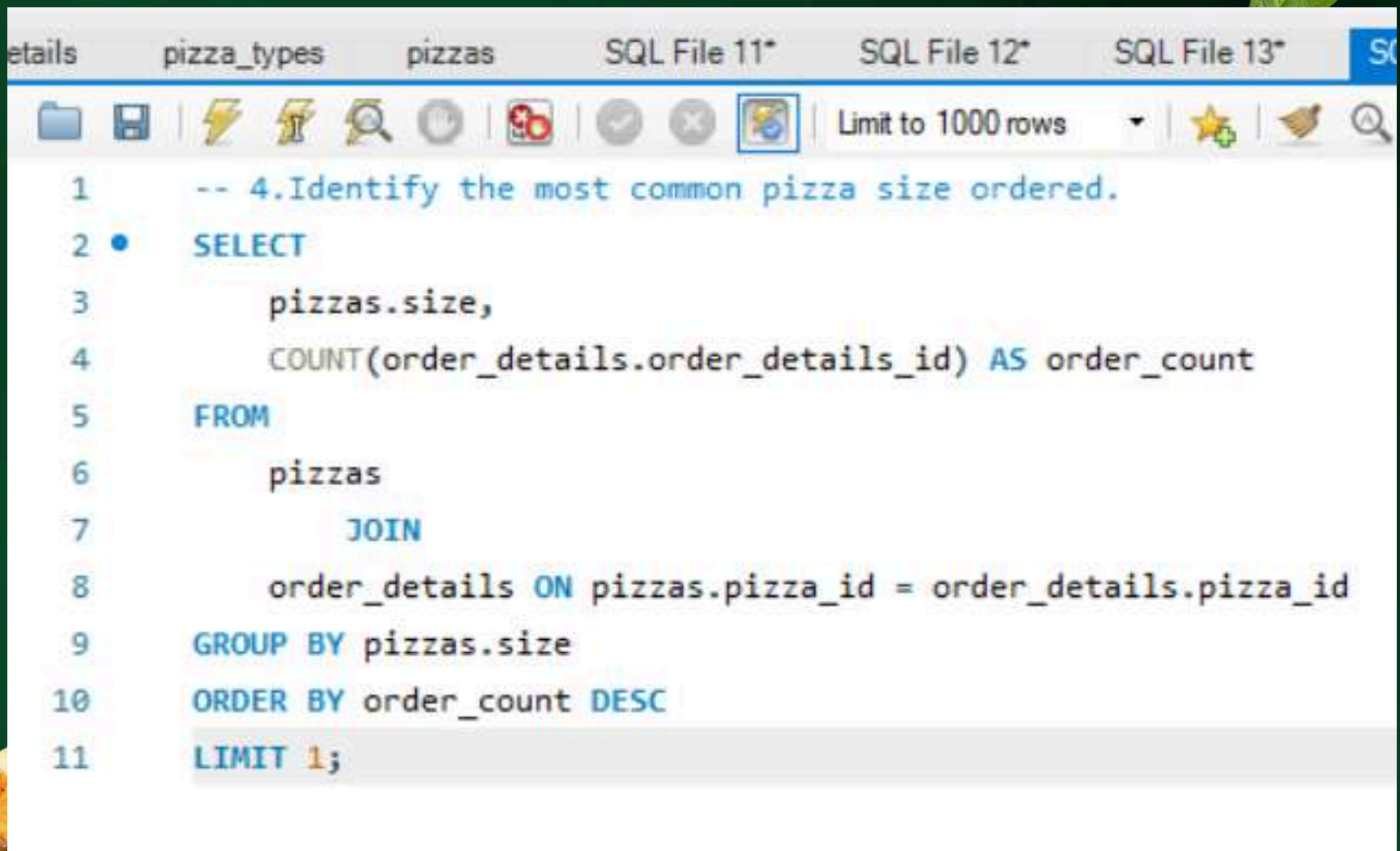

Identify the highest-priced pizza.



```
1  -- 3. Identify the highest-priced pizza.
2  SELECT
3      pizza_types.name, pizzas.price
4  FROM
5      pizza_types
6      JOIN
7      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
8  ORDER BY pizzas.price DESC
9  LIMIT 1;
10
```



Identify the most common pizza size ordered.



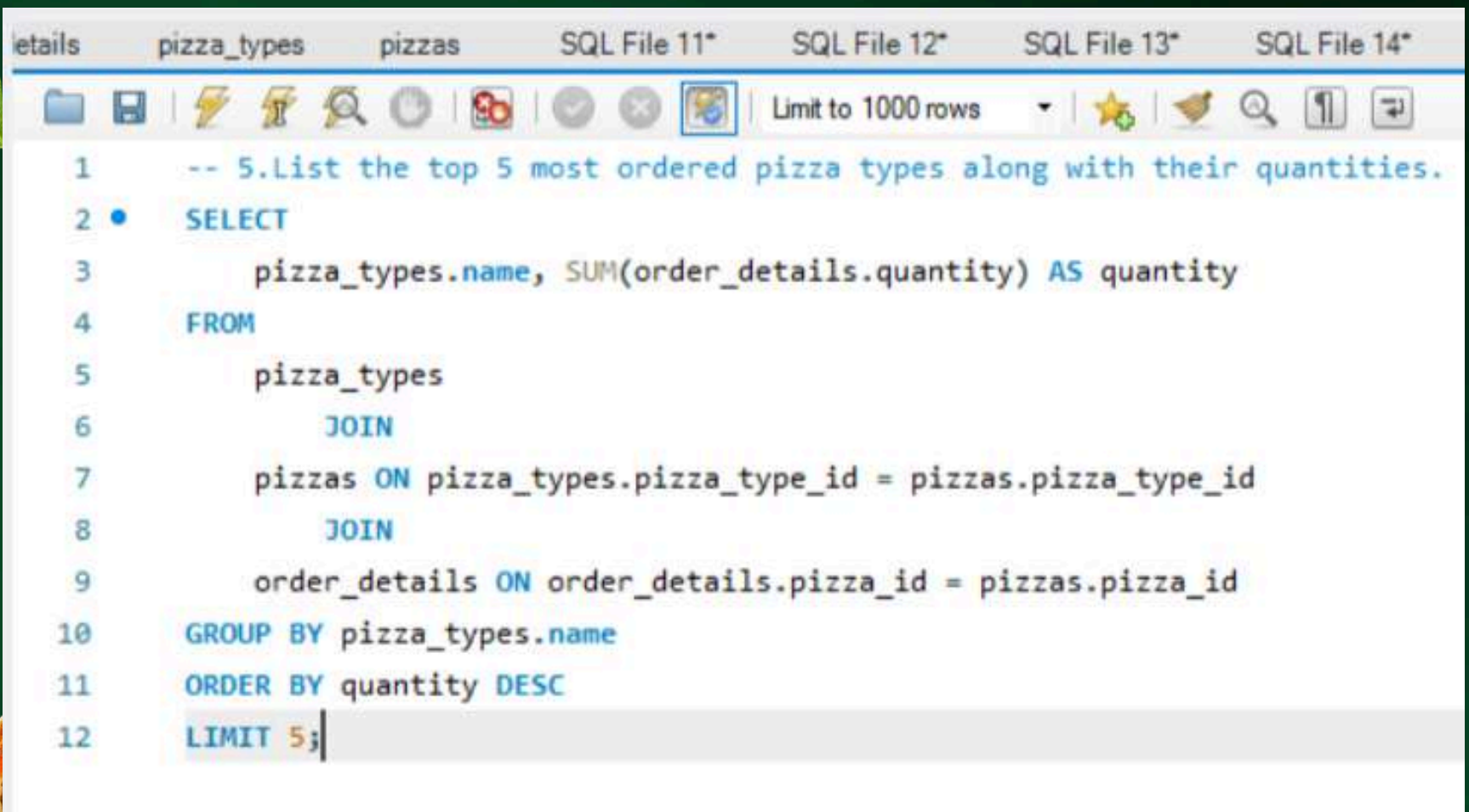
The screenshot shows a SQL IDE window with multiple tabs: 'details', 'pizza_types', 'pizzas', 'SQL File 11*', 'SQL File 12*', and 'SQL File 13*'. The 'pizzas' tab is active. The query editor displays the following SQL code:

```
1  -- 4. Identify the most common pizza size ordered.
2  •  SELECT
3      pizzas.size,
4      COUNT(order_details.order_details_id) AS order_count
5  FROM
6      pizzas
7      JOIN
8      order_details ON pizzas.pizza_id = order_details.pizza_id
9  GROUP BY pizzas.size
10 ORDER BY order_count DESC
11 LIMIT 1;
```

The query is designed to find the most common pizza size by joining the 'pizzas' table with the 'order_details' table, grouping by size, and ordering by the count of orders in descending order, limiting the result to one row.

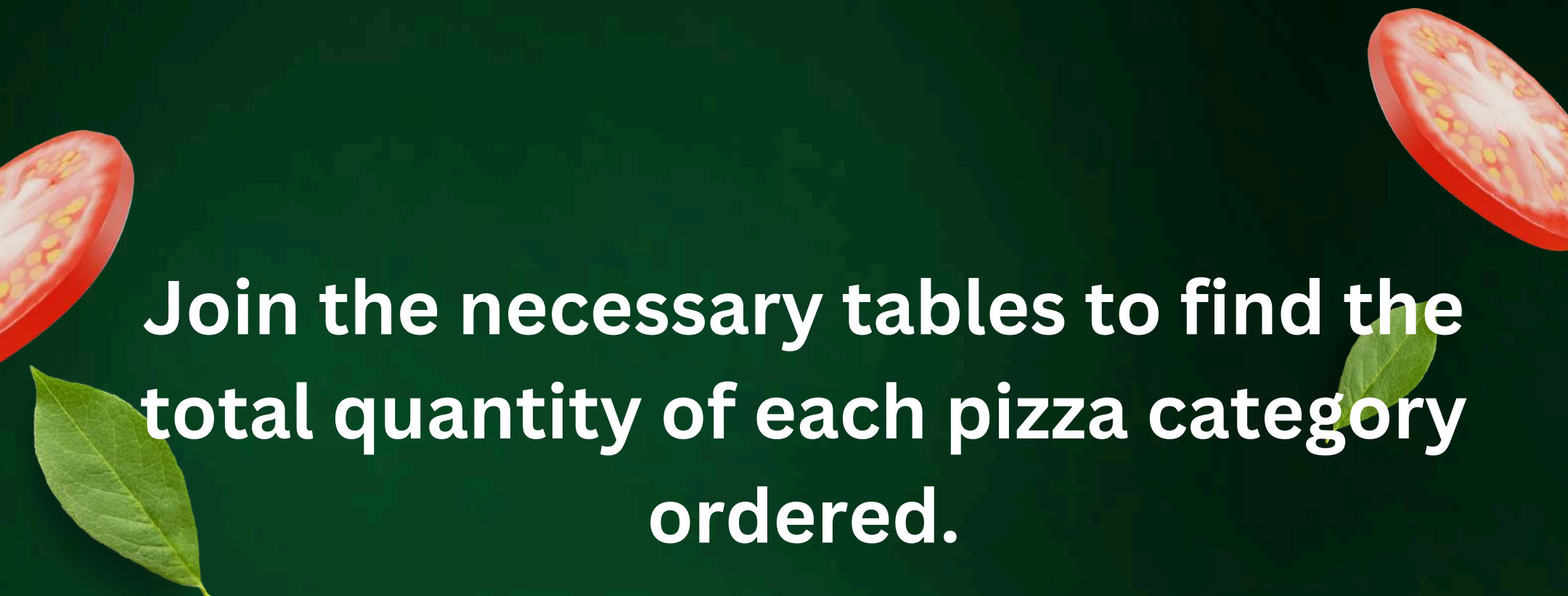


List the top 5 most ordered pizza types along with their quantities.

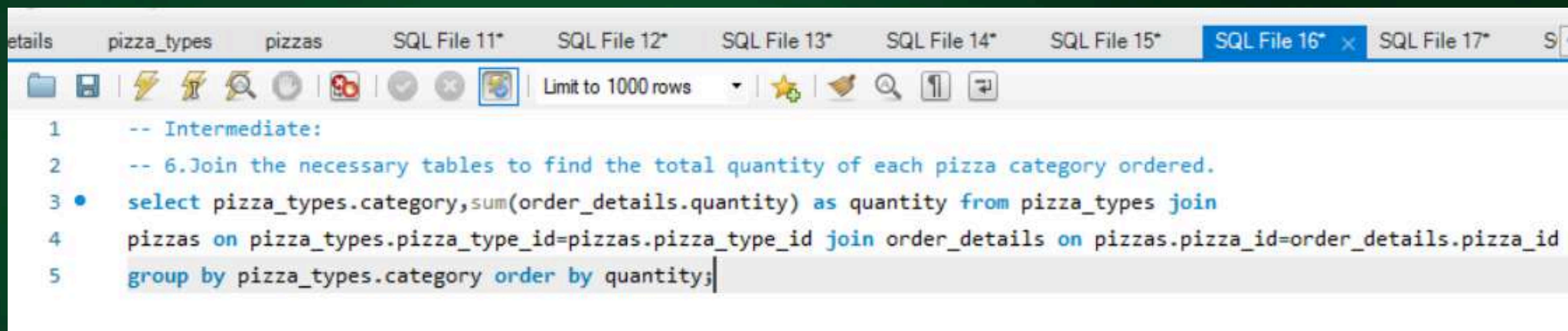


```
1  -- 5.List the top 5 most ordered pizza types along with their quantities.
2  •  SELECT
3      pizza_types.name, SUM(order_details.quantity) AS quantity
4  FROM
5      pizza_types
6      JOIN
7      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
8      JOIN
9      order_details ON order_details.pizza_id = pizzas.pizza_id
10 GROUP BY pizza_types.name
11 ORDER BY quantity DESC
12 LIMIT 5;
```





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

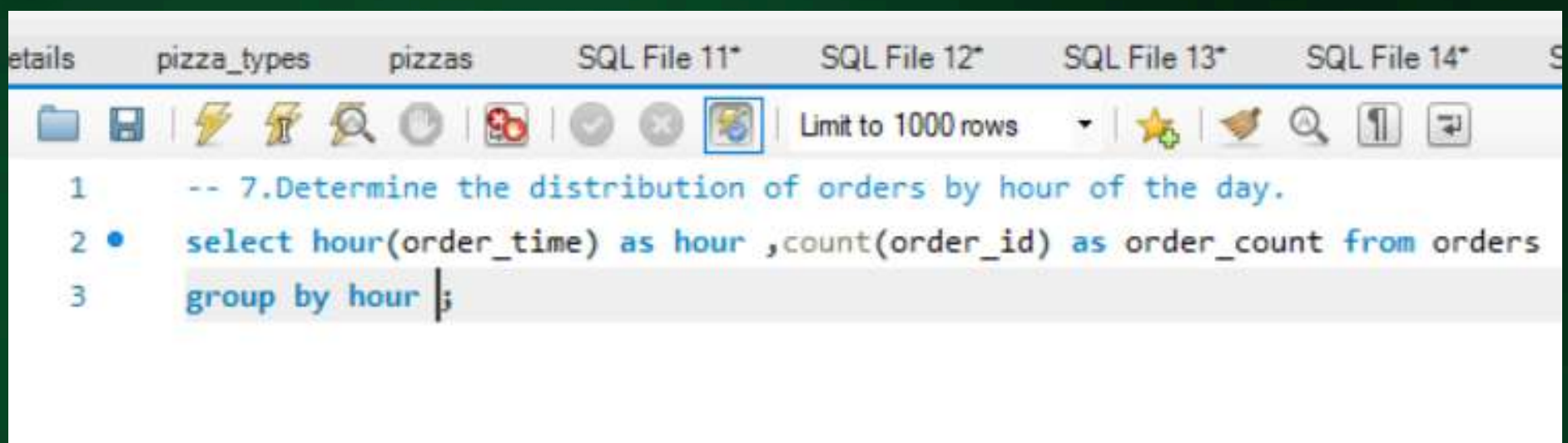


```
1  -- Intermediate:
2  -- 6.Join the necessary tables to find the total quantity of each pizza category ordered.
3  • select pizza_types.category,sum(order_details.quantity) as quantity from pizza_types join
4  pizzas on pizza_types.pizza_type_id=pizzas.pizza_type_id join order_details on pizzas.pizza_id=order_details.pizza_id
5  group by pizza_types.category order by quantity;
```





Determine the distribution of orders
by hour of the day.

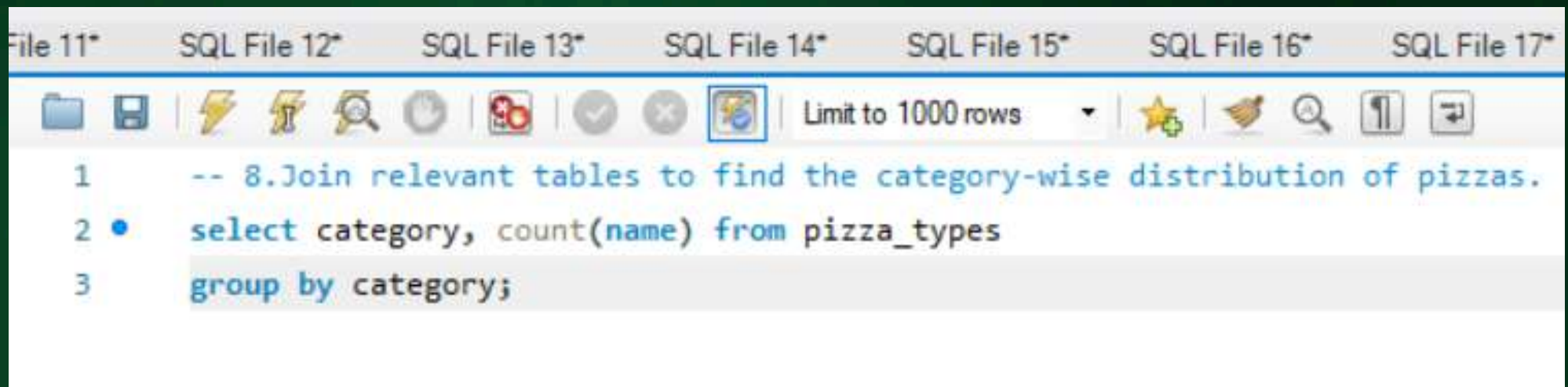


```
1  -- 7.Determine the distribution of orders by hour of the day.
2  • select hour(order_time) as hour ,count(order_id) as order_count from orders
3  group by hour ;
```



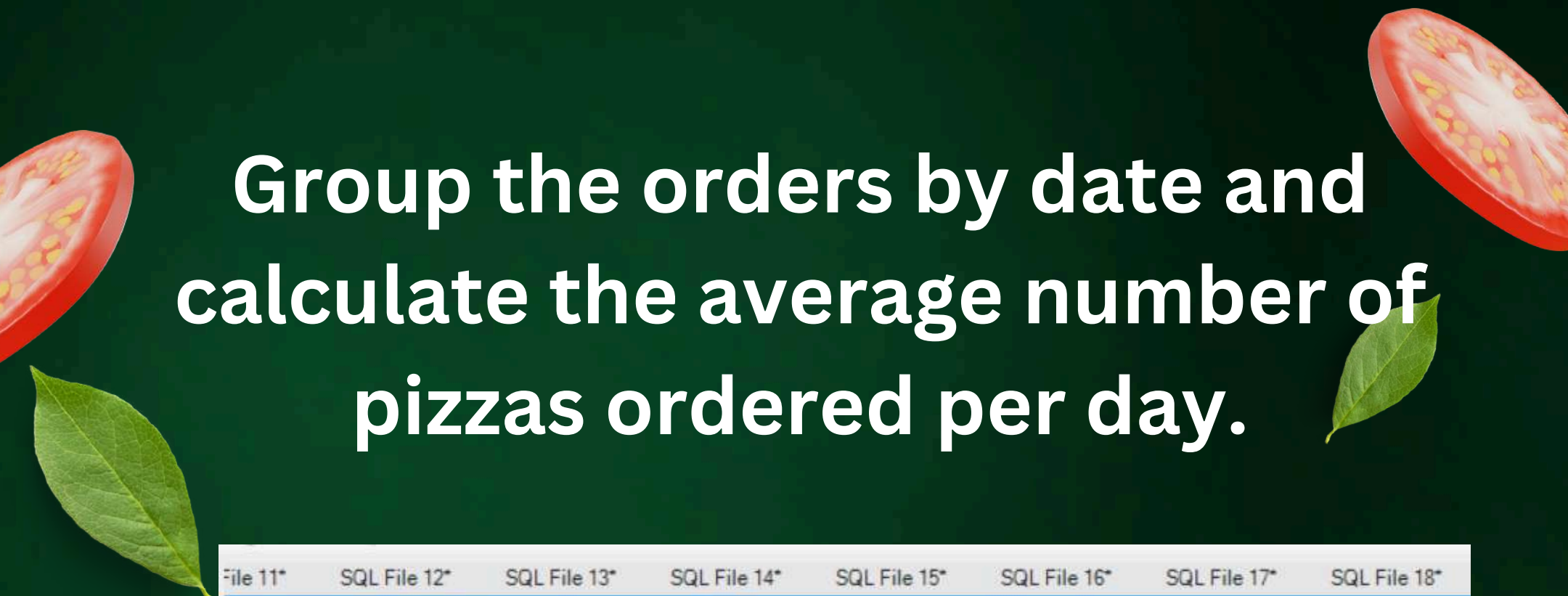


Join relevant tables to find the category-wise distribution of pizzas.

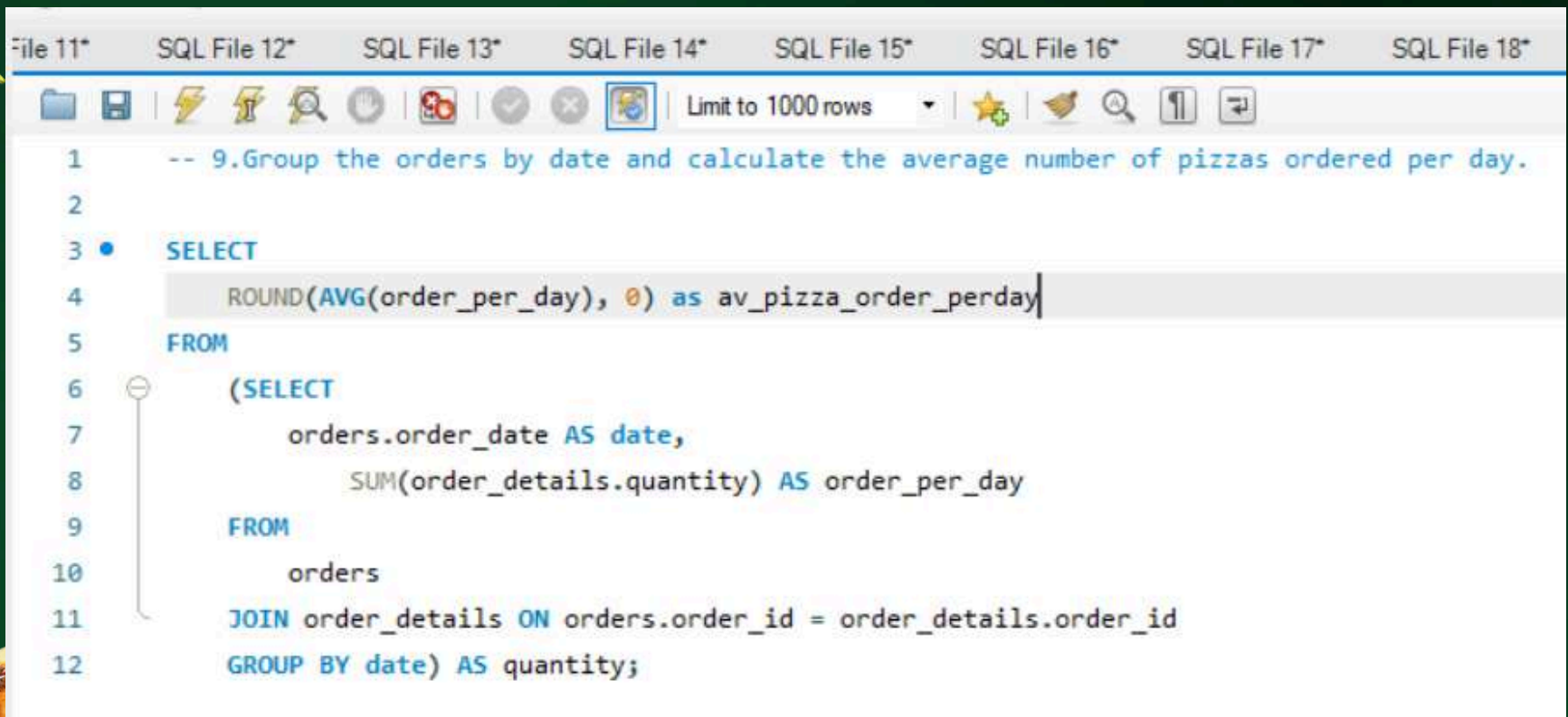


```
File 11*  SQL File 12*  SQL File 13*  SQL File 14*  SQL File 15*  SQL File 16*  SQL File 17*
[Icons] Limit to 1000 rows
1  -- 8.Join relevant tables to find the category-wise distribution of pizzas.
2  • select category, count(name) from pizza_types
3  group by category;
```



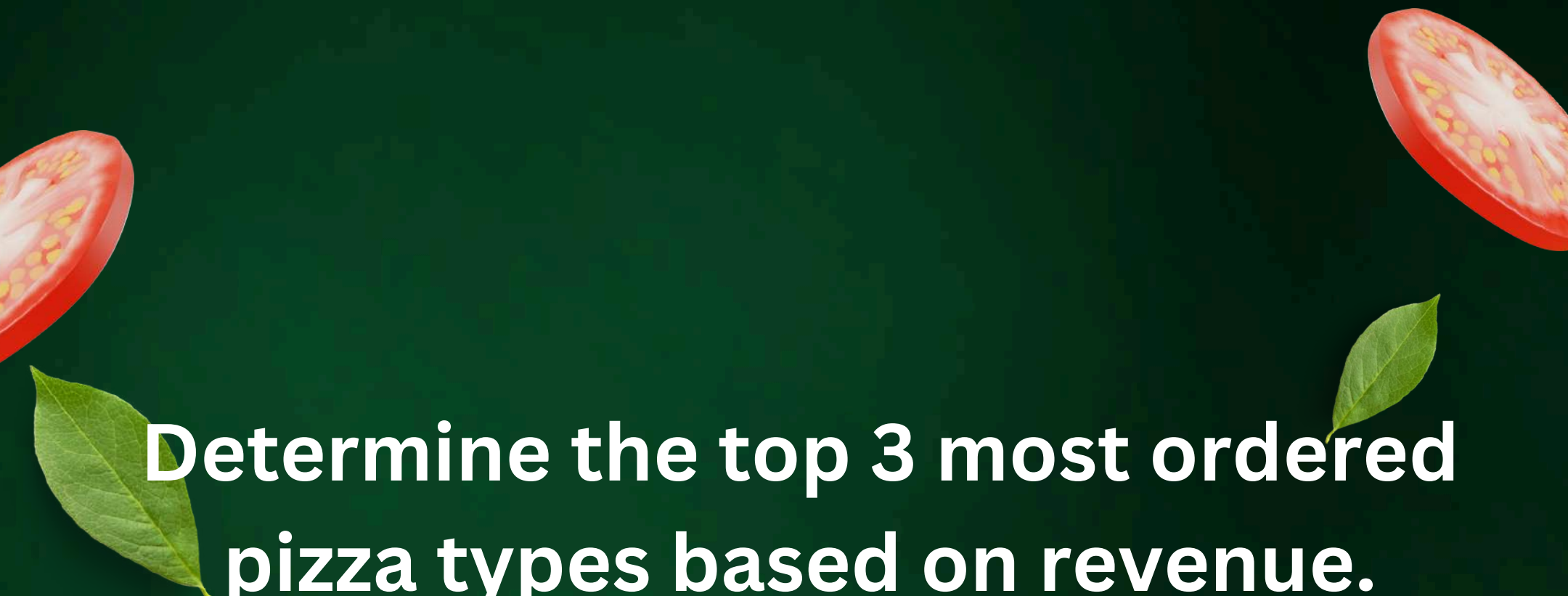


Group the orders by date and calculate the average number of pizzas ordered per day.

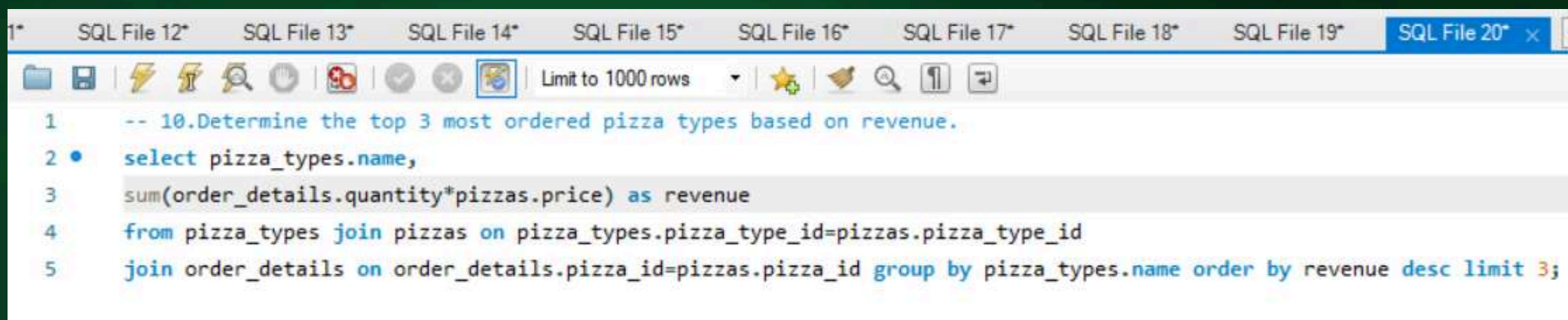


```
File 11*  SQL File 12*  SQL File 13*  SQL File 14*  SQL File 15*  SQL File 16*  SQL File 17*  SQL File 18*
Limit to 1000 rows
1  -- 9.Group the orders by date and calculate the average number of pizzas ordered per day.
2
3  SELECT
4  ROUND(AVG(order_per_day), 0) as av_pizza_order_perday
5  FROM
6  (SELECT
7   orders.order_date AS date,
8   SUM(order_details.quantity) AS order_per_day
9   FROM
10   orders
11   JOIN order_details ON orders.order_id = order_details.order_id
12   GROUP BY date) AS quantity;
```



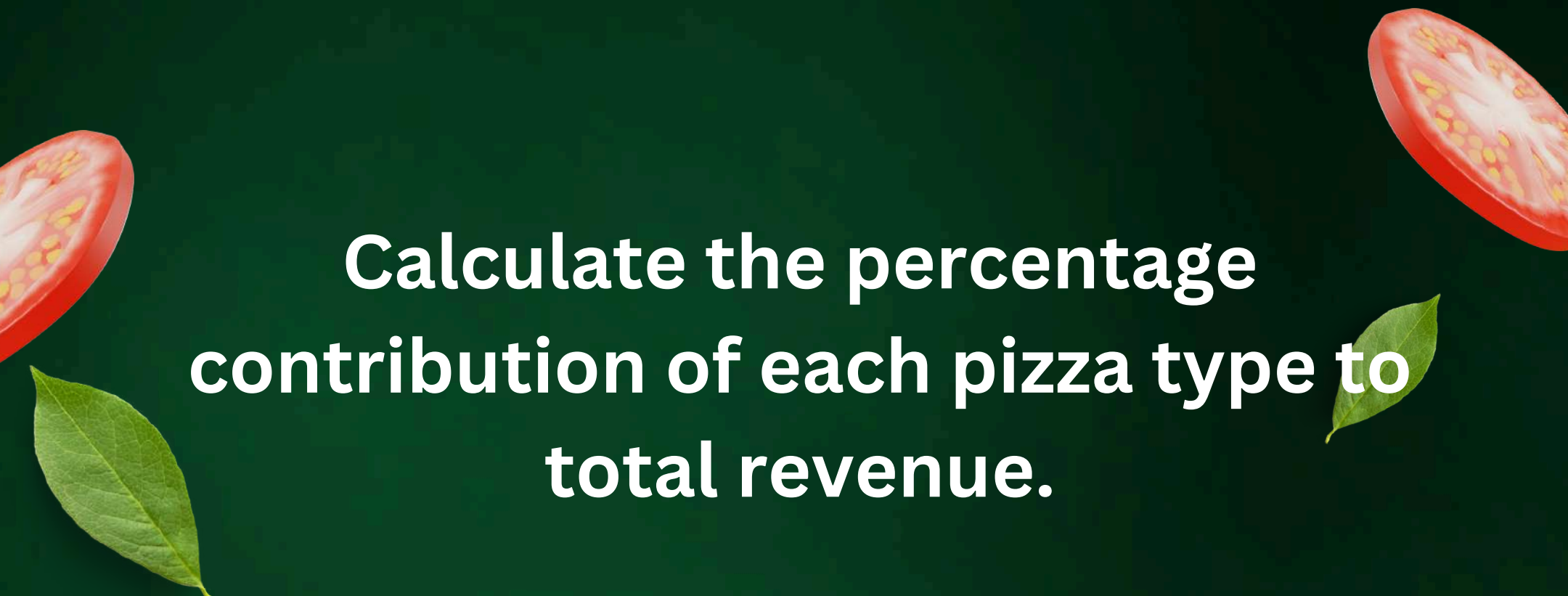


Determine the top 3 most ordered
pizza types based on revenue.

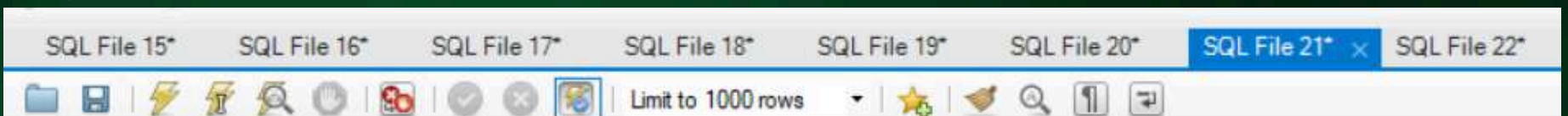


```
1  -- 10.Determine the top 3 most ordered pizza types based on revenue.
2  • select pizza_types.name,
3     sum(order_details.quantity*pizzas.price) as revenue
4  from pizza_types join pizzas on pizza_types.pizza_type_id=pizzas.pizza_type_id
5  join order_details on order_details.pizza_id=pizzas.pizza_id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





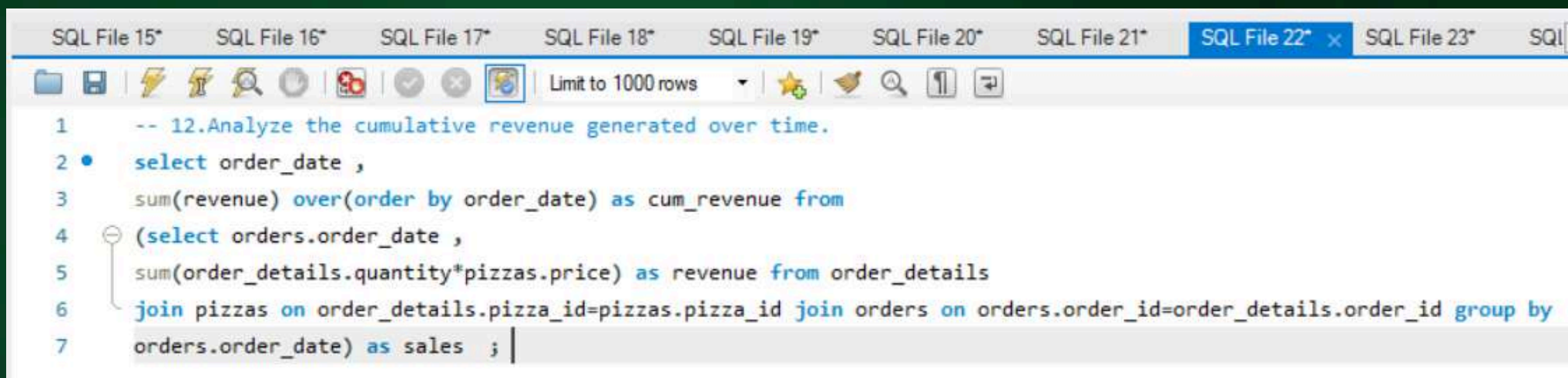
Calculate the percentage contribution of each pizza type to total revenue.



```
1  -- Advanced:
2  -- 11. Calculate the percentage contribution of each pizza type to total revenue.
3  • select pizza_types.category as pizza,
4     round((sum(order_details.quantity*pizzas.price)/(SELECT
5         ROUND(SUM(order_details.quantity * pizzas.price),
6             2) AS total_sales
7     FROM
8         order_details
9     JOIN
10         pizzas ON pizzas.pizza_id = order_details.pizza_id))*100,2) as revenue from
11     pizza_types join pizzas on pizza_types.pizza_type_id=pizzas.pizza_type_id
12     join order_details on order_details.pizza_id=pizzas.pizza_id group by pizza order by revenue desc;
```



Analyze the cumulative revenue generated over time.



The screenshot shows a SQL IDE with multiple tabs labeled 'SQL File 15*' through 'SQL File 23*'. The active tab is 'SQL File 22*'. The query editor contains the following SQL code:

```
1  -- 12.Analyze the cumulative revenue generated over time.
2  • select order_date ,
3     sum(revenue) over(order by order_date) as cum_revenue from
4  (select orders.order_date ,
5     sum(order_details.quantity*pizzas.price) as revenue from order_details
6   join pizzas on order_details.pizza_id=pizzas.pizza_id join orders on orders.order_id=order_details.order_id group by
7   orders.order_date) as sales ; |
```

The IDE interface includes a toolbar with icons for file operations, a 'Limit to 1000 rows' dropdown, and a search icon.

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

```
SQL File 15*  SQL File 16*  SQL File 17*  SQL File 18*  SQL File 19*  SQL File 20*  SQL File 21*  SQL File 22*
Limit to 1000 rows
1  -- 13.Determine the top 3 most ordered pizza types based on revenue for each pizza category.
2  • select name,category,revenue from
3  (select category,name,revenue,
4  rank() over(partition by category order by revenue desc) as rn
5  from
6  (select pizza_types.category,pizza_types.name ,
7  sum((order_details.quantity) *pizzas.price) as revenue
8  from pizza_types join pizzas on pizza_types.pizza_type_id=pizzas.pizza_type_id join order_details on
9  order_details.pizza_id=pizzas.pizza_id group by pizza_types.category,pizza_types.name) as a) as b
10 where rn<=3;
```

