

Domino's Pizza Sales SQL Analysis

An Interactive Data Exploration Report
Made by Nandini



Table of Contents

Q1. Retrieve the total number of orders placed.

Q2. Calculate the total revenue generated from pizza sales.

Q3. Identify the highest-priced pizza.

Q4. Identify the most common pizza size ordered.

Q5. List the top 5 most ordered pizza types along with their quantities.

Q6. Total quantity ordered per pizza type.

Q7. Determine the distribution of orders by hour of the day.

Q8. Category-wise distribution of pizzas.

Q9. Average number of pizzas ordered per day.

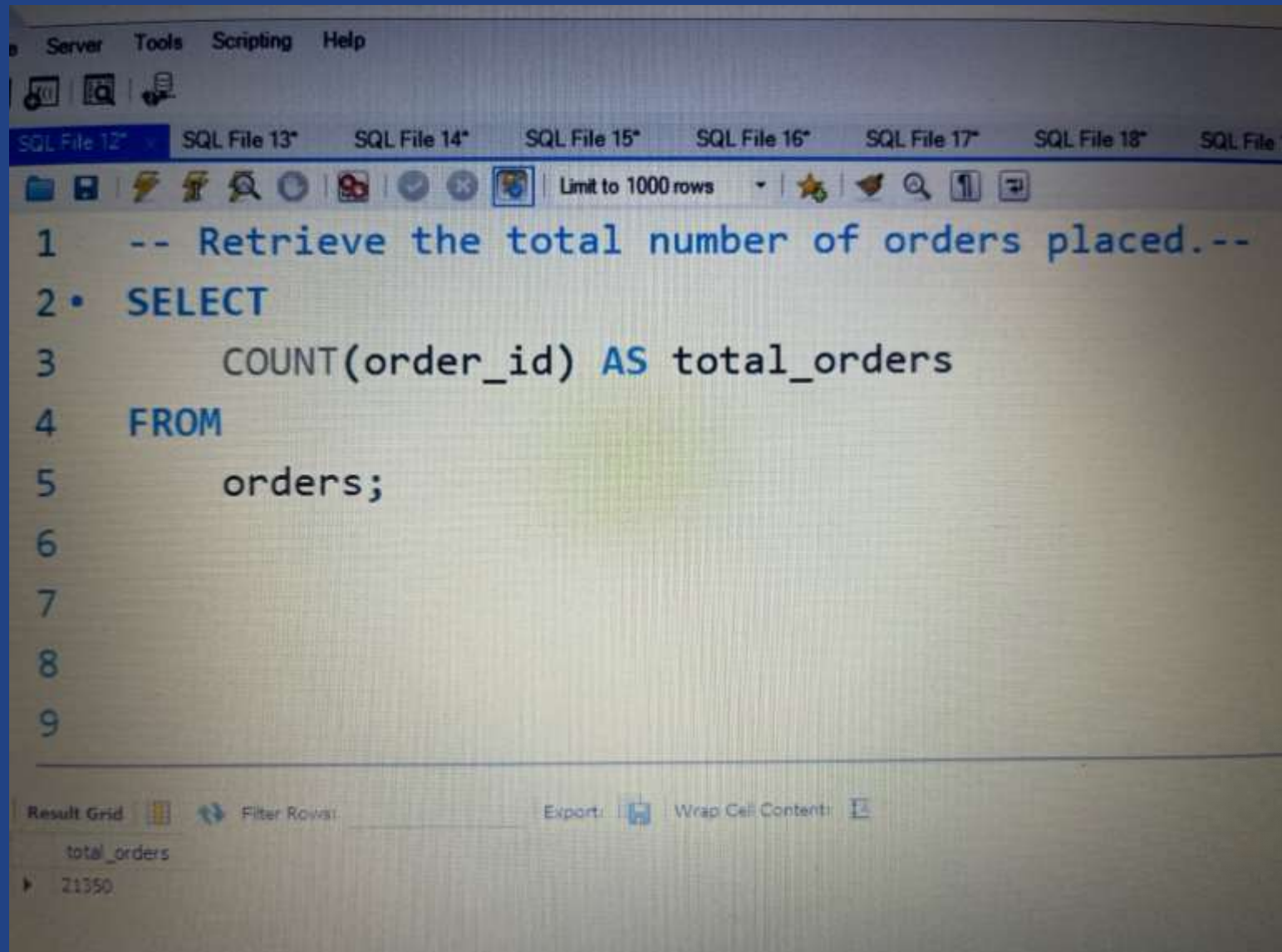
Q10. Top 3 pizza types by revenue.

Q11. Calculate the percentage contribution of each pizza type to total revenue.

Q12. Analyze the cumulative revenue generated over time

Q13. Determine the top 3 most ordered pizza types based on revenue for each pizza category.

Q1. Retrieve the total number of orders placed.

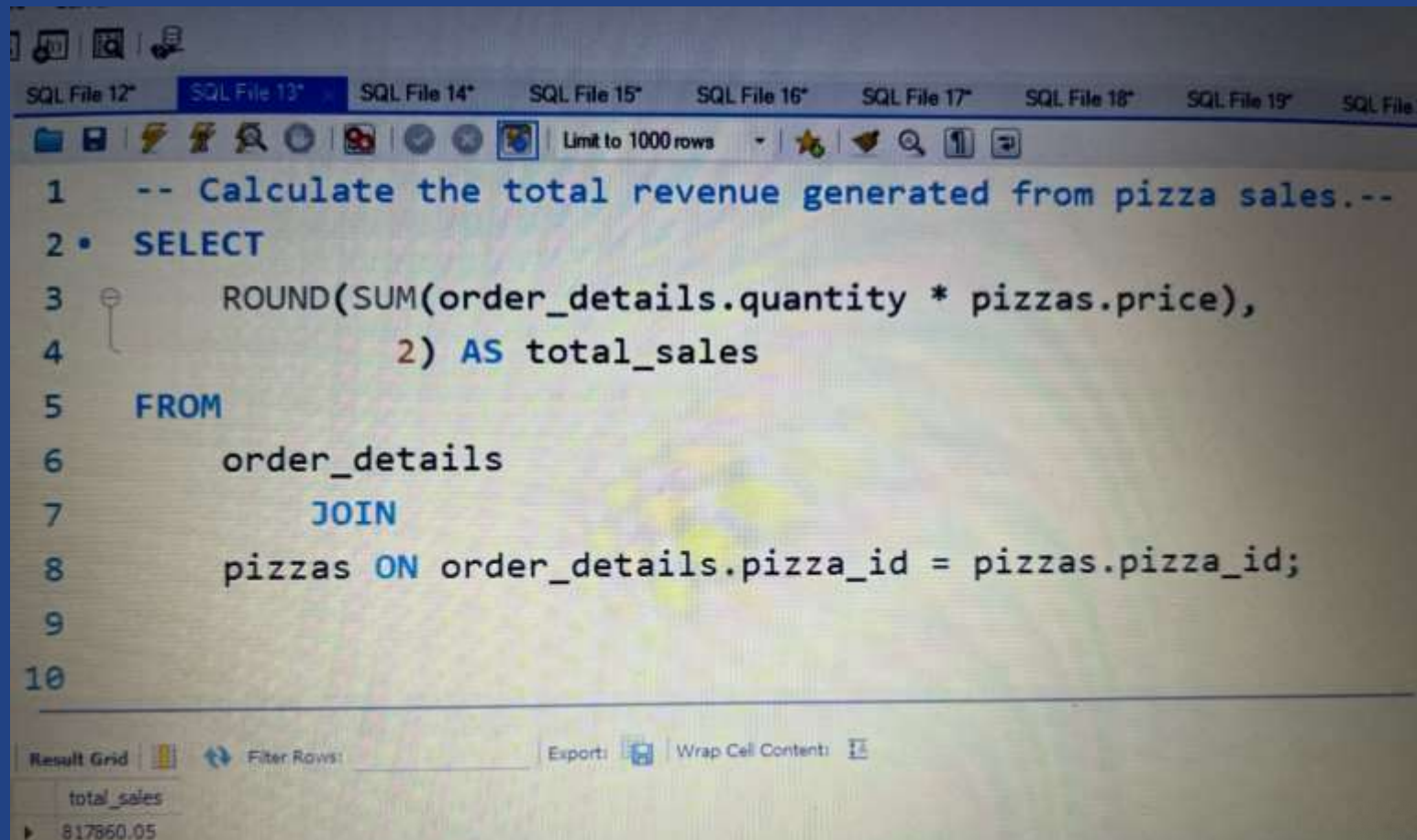


The screenshot shows a SQL IDE window with multiple tabs labeled 'SQL File 12*' through 'SQL File 18*'. The active tab is 'SQL File 12*'. The menu bar includes 'Server', 'Tools', 'Scripting', and 'Help'. The toolbar contains various icons, including a 'Limit to 1000 rows' dropdown. The SQL editor displays the following query:

```
1  -- Retrieve the total number of orders placed.--
2  • SELECT
3      COUNT(order_id) AS total_orders
4  FROM
5      orders;
6
7
8
9
```

At the bottom, the 'Result Grid' is visible, showing a single column header 'total_orders' and a single row with the value '21350'. The 'Filter Rows' button is also visible.

Q2. Calculate the total revenue generated from pizza sales.



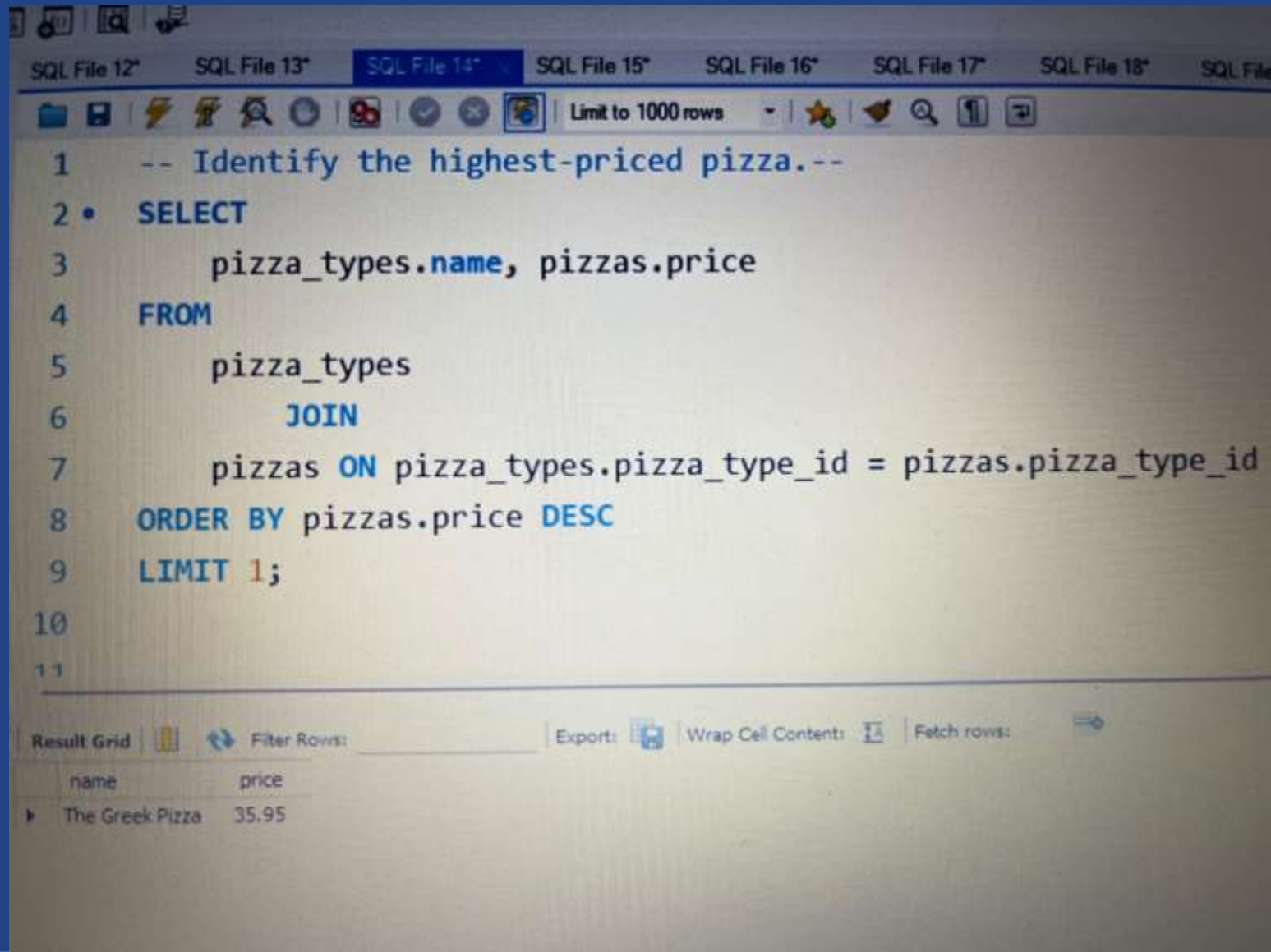
The screenshot shows a SQL IDE interface with multiple tabs labeled 'SQL File 12*' through 'SQL File 19*'. The active tab is 'SQL File 13*'. The query editor contains the following SQL code:

```
1  -- 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 order_details.pizza_id = pizzas.pizza_id;
9
10
```

Below the query editor, the 'Result Grid' is visible, showing a single column named 'total_sales' with a value of 817860.05.

total_sales
817860.05

Q3. Identify the highest-priced pizza.



The screenshot shows a SQL IDE with multiple tabs labeled 'SQL File 12*' through 'SQL File 18*'. The active tab is 'SQL File 14*'. The query editor contains the following SQL code:

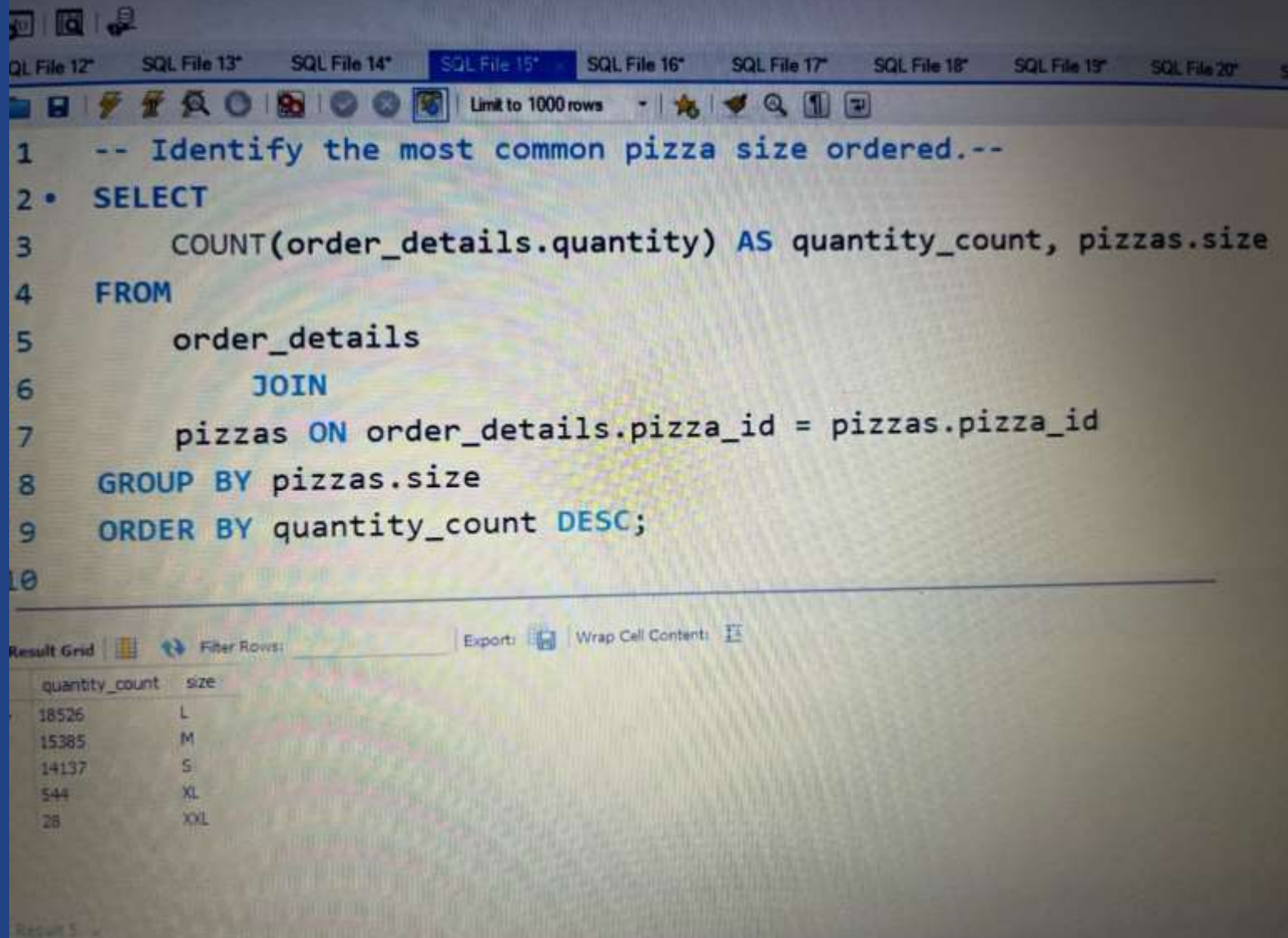
```
1  -- Identify the highest-priced pizza.--
2  • SELECT
3      pizza_types.name, pizzas.price
4  FROM
5      pizza_types
6      JOIN
7      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
8  ORDER BY pizzas.price DESC
9  LIMIT 1;
10
11
```

Below the query editor, the 'Result Grid' is visible, showing the results of the query. The first row is expanded, showing the name and price of the highest-priced pizza.

name	price
The Greek Pizza	35.95

In the bottom right corner, there is a logo for 'Domino's' featuring a red and blue diamond shape with a white 'D' and the word 'Domino's' in red text.

Q4. Identify the most common pizza size ordered.



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

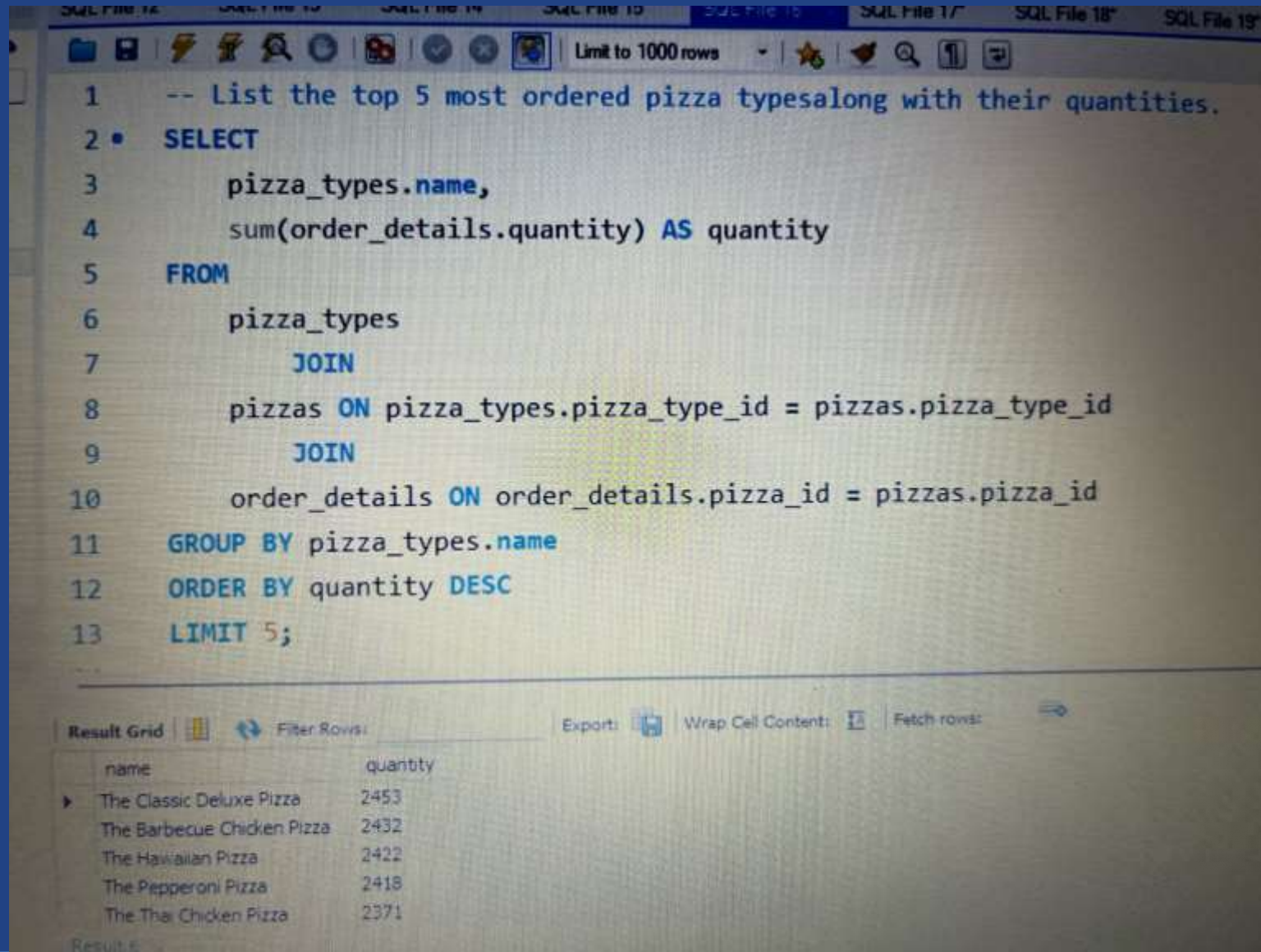
```
1  -- Identify the most common pizza size ordered.--
2  • SELECT
3      COUNT(order_details.quantity) AS quantity_count, pizzas.size
4  FROM
5      order_details
6      JOIN
7      pizzas ON order_details.pizza_id = pizzas.pizza_id
8  GROUP BY pizzas.size
9  ORDER BY quantity_count DESC;
```

Below the query editor, the 'Result Grid' is visible, showing the results of the query. The results are sorted by quantity_count in descending order.

quantity_count	size
18526	L
15385	M
14137	S
544	XL
28	XXL

At the bottom left, it says 'Result 5'.

Q5. List the top 5 most ordered pizza types along with their quantities.



The screenshot shows a SQL IDE interface with a query editor and a result grid. The query is as follows:

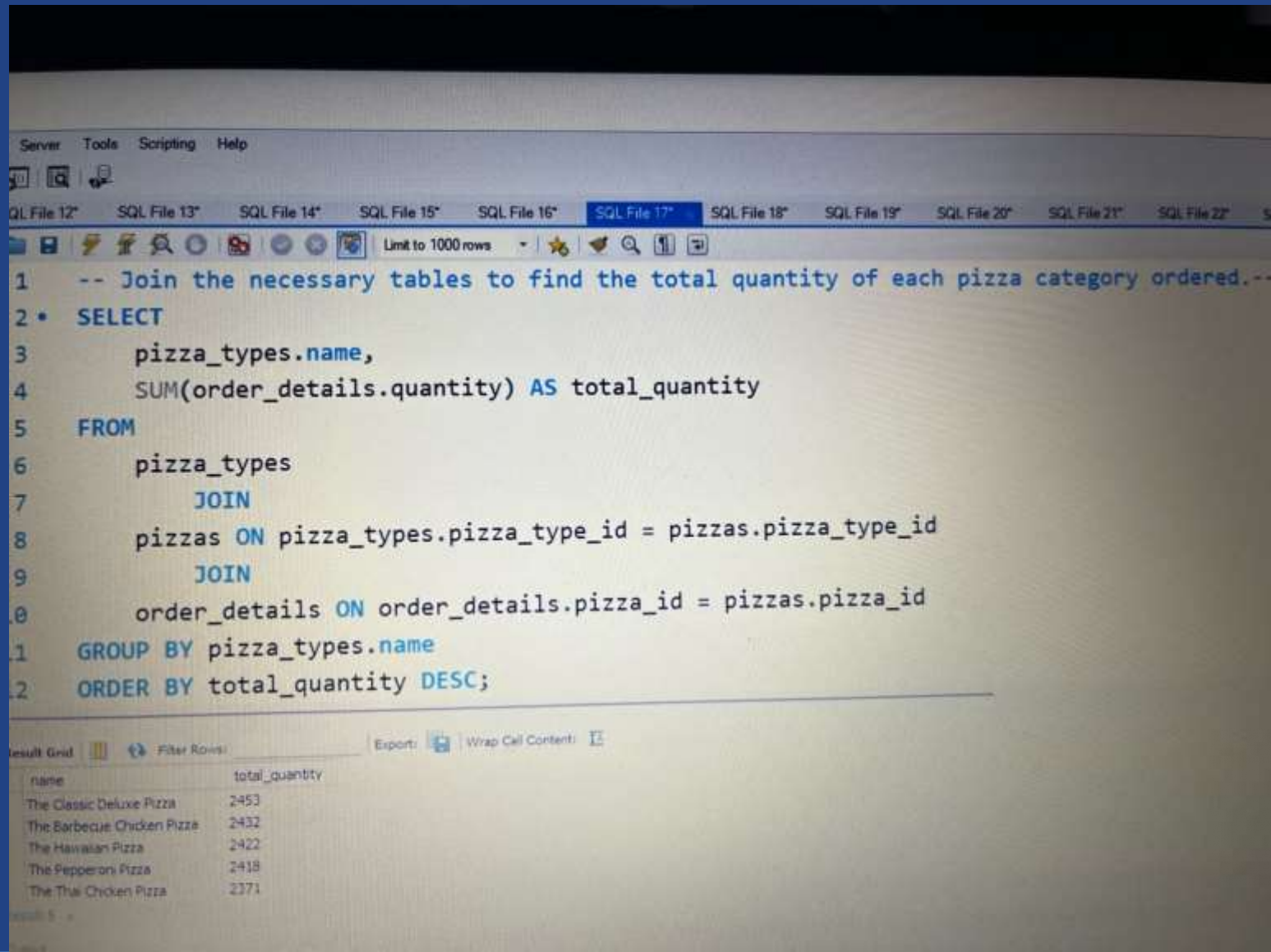
```
1  -- List the top 5 most ordered pizza types along with their quantities.
2  • SELECT
3      pizza_types.name,
4      sum(order_details.quantity) AS quantity
5  FROM
6      pizza_types
7      JOIN
8      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
9      JOIN
10     order_details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

The result grid displays the following data:

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

Result 1

Q6. Total quantity ordered per pizza type.



The screenshot shows a SQL IDE window with a menu bar (Server, Tools, Scripting, Help) and a toolbar. The main editor displays a SQL query. The query is as follows:

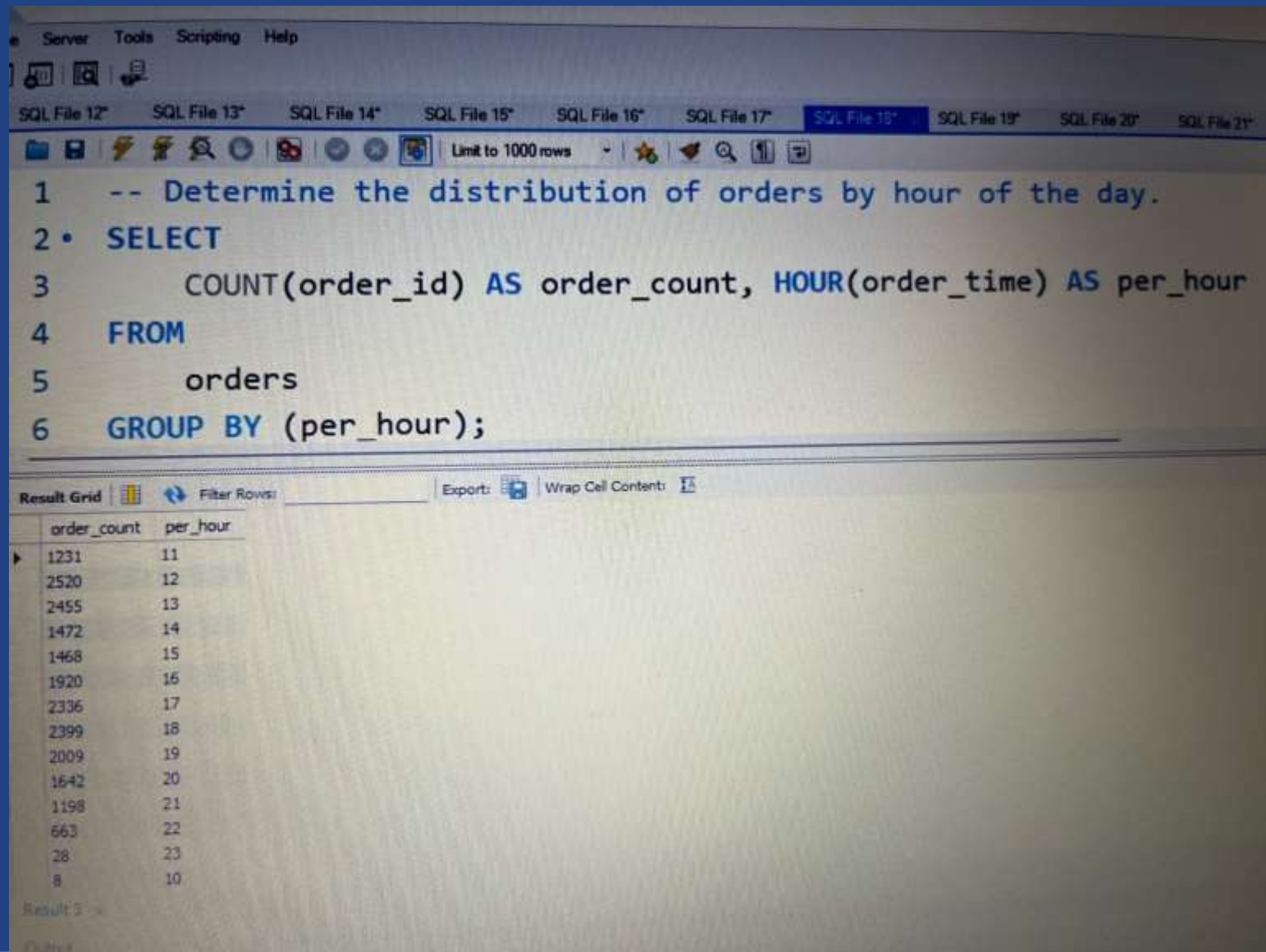
```
-- Join the necessary tables to find the total quantity of each pizza category ordered.--
SELECT
    pizza_types.name,
    SUM(order_details.quantity) AS total_quantity
FROM
    pizza_types
    JOIN
        pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
    JOIN
        order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY total_quantity DESC;
```

Below the query editor, the 'Result Grid' is visible, showing the results of the query. The grid has two columns: 'name' and 'total_quantity'. The results are as follows:

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

In the bottom right corner, there is a logo for 'Domino's' consisting of two overlapping red and blue squares.

Q7. Determine the distribution of orders by hour of the day.



The screenshot shows a SQL IDE interface with a menu bar (Server, Tools, Scripting, Help) and a toolbar. The main window displays a SQL query in a file named 'SQL File 18*'. The query is as follows:

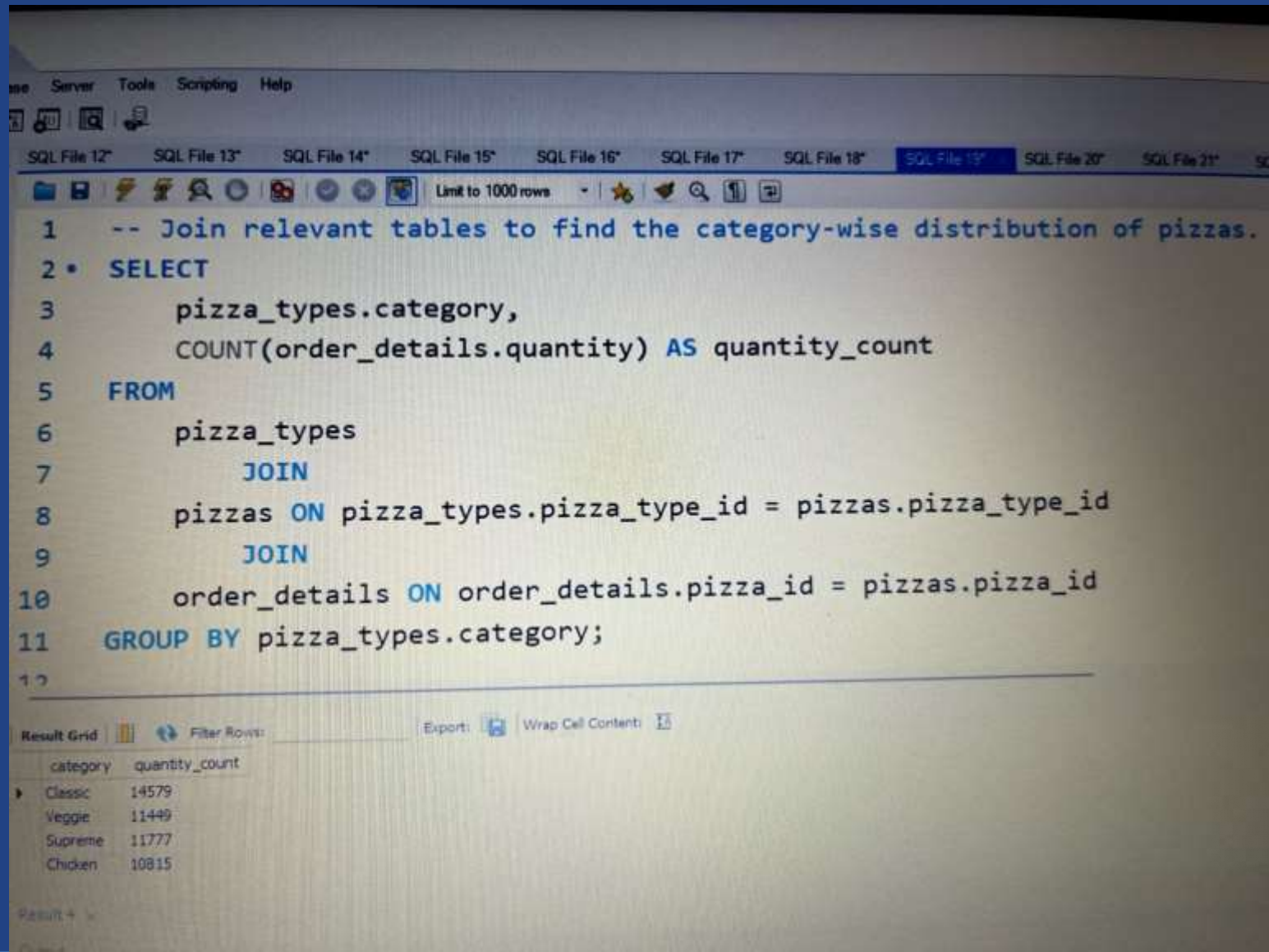
```
1  -- Determine the distribution of orders by hour of the day.
2  • SELECT
3      COUNT(order_id) AS order_count, HOUR(order_time) AS per_hour
4  FROM
5      orders
6  GROUP BY (per_hour);
```

Below the query editor, the 'Result Grid' tab is active, showing the results of the query. The results are displayed in a table with two columns: 'order_count' and 'per_hour'. The data shows the number of orders for each hour of the day, with the highest count of 2520 orders occurring at hour 12.

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

At the bottom left, it says 'Result 3'.

Q8. Category-wise distribution of pizzas.



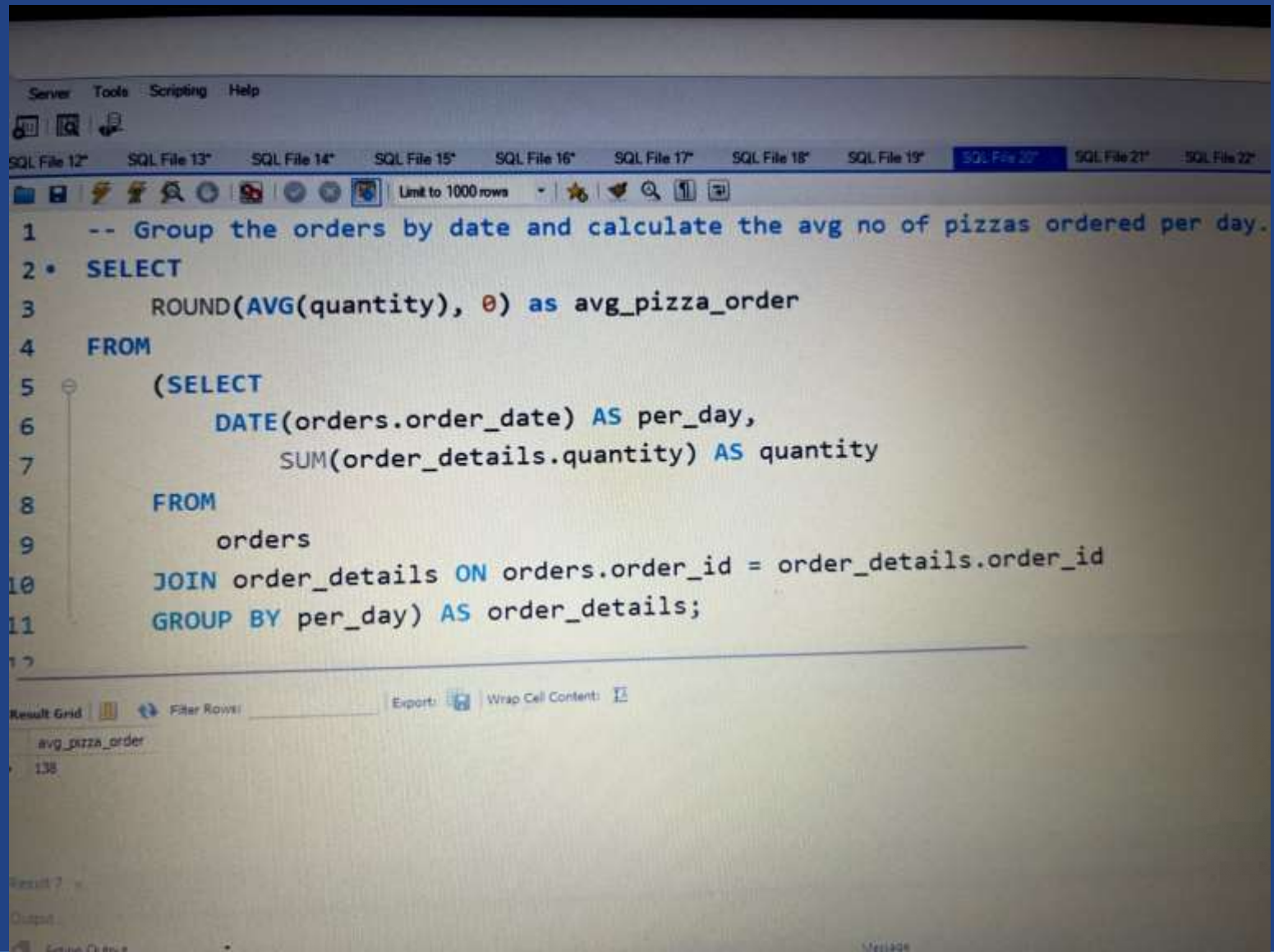
The screenshot shows a SQL IDE interface with a query editor and a result grid. The query is designed to find the category-wise distribution of pizzas by joining the `pizza_types`, `pizzas`, and `order_details` tables. The result grid displays the output of the query, showing the category and the corresponding quantity count.

```
1  -- Join relevant tables to find the category-wise distribution of pizzas.
2  • SELECT
3      pizza_types.category,
4      COUNT(order_details.quantity) AS quantity_count
5  FROM
6      pizza_types
7      JOIN
8      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
9      JOIN
10     order_details ON order_details.pizza_id = pizzas.pizza_id
11 GROUP BY pizza_types.category;
```

Result Grid:

category	quantity_count
Classic	14579
Veggie	11449
Supreme	11777
Chicken	10815

Q9. Average number of pizzas ordered per day.

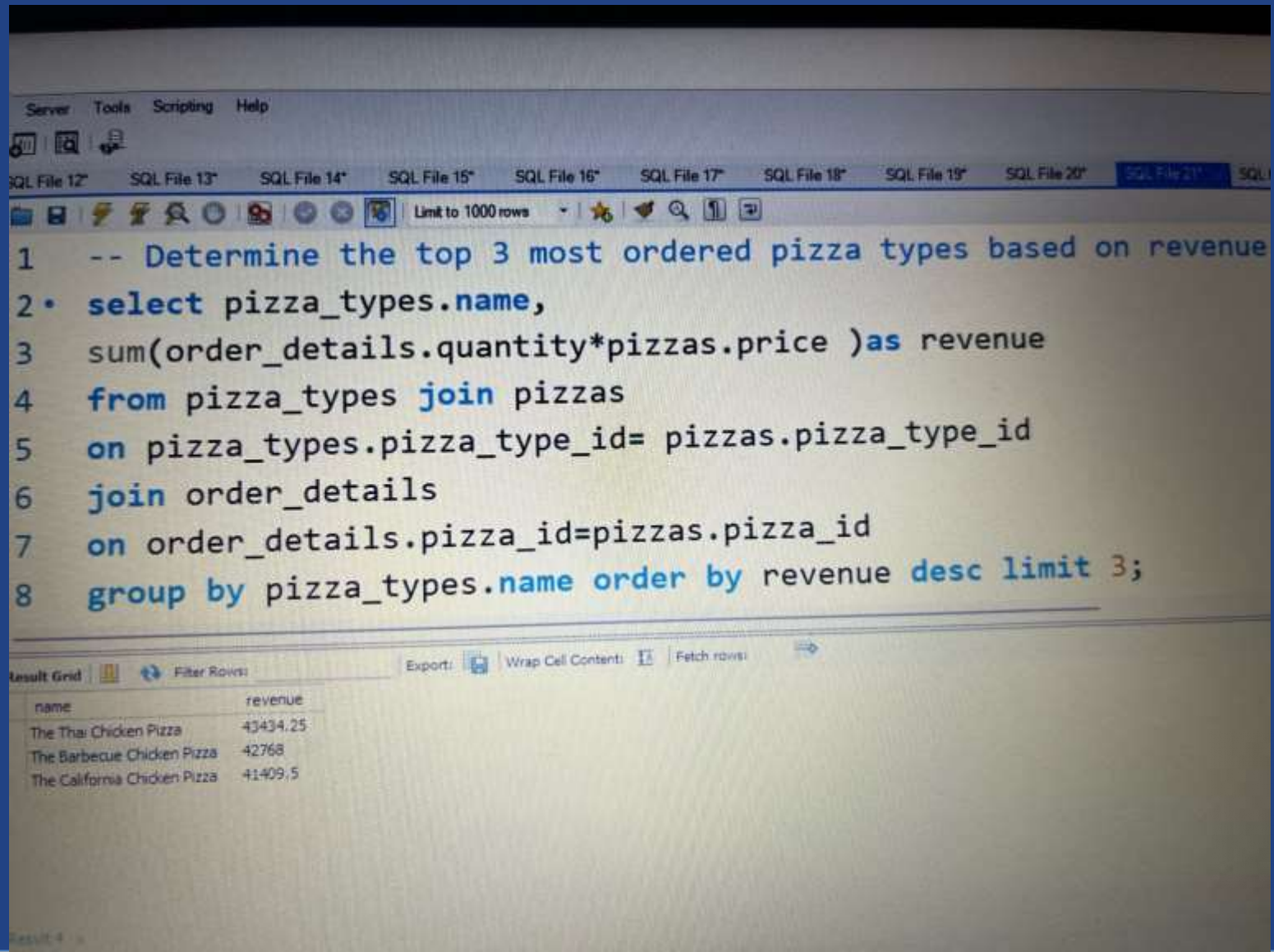


The screenshot shows a SQL IDE interface with a menu bar (Server, Tools, Scripting, Help) and a toolbar. The main window displays a SQL query in a file named 'SQL File 20*'. The query is as follows:

```
1  -- Group the orders by date and calculate the avg no of pizzas ordered per day.
2  • SELECT
3      ROUND(AVG(quantity), 0) as avg_pizza_order
4  FROM
5      (SELECT
6          DATE(orders.order_date) AS per_day,
7          SUM(order_details.quantity) AS quantity
8      FROM
9          orders
10         JOIN order_details ON orders.order_id = order_details.order_id
11        GROUP BY per_day) AS order_details;
```

Below the query editor, there is a 'Result Grid' section. It shows a table with one column, 'avg_pizza_order', and one row with the value '138'. The 'Filter Rows' section is empty. The 'Export' button is visible, and the 'Wrap Cell Content' checkbox is checked.

Q10. Top 3 pizza types by revenue.



The screenshot shows a SQL IDE interface with a query editor and a result grid. The query editor contains the following SQL code:

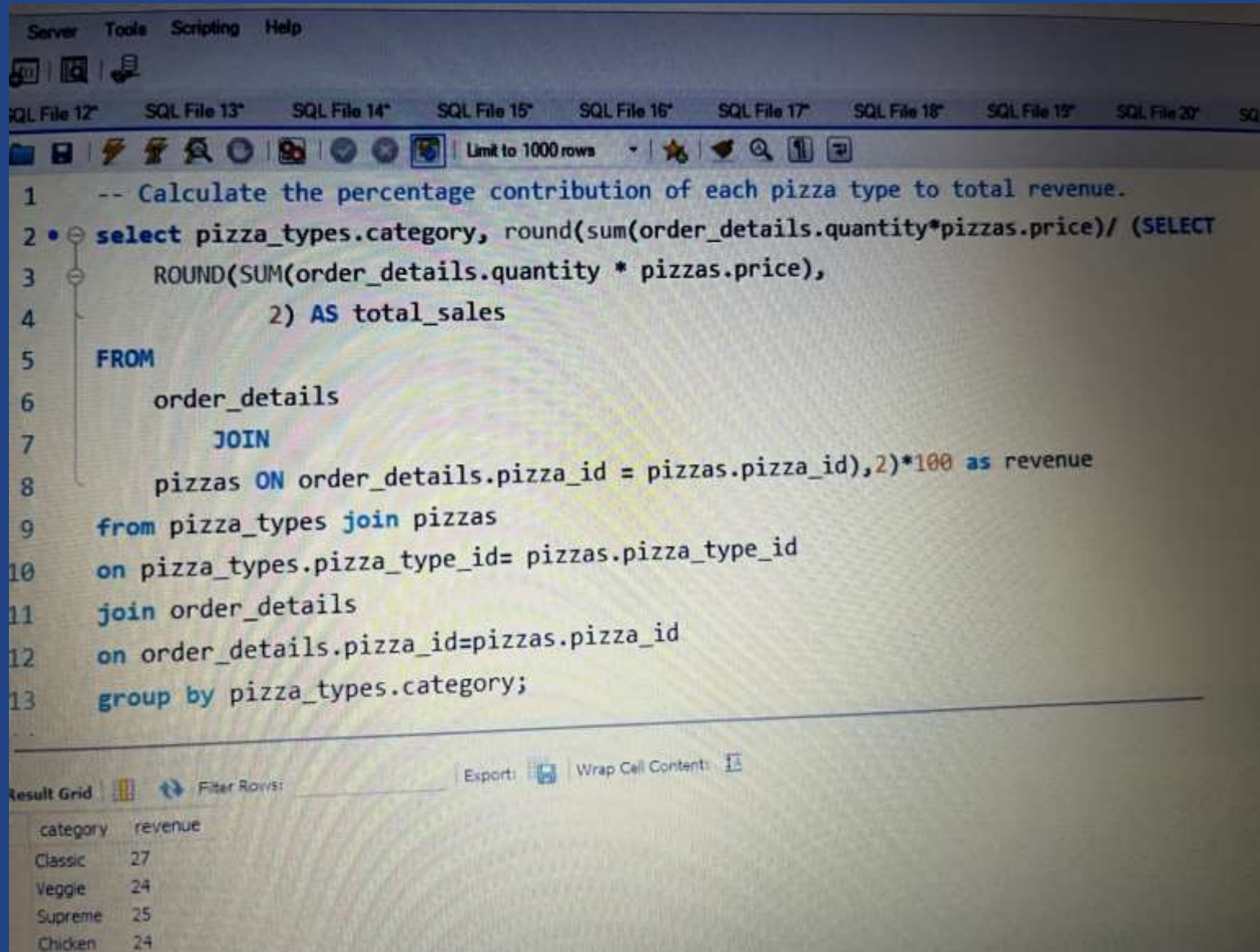
```
1  -- Determine the top 3 most ordered pizza types based on revenue
2 • select pizza_types.name,
3     sum(order_details.quantity*pizzas.price )as revenue
4 from pizza_types join pizzas
5 on pizza_types.pizza_type_id= pizzas.pizza_type_id
6 join order_details
7 on order_details.pizza_id=pizzas.pizza_id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

The result grid displays the following data:

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

Result 4

Q11. Calculate the percentage contribution of each pizza type to total revenue.

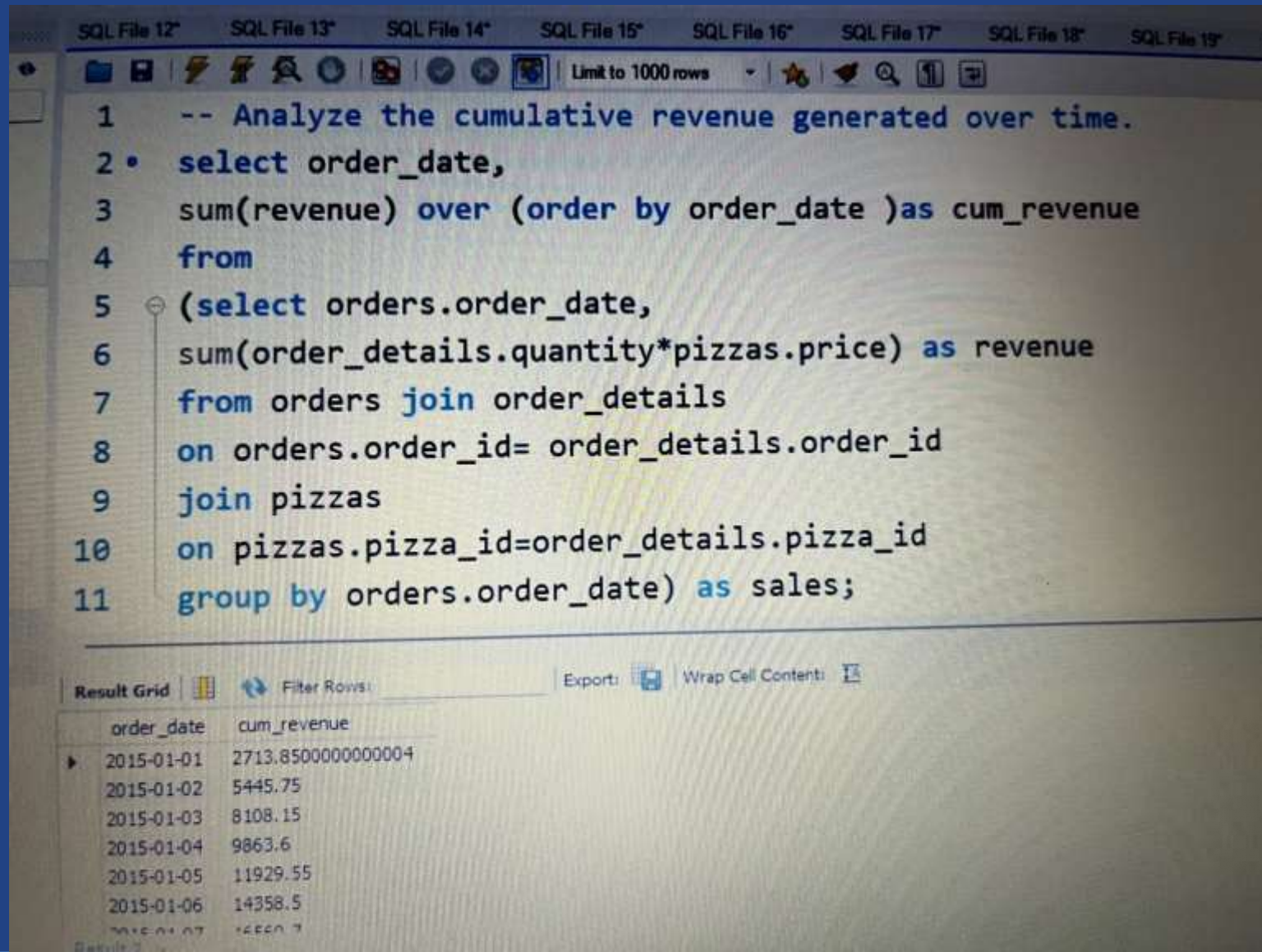


```
1  -- Calculate the percentage contribution of each pizza type to total revenue.
2  • select pizza_types.category, round(sum(order_details.quantity*pizzas.price)/ (SELECT
3     ROUND(SUM(order_details.quantity * pizzas.price),
4         2) AS total_sales
5  FROM
6     order_details
7     JOIN
8 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

Result Grid

category	revenue
Classic	27
Veggie	24
Supreme	25
Chicken	24

Q12. Analyze the cumulative revenue generated over time.



The screenshot shows a SQL IDE with multiple tabs labeled 'SQL File 12*' through 'SQL File 19*'. The active tab displays a SQL query designed to calculate the cumulative revenue generated over time. The query uses a window function, `sum() over (order by order_date)`, to calculate the cumulative revenue. The query is as follows:

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

Below the query editor, the 'Result Grid' tab is active, displaying the results of the query. The results show the order date and the corresponding cumulative revenue.

order_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
2015-01-06	14358.5
2015-01-07	16660.7

The IDE interface includes various toolbars at the top and bottom, such as 'Limit to 1000 rows', 'Export', and 'Wrap Cell Content'.

Q13. Determine the top 3 most ordered pizza types based on revenue for each pizza category.

The screenshot shows a SQL IDE with multiple tabs (SQL File 12 to 19). The active tab displays a SQL query designed to rank pizza types by revenue within each category. The query uses a subquery to calculate revenue (quantity * price) and then applies a window function (rank()) to determine the top 3 in each category. The results are shown in a 'Result Grid' at the bottom, listing the top 3 pizzas for each category along with their revenue and ranking.

```
1  -- Determine the top 3 most ordered pizza types based on revenue for each pizza category.
2  • select name, revenue, ranking from
3  (select category, name, revenue ,
4   rank() over( partition by category order by revenue desc) as ranking
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
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.category, pizza_types.name) as abc) as def
13  where ranking<=3;
```

Result Grid

name	revenue	ranking
The Thai Chicken Pizza	43434.25	1
The Barbecue Chicken Pizza	42768	2
The California Chicken Pizza	41409.5	3
The Classic Deluxe Pizza	38180.5	1
The Hawaiian Pizza	32273.25	2
The Pepperoni Pizza	30161.75	3
The Spicy Italian Pizza	34831.25	1
The Italian Supreme Pizza	33475.75	2

Result 2

Thank You!

This Domino's Pizza SQL Analysis was crafted with data insights.

