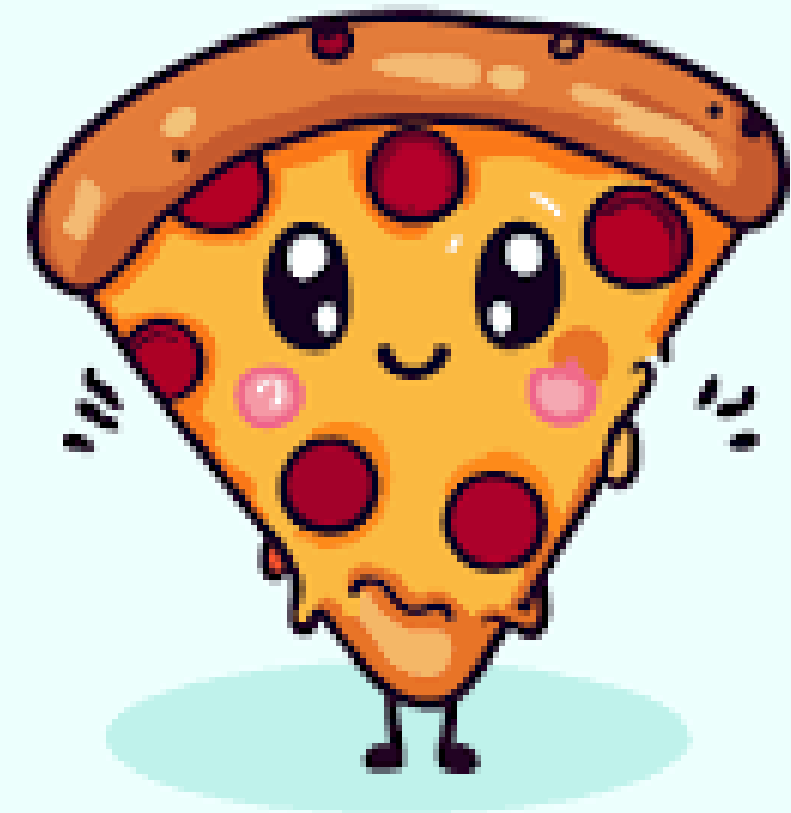


# Pizza Blue

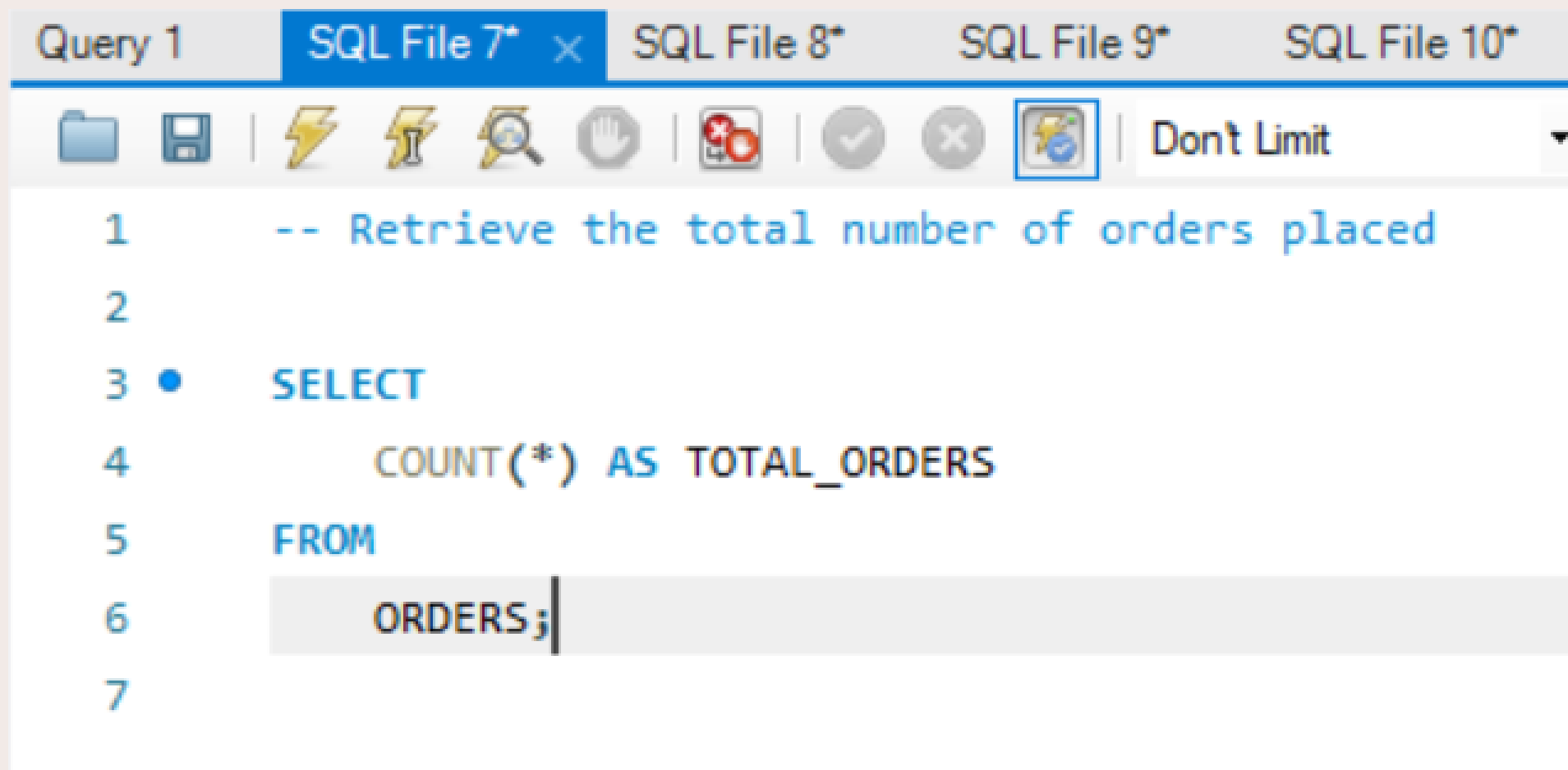
*A sql Project*



# *Pizza Blue Sales Analysis Project*

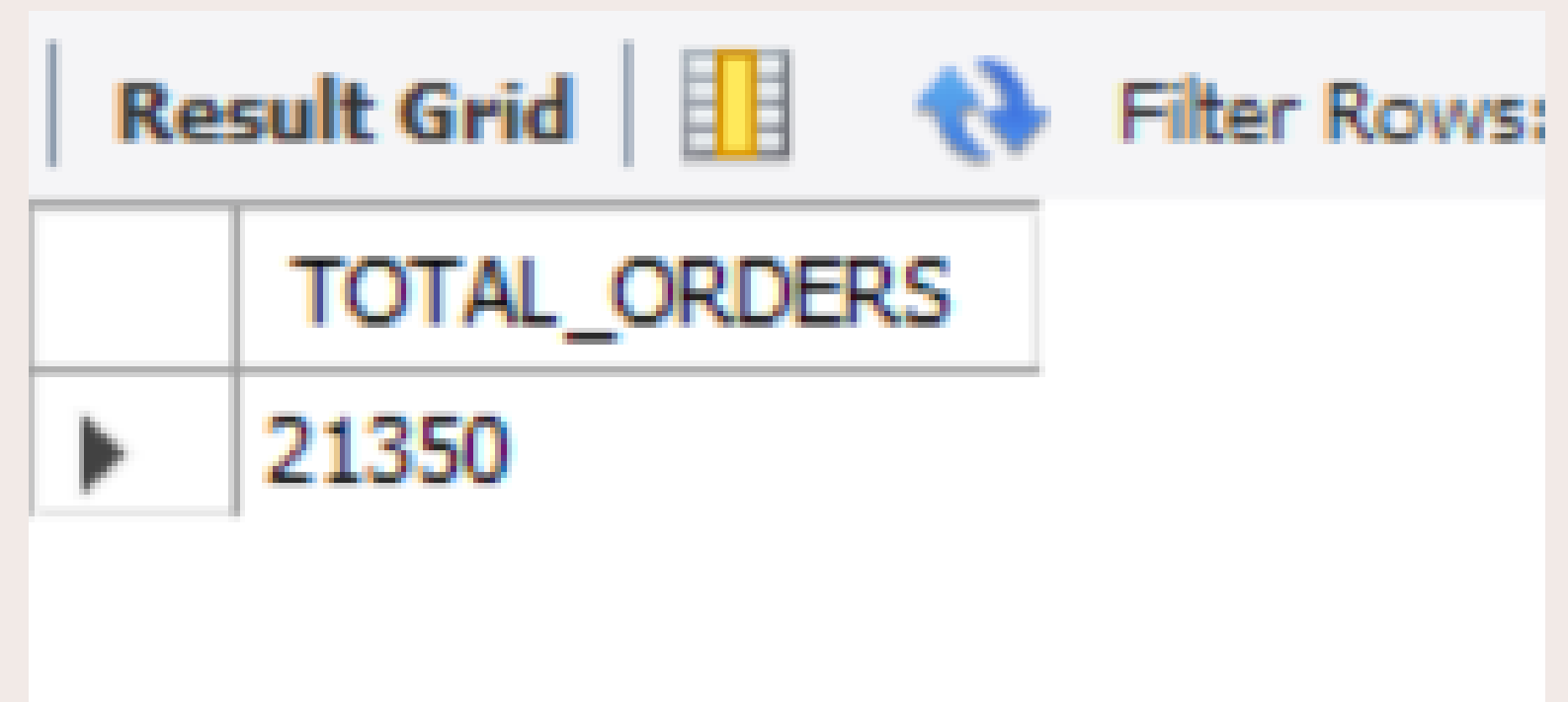
Welcome to the Pizza Blue Sales Analysis Project! Using SQL, we uncover key insights and trends in pizza sales. Our analysis covers sales performance, revenue, and customer preferences, aiding strategic decision-making and optimization. Join us to explore the data and reveal the story behind Pizza Blue's success.

# Retrieve the total number of orders placed



The screenshot shows a SQL IDE with a toolbar at the top containing icons for file operations, execution, and debugging. Below the toolbar, a query editor displays the following SQL code:

```
1  -- Retrieve the total number of orders placed
2
3  •  SELECT
4      COUNT(*) AS TOTAL_ORDERS
5  FROM
6      ORDERS;
7
```



The screenshot shows the Result Grid of the SQL IDE. It displays a single row with the column name **TOTAL\_ORDERS** and the value **21350**. The grid has a header row and a data row.

	TOTAL_ORDERS
▶	21350

# CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES

Query 1   SQL File 7\*   **SQL File 8\*** ×   SQL File 9\*   SQL File 10\*   SQL File 12\*

Ⓛ   Ⓜ   ⚡   ⚡   🔍   🖱   🚫   ⓧ   ⓧ   ⚡   Don't Limit   ⚙   ⭐   🖱   🔄

```
1  -- Calculate the total revenue generated from pizza sales
2
3  •  SELECT
4  Ⓜ    ROUND(SUM(order_details.quantity * pizzas.price),
5         2) AS TOTAL_REVENUE
6  FROM
7      order_details
8      JOIN
9      pizzas USING (pizza_id);
```

Result Grid		Filter Rows:
	TOTAL_REVENUE	
▶	817860.05	

# IDENTIFY THE HIGHEST PRICED PIZZA

```
Query 1  SQL File 7*  SQL File 8*  SQL File 9* x  SQL File
[Icons] Don't Limit
1  -- Identify the highest priced pizza
2
3  •  SELECT
4      pizza_typeS.name, pizzas.price
5  FROM
6      pizza_types
7      JOIN
8      pizzas USING (pizza_type_id)
9  ORDER BY price DESC
10  LIMIT 1
```

Result Grid			Filter Rows:
	name	price	
▶	The Greek Pizza	35.95	

# Identify the most common pizza size orderd



```
Query 1  SQL File 7*  SQL File 8*  SQL File 9*  SQL File 10* x  SQL File 11
[Icons]  Don't Limit
1  -- Identify the most common pizza size orderd
2
3  •  SELECT
4      pizzas.size,
5      COUNT(order_details.order_details_id) AS order_count
6  FROM
7      pizzas
8      JOIN
9      order_details USING (pizza_id)
10  GROUP BY pizzas.size
11  ORDER BY order_count DESC
12  LIMIT 1;
```

Result Grid			Filter Rows:
	size	order_count	
▶	L	18526	

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

```
Query 1  SQL File 7*  SQL File 8*  SQL File 9*  SQL File 10*  SQL File 12* x  SQL File 1
-- List the top 5 most ordered pizza types along with their quantities

1
2
3 • SELECT
4     pizza_types.name,
5     SUM(order_details.quantity) AS pizza_quantity
6 FROM
7     pizza_types
8     JOIN
9     pizzas USING (pizza_type_id)
10    JOIN
11    order_details USING (pizza_id)
12 GROUP BY pizza_types.name
13 ORDER BY pizza_quantity DESC
14 LIMIT 5;
```

Result Grid     Filter Rows: <input type="text"/>		
	name	pizza_quantity
▶	The Classic Deluxe Pizza	2453
	The Barbecue Chicken Pizza	2432
	The Hawaiian Pizza	2422
	The Pepperoni Pizza	2418
	The Thai Chicken Pizza	2371

# JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY

```
Query 1  SQL File 7*  SQL File 8*  SQL File 9*  SQL File 10*  SQL File 12*  SQL File 13* x
[Icons] Don't Limit
1  -- Join the necessary tables to find the total quantity of each pizza category
2
3  •  SELECT
4      pizza_types.category,
5      SUM(order_details.quantity) AS quantity
6  FROM
7      pizza_types
8      JOIN
9      pizzas USING (pizza_type_id)
10     JOIN
11     order_details USING (pizza_id)
12  GROUP BY pizza_types.category
13  ORDER BY quantity DESC
14
```

Result Grid   [Icon] Filter Rows		
	category	quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

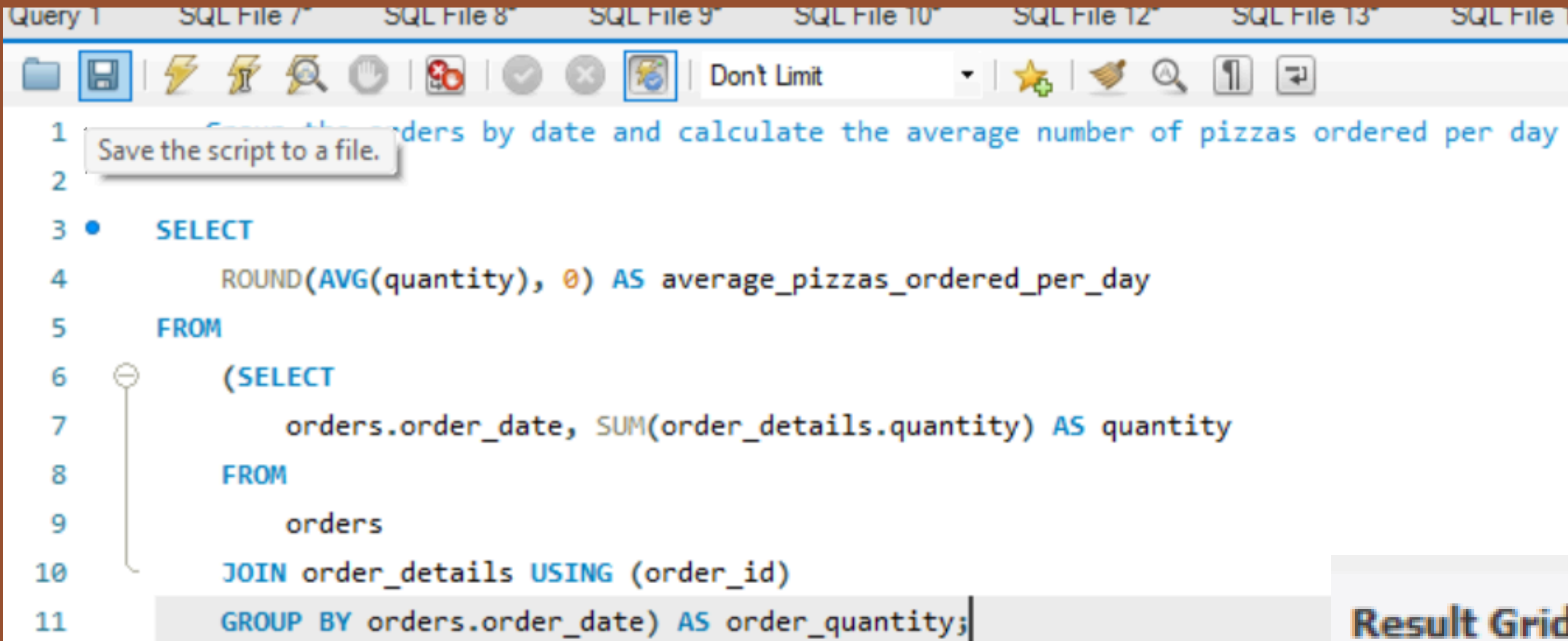


# DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY

```
Query 1    SQL File 7*    SQL File 8*    SQL File 9*    SQL File 10*    SQL File 12*
[Icons] [Don't Limit]
1  -- Determine the distribution of orders by hour of the day
2
3  ●  SELECT
4      HOUR(order_time), COUNT(order_id) AS order_count
5  FROM
6      orders
7  GROUP BY HOUR(order_time)
8  ORDER BY order_count DESC;
```

Result Grid			Filter Rows:
	HOUR(order_time)	order_count	
▶	12	2520	
	13	2455	
	18	2399	
	17	2336	
	19	2009	
	16	1920	1920
	20	1642	
	14	1472	
	15	1468	
	11	1231	
	21	1198	
	22	663	
	23	28	
	10	8	
	9	1	

# GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY



The screenshot shows a SQL IDE interface with multiple tabs at the top: 'Query 1', 'SQL File 7\*', 'SQL File 8\*', 'SQL File 9\*', 'SQL File 10\*', 'SQL File 12\*', 'SQL File 13\*', and 'SQL File 1'. The 'Query 1' tab is active, displaying a SQL script. A tooltip 'Save the script to a file.' is visible over the first line. The script is as follows:

```
1  Group the orders by date and calculate the average number of pizzas ordered per day
2
3  •  SELECT
4      ROUND(AVG(quantity), 0) AS average_pizzas_ordered_per_day
5  FROM
6      (SELECT
7          orders.order_date, SUM(order_details.quantity) AS quantity
8      FROM
9          orders
10     JOIN order_details USING (order_id)
11     GROUP BY orders.order_date) AS order_quantity;
```

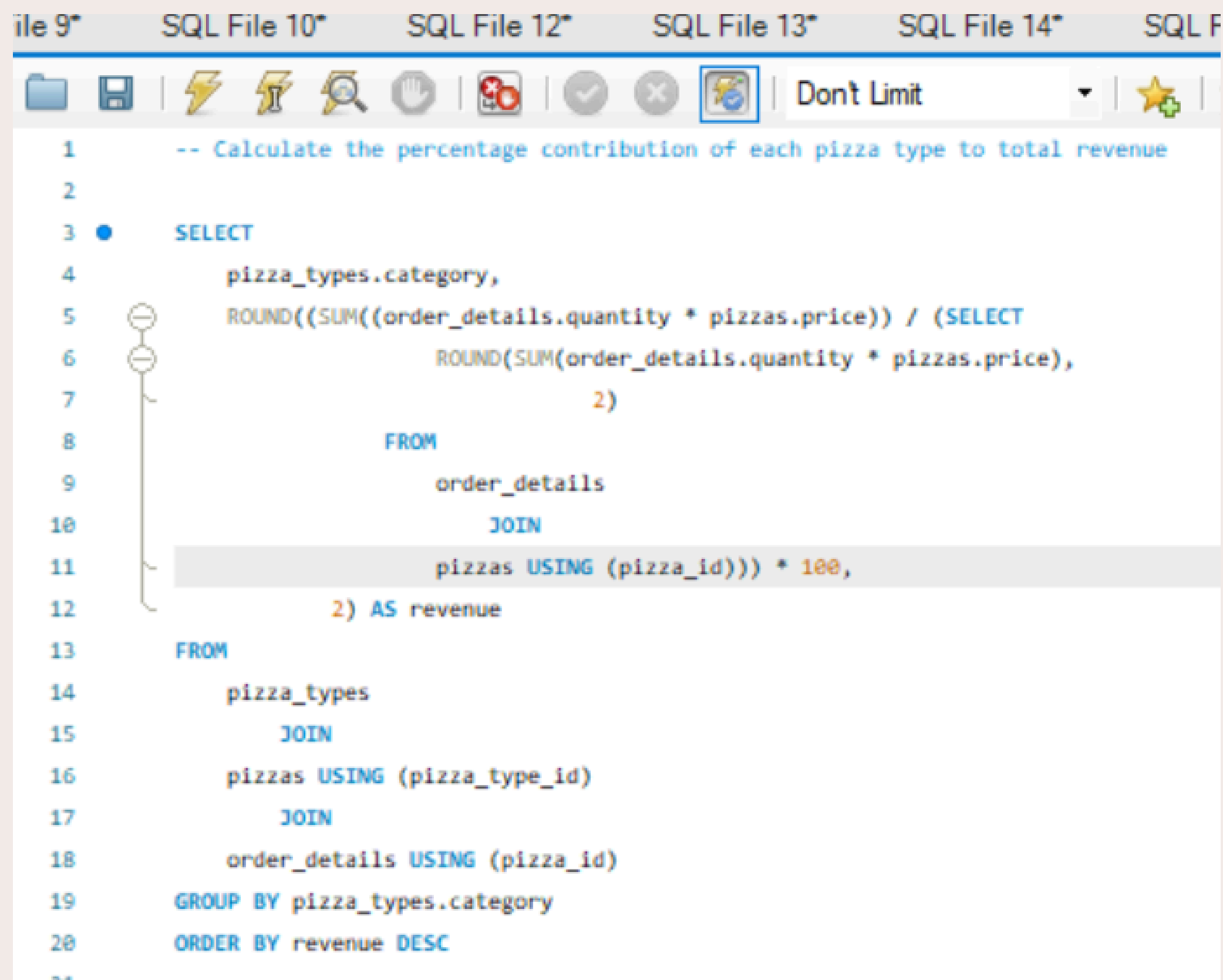
Result Grid		Filter Rows:
	average_pizzas_ordered_per_day	
▶	138	

# DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE

```
1  -- Determine the top 3 most ordered pizza types based on revenue
2
3  • SELECT
4      pizza_types.name,
5      SUM((order_details.quantity * pizzas.price)) AS revenue
6  FROM
7      pizza_types
8      JOIN
9      pizzas USING (pizza_type_id)
10     JOIN
11     order_details USING (pizza_id)
12 GROUP BY pizza_types.name
13 ORDER BY revenue DESC
14 LIMIT 3
```

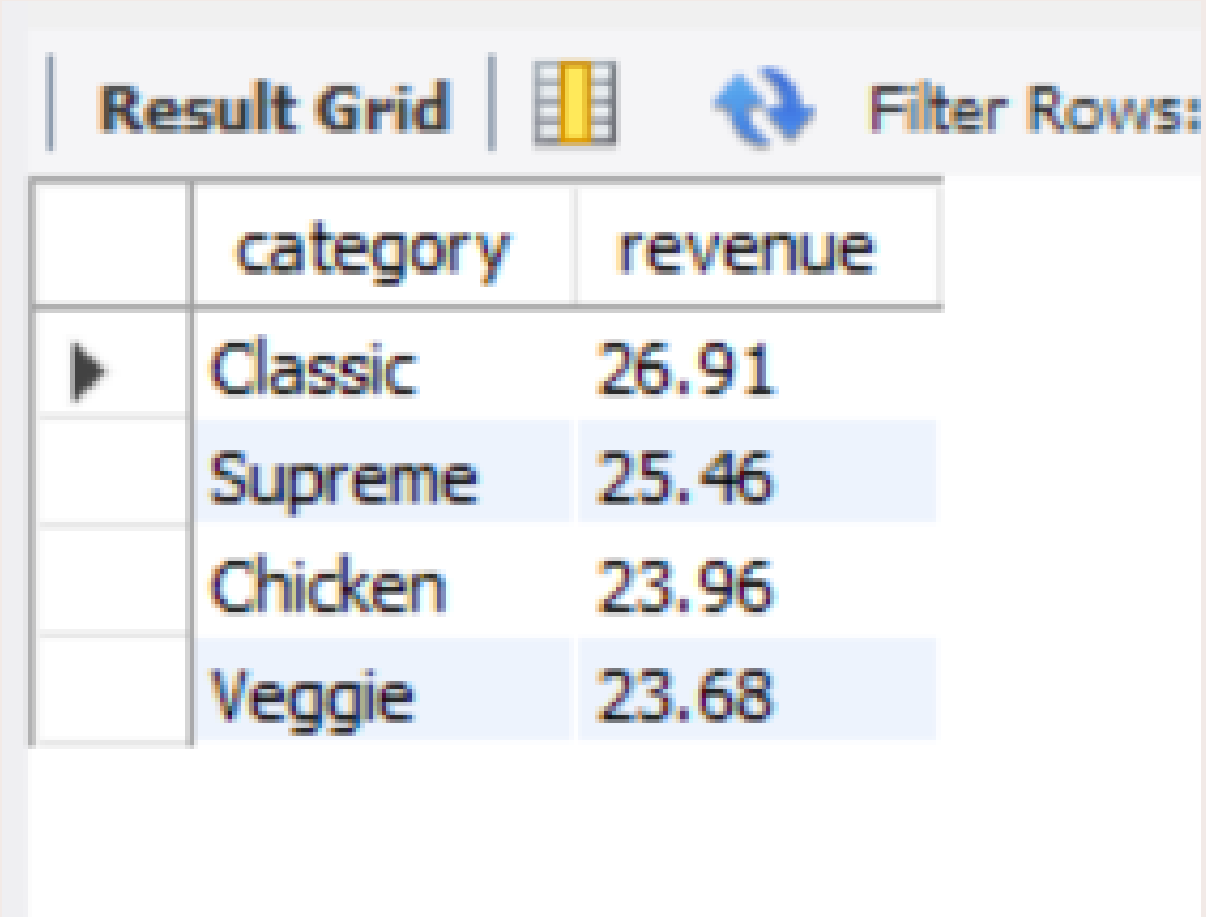
Result Grid			Filter Rows:
	name	revenue	
▶	The Thai Chicken Pizza	43434.25	
	The Barbecue Chicken Pizza	42768	
	The California Chicken Pizza	41409.5	

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



The screenshot shows a SQL IDE with multiple tabs. The active tab is 'SQL File 13'. The query editor contains the following SQL code:

```
1  -- Calculate the percentage contribution of each pizza type to total revenue
2
3  SELECT
4      pizza_types.category,
5      ROUND((SUM((order_details.quantity * pizzas.price)) / (SELECT
6          ROUND(SUM(order_details.quantity * pizzas.price),
7              2)
8      FROM
9          order_details
10         JOIN
11             pizzas USING (pizza_id))) * 100,
12          2) AS revenue
13  FROM
14      pizza_types
15      JOIN
16      pizzas USING (pizza_type_id)
17      JOIN
18      order_details USING (pizza_id)
19  GROUP BY pizza_types.category
20  ORDER BY revenue DESC
```





The screenshot shows the 'Result Grid' of the SQL IDE. It displays the results of the query, which are ordered by revenue in descending order. The table has two columns: 'category' and 'revenue'.

	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

# ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME

```
1  -- Analyze the cumulative revenue generated over time
2
3  •  SELECT order_date, ROUND(SUM(revenue) over(order by order_date), 2) as cumulative_revenue
4     FROM
5     (SELECT orders.order_date, SUM(order_details.quantity*pizzas.price) AS revenue
6      FROM orders
7      JOIN order_details
8      USING(order_id)
9      JOIN pizzas
10     USING(pizza_id)
11     GROUP BY orders.order_date) as sales
```

Result Grid     Filter Rows: <input type="text"/>		
	order_date	cumulative_revenue
	2015-12-15	787777
	2015-12-16	790011.8
	2015-12-17	791892.55
	2015-12-18	794778.85
	2015-12-19	797083.05
	2015-12-20	799187.95
	2015-12-21	801288.65
	2015-12-22	803171.6
	2015-12-23	805415.9
	2015-12-24	807553.75
	2015-12-26	809196.8
	2015-12-27	810615.8
	2015-12-28	812253
	2015-12-29	813606.25
	2015-12-30	814944.05
	2015-12-31	817860.05

# DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY

```
1  -- Determine the top 3 most ordered pizza types based on revenue for each pizza category
2
3  ● SELECT name, revenue, rn
4  FROM
5  (SELECT category, name, revenue, rank() over(PARTITION BY category order by revenue desc) as rn
6  FROM
7  (SELECT pizza_types.category, pizza_types.name, SUM((order_details.quantity * pizzas.price)) as revenue
8  FROM pizza_types
9  JOIN pizzas
10 USING(pizza_type_id)
11 JOIN order_details
12 USING(pizza_id)
13 GROUP BY pizza_types.category, pizza_types.name) as a) as b
14 WHERE rn <= 3;
```

Result Grid				Filter Rows:	Export:	W
	name	revenue	rn			
▶	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	33476.75	2			
	The Sicilian Pizza	30940.5	3			
	The Four Cheese Pizza	32265.700000000065	1			
	The Mexicana Pizza	26780.75	2			
	The Five Cheese Pizza	26066.5	3			

THANK YOU !