



PIZZA SALES





HELLO ALL,

MY NAME IS
RUTUJA NAVGHARE

I have utilized some SQL
queries to solve questions
related to pizza sale





HERE ARE THE QUESTIONS WHICH I SOLVED

1. Retrieve the total number of orders placed.
2. Calculate the total revenue generated from pizza sales.
3. Identify the highest-priced pizza.
4. Identify the most common pizza size ordered.
5. List the 5 most ordered pizza types along with their quantities.
6. Join the necessary tables to find the total quantity of each pizza category ordered.
7. Determine the description of orders by hour of the day.
8. Join the relevant tables to find the category-wise distribution of pizzas.
9. Group the orders by date and calculate the average number of pizzas ordered per date.
10. Determine the top 3 most ordered pizza types based on revenue.
11. Calculate the percentage contribution of each pizza type to total revenue.
12. Analyse the cumulative revenue generated over time.
13. Determine the top 3 most ordered pizza types based on revenue for each pizza category.

1.RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

View Query Database Server Tools Scripting Help

SQL File 3* SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 8* SQL File 9* SQL File 10* SQL File 11* SQL File 12* SQL File 13* SQLAdditions

-- Retrieve the total number of orders placed.

```
1 -- Retrieve the total number of orders placed.
2
3 • SELECT
4     *
5 FROM
6     orders;
7 • SELECT
8     COUNT(order_id) AS total_orders
9 FROM
10    orders;
11
```

Schemas

order_details

details_id	total_orders
	21350

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

Result 2 x Context Help Snippets

Session Output

2. CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SAES.

The screenshot shows the MySQL Workbench interface with a query editor and results grid.

Query Editor:

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

Results Grid:

total_sales
817860.05

Information Panel:

Table: **order_details**

Columns:

order_details_id	int
order_id	int
pizza_id	text
quantity	int

4. IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

Schemas

order_details

order_details_id int PK
order_id int
pizza_id text
quantity int

```
1 -- 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;
```

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

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

Result 7 Result 8 X

Info Session Output

Read Only Context Help Snippets

Type here to search

28°C Haze 4:04 PM 12/6/2024

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or toggle automatic help.

5. LIST THE 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.

local instance MySQL80 ×

View Query Database Server Tools Scripting Help

SQL File 3* SQL File 3* SQL File 4* SQL File 5* SQL File 6* SQL File 7* × SQL File 8* SQL File 9* SQL File 10* SQL File 11* SQL File 12* SQL File 13* SQLAdditions

Projects Page About Tables Views Stored Procedures Functions

```
1 -- List the 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;
```

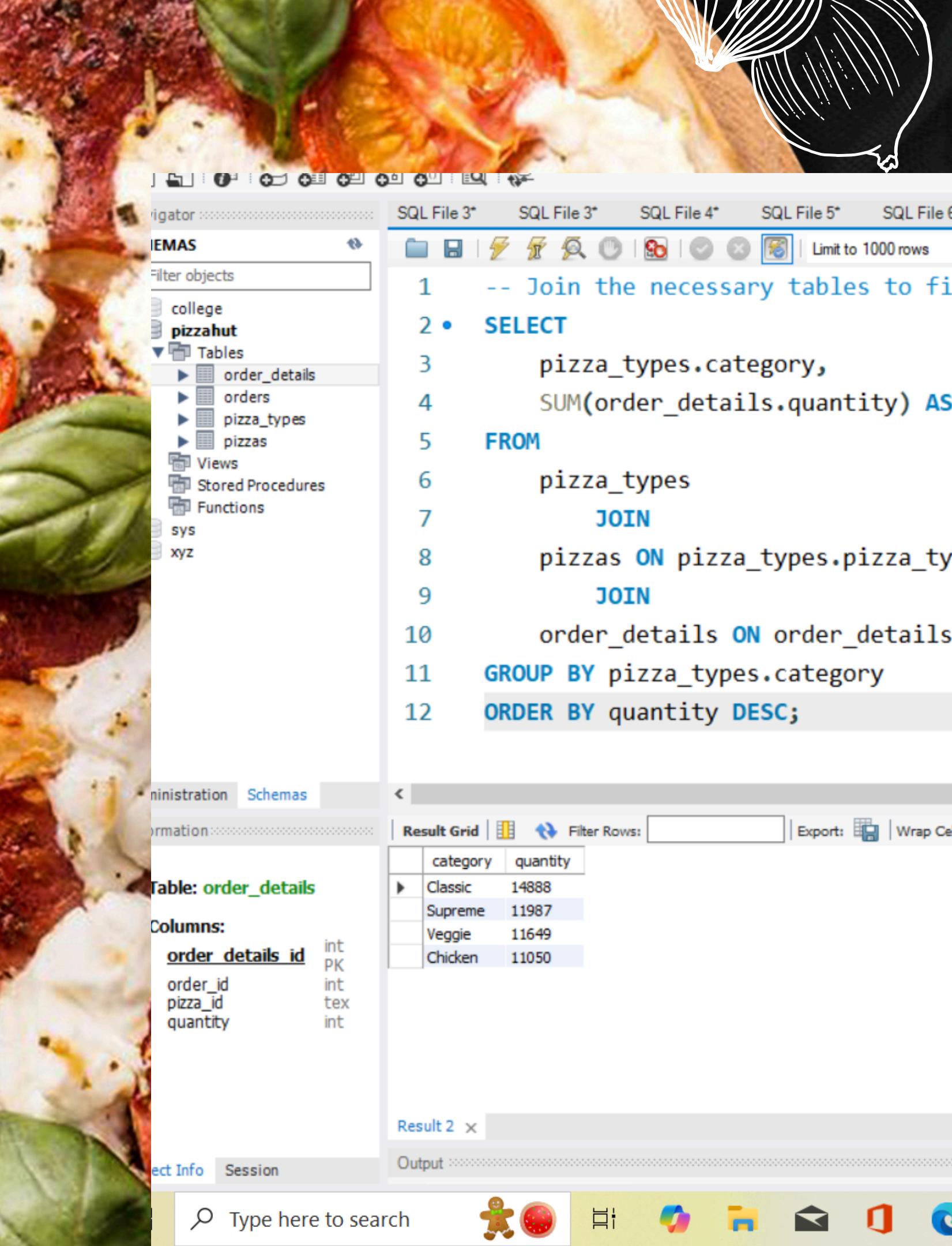
Result Grid | Filter Rows: Export: Wrap Cell Content: Fetch rows:

	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

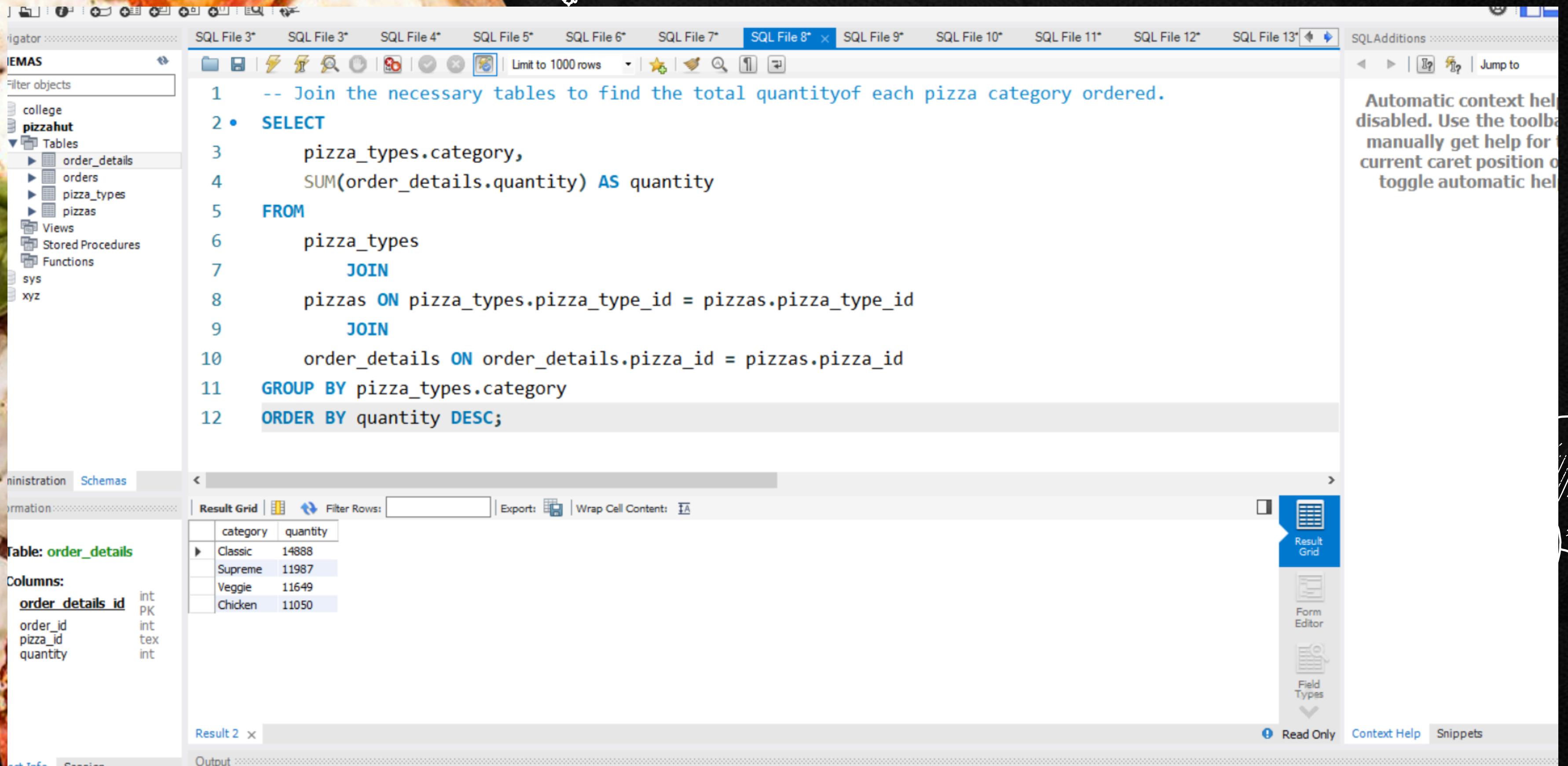
Result 2 × Read Only Context Help Snippets

Session Output

Automatic context help disabled. Use the toolbar manually get help for the current caret position or toggle automatic help.



6.JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.



```
1 -- Join the necessary tables to find the total quantity of each pizza category ordered.
2 • SELECT
3     pizza_types.category,
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.category
12   ORDER BY quantity DESC;
```

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

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Result 2 x Read Only Context Help Snippets

Type here to search

7.DETERMINE THE DESCRIPTION OF ORDERS BY HOUR OF THE DAY.

The screenshot shows the SSMS interface with the following details:

- Toolbar:** Standard SSMS toolbar with various icons for file operations, database management, and help.
- Object Explorer:** Shows the database structure under the 'zzahut' database, including tables like 'order_details', 'orders', 'pizza_types', and 'pizzas'.
- Query Window:** The main window contains a SQL script:

```
1 -- Determine the description of orders by hour of the day.
2 • SELECT
3     HOUR(order_time), COUNT(order_id) AS order_count
4 FROM
5     orders
6 GROUP BY HOUR(order_time);
```
- Results Grid:** A table showing the results of the query:

hour(order_time)	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
- Result Grid Panel:** A sidebar panel titled 'Result Grid' containing icons for 'Form Editor', 'Field Types', 'Query Stats', and 'Execution Plan'.
- Status Bar:** Shows 'Read Only' status, 'Context Help' link, and 'Snippets' link.

8. JOIN THE RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

The screenshot shows a SQL development environment with the following interface elements:

- Toolbar:** Includes icons for file operations, database management, and search.
- Tab Bar:** Shows multiple open files: SQL File 3*, SQL File 3*, SQL File 4*, SQL File 5*, SQL File 6*, SQL File 7*, SQL File 8*, SQL File 9*, SQL File 10* (active), SQL File 11*, SQL File 12*, SQL File 13*, and SQLAdditions.
- Left Sidebar:** Displays the database schema for the 'MAS' database, specifically the 'pizzahut' schema, listing tables like 'order_details', 'orders', 'pizza_types', and 'pizzas'.
- Central Editor Area:** Contains the following SQL code:

```
1 -- Join the relevant tables to find the
2 -- category-wise distribution of pizzas.
3
4 • SELECT
5     category, COUNT(name)
6 FROM
7     pizza_types
8 GROUP BY category
```
- Result Grid:** A table showing the count of pizzas by category. The data is:

category	COUNT(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9
- Status Bar:** Shows 'Read Only', 'Context Help', and 'Snippets' status indicators.
- Bottom Status:** Shows the time '4:14 PM'.

9. GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DATE

The screenshot shows a SQL development environment with the following details:

- Toolbar:** Includes icons for file operations, search, and help.
- File Tab:** Shows multiple open files: SQL File 3*, SQL File 4*, SQL File 5*, SQL File 6*, SQL File 7*, SQL File 8*, SQL File 9*, SQL File 10*, SQL File 11* (highlighted in blue), SQL File 12*, and SQL File 13*.
- Help Panel:** Displays the message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."
- Code Area:** Contains the following SQL query:

```
1 -- Group the orders by date and calculate the average
2 -- number of pizzas ordered per date
3 • SELECT
4     ROUND(AVG(quantity), 0) as avg_pizza_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 ON orders.order_id = order_details.order_id
11    GROUP BY orders.order_date) AS order_quantity;
```
- Result Grid:** Shows the output of the query:

avg_pizza_ordered_per_day
138
- Status Bar:** Shows "Result 3 x" and "Output".
- Bottom Icons:** Includes "Read Only", "Context Help", and "Snippets".

10. DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

The screenshot shows a SQL database interface with the following details:

- Toolbar:** Includes icons for file operations, database management, and search.
- Tab Bar:** Shows multiple open files: SQL File 3*, SQL File 4*, SQL File 5*, SQL File 6*, SQL File 7*, SQL File 8*, SQL File 9*, SQL File 10*, SQL File 11*, SQL File 12*, SQL File 13*, and SQLAdditions.
- Left Sidebar:** Displays a tree view of database objects under the schema "pizza_db".
- Code Area:** Contains the following SQL query:

```
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 ON pizzas.pizza_type_id = pizza_types.pizza_type_id
10     JOIN
11     order_details ON order_details.pizza_id = pizzas.pizza_id
12 GROUP BY pizza_types.name
13 ORDER BY revenue DESC
14 LIMIT 3;
15
```
- Result Grid:** Shows the results of the query in a tabular format.

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
- Status Bar:** Shows "Result 3" and other status indicators like "Read Only", "Context Help", and "Snippets".

11. CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE

The screenshot shows the SQL Server Management Studio interface. The top menu bar includes View, Query, Database, Server, Tools, Scripting, and Help. The toolbar has various icons for file operations, search, and navigation. The title bar shows multiple tabs: File 3*, SQL File 3*, SQL File 4*, SQL File 5*, SQL File 6*, SQL File 7*, SQL File 8*, SQL File 9*, SQL File 10*, SQL File 11*, SQL File 12*, SQL File 13*, and SQLAdditions. A status bar at the bottom indicates 'Result 1' and 'Output'.

The main area contains a query window with 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) AS total_sales
8
9             FROM
10            order_details
11            JOIN
12            pizzas ON pizzas.pizza_id = order_details.pizza_id) * 100,
13        2) AS revenue
14
15     FROM
16     pizza_types
17     JOIN
18     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
19     JOIN
20     order_details ON order_details.pizza_id = pizzas.pizza_id
21     GROUP BY pizza_types.category
22     ORDER BY revenue DESC;
```

The results grid below the code shows the output:

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

A tooltip message on the right side of the interface reads: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

12. ANALYSE THE CUMMULATIVE REVENUE GENERATED OVER TIME.

The screenshot shows a SQL Server Management Studio (SSMS) interface. The top menu bar includes 'Query', 'Database', 'Server', 'Tools', 'Scripting', and 'Help'. The toolbar has various icons for file operations, search, and navigation. The title bar shows multiple tabs: 'File 3*', 'SQL File 4*', 'SQL File 5*', 'SQL File 6*', 'SQL File 7*', 'SQL File 8*', 'SQL File 9*', 'SQL File 10*', 'SQL File 11*', 'SQL File 12*', 'SQL File 13*', and 'SQL File 14*' (the active tab). A status bar at the bottom right indicates 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

The main code editor window contains the following SQL script:

```
1 -- Analyse the cummulative 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 order_details join pizzas  
8       on order_details.pizza_id = pizzas.pizza_id  
9     join orders  
10    on orders.order_id = order_details.order_id  
11   group by orders.order_date) as sales;  
12  
13
```

The results pane below shows a 'Result Grid' with the following data:

	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

At the bottom of the results pane, there are buttons for 'Result Grid' (selected), 'Read Only', 'Context Help', and 'Snippets'.

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

The screenshot shows a SQL development environment with the following interface elements:

- Toolbar:** Includes icons for file operations, search, and help.
- File Menu:** Labeled "SQL File 4*" through "SQL File 15*".
- Help Menu:** Labeled "SQLAdditions".
- Result Grid:** A table showing the results of the query execution.
- Code Editor:** Displays the SQL query for determining the top 3 most ordered pizza types based on revenue for each pizza category.
- Output Window:** Labeled "Output" at the bottom left.
- Status Bar:** Shows "Result 4" and other status indicators.

Code Editor Content:

```
1 -- Determine the top 3 most ordered pizza types based on revenue for each pizza category
2 • select name,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
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 a) as b
13 where rn <=3;
```

Result Grid Data:

	name	revenue
▶	The Thai Chicken Pizza	43434.25
	The Barbecue Chicken Pizza	42768
	The California Chicken Pizza	41409.5
	The Classic Deluxe Pizza	38180.5
	The Hawaiian Pizza	32273.25
	The Pepperoni Pizza	30161.75
	The Spicy Italian Pizza	34831.25
	The Italian Supreme Pizza	33476.75



THANK YOU!

