

# Pizza Sales Analysis using SQL

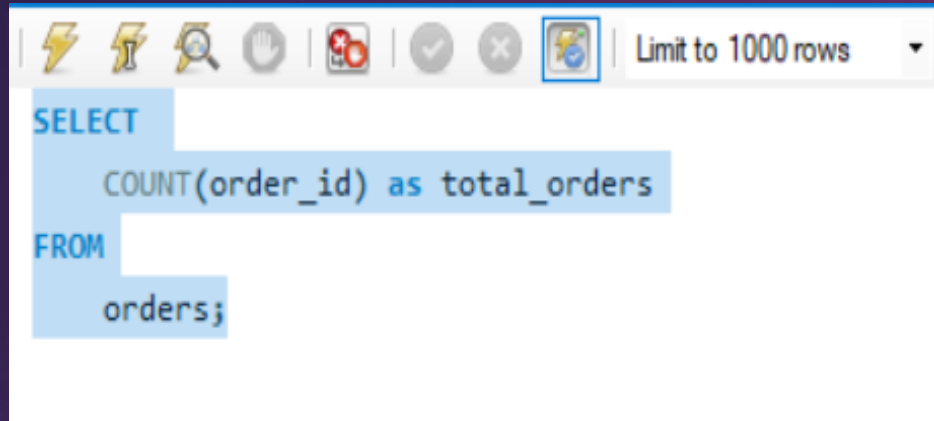
- **Project Overview:** Analyzed operational and sales data of PizzaHub using SQL to derive actionable business insights.
- **Data Integration:** Combined data from various sources to create a comprehensive dataset.
- **SQL Queries:** Developed and executed complex SQL queries to extract and manipulate data, including joins, aggregations, and subqueries.
- **Reporting:** Created report as pdf file to visualize findings, using SQL to generate summary statistics and performance metrics by placing snapshots of each query operation.

## Tables –

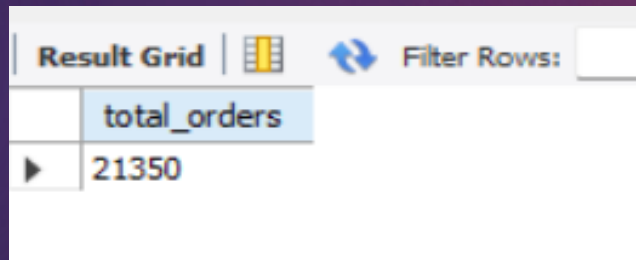
The database consist of following four tables –

1. Orders
2. Order\_details
3. Pizza\_types
4. Pizzas

Question1 - Retrieve the total number of orders placed.



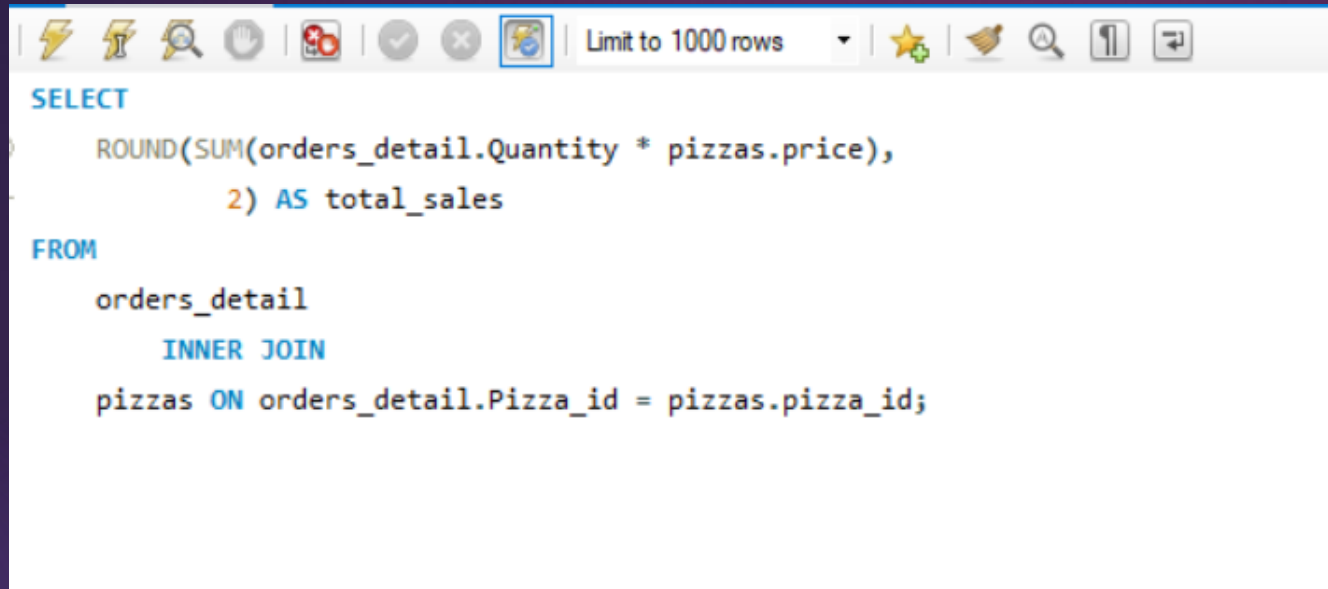
```
SELECT
    COUNT(order_id) as total_orders
FROM
    orders;
```



Result Grid		Filter Rows:
	total_orders	
▶	21350	

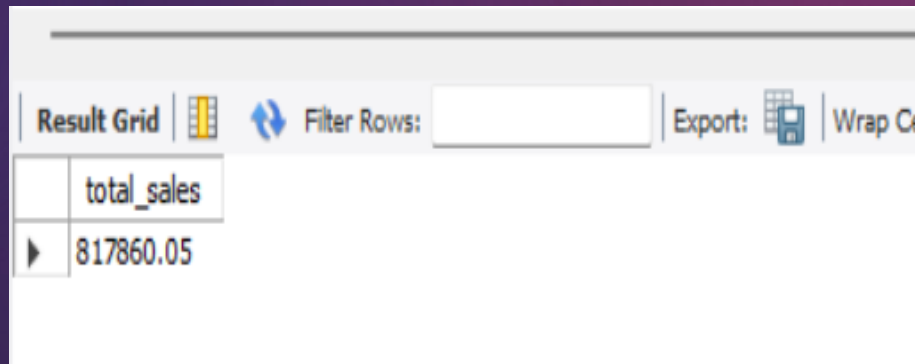


Question2 - Calculate the total revenue generated from pizza sales.



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

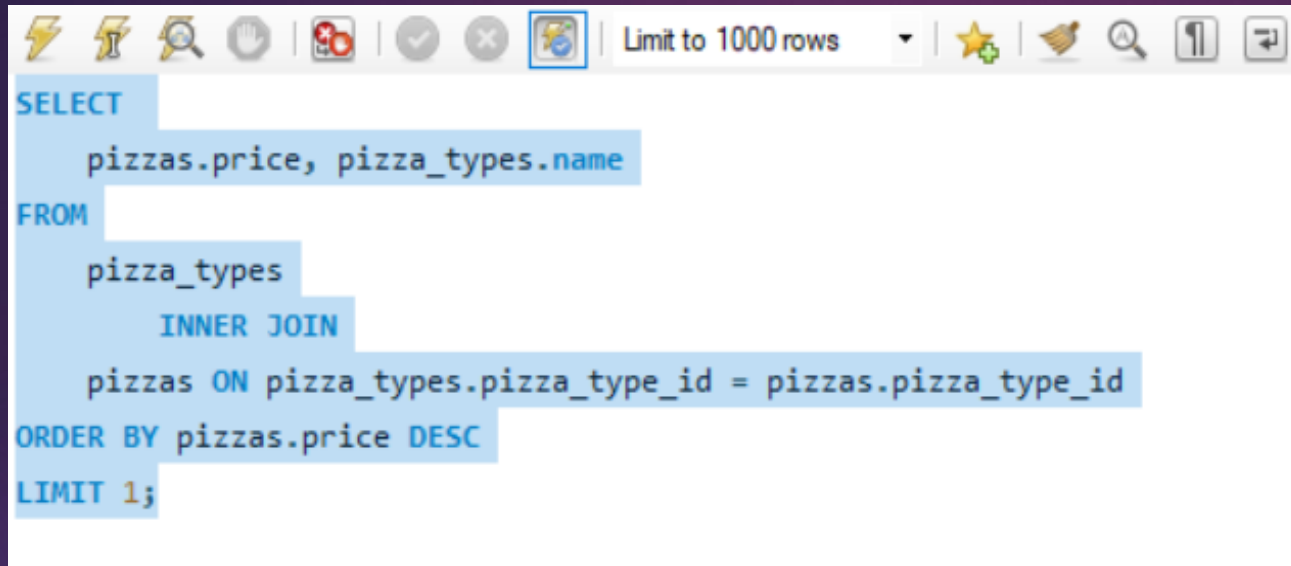
```
SELECT  
    ROUND(SUM(orders_detail.Quantity * pizzas.price),  
          2) AS total_sales  
FROM  
    orders_detail  
    INNER JOIN  
    pizzas ON orders_detail.Pizza_id = pizzas.pizza_id;
```



The screenshot shows a database result grid with the following data:

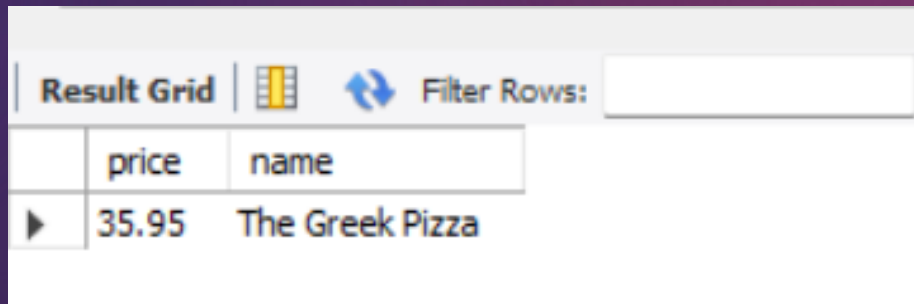
	total_sales
▶	817860.05

Question3- Identify the highest-priced pizza.



The screenshot shows a SQL query editor with a toolbar at the top containing icons for undo, redo, search, and other functions. The query text is as follows:

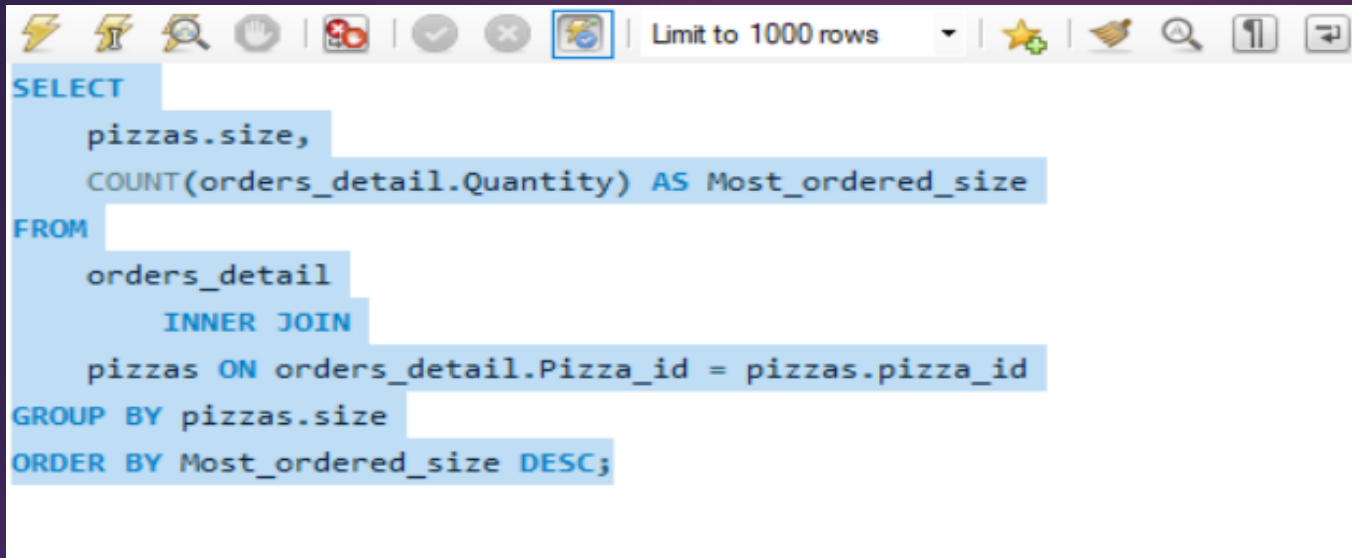
```
SELECT
    pizzas.price, pizza_types.name
FROM
    pizza_types
    INNER JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
```



The screenshot shows a 'Result Grid' window with a 'Filter Rows' input field. The results are displayed in a table with two columns: 'price' and 'name'.

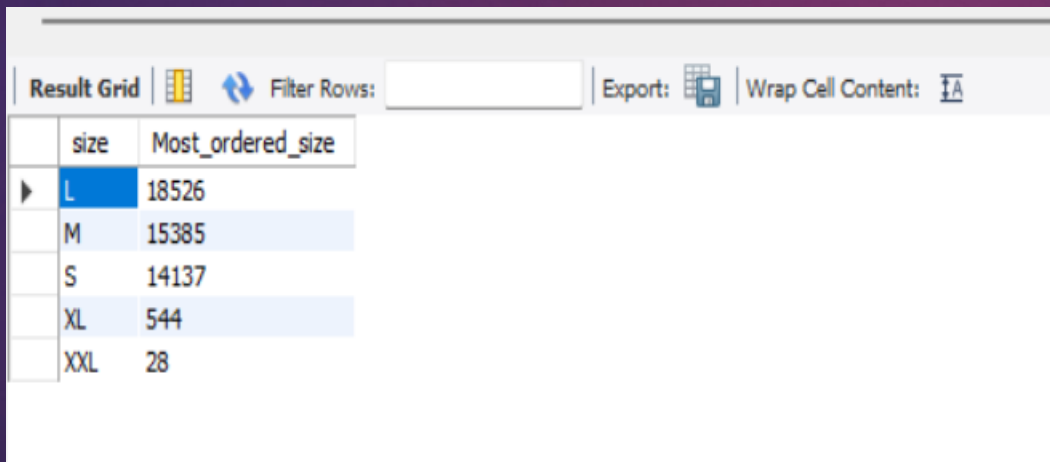
	price	name
▶	35.95	The Greek Pizza

Question4 - Identify the most common pizza size ordered.



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:


```
SELECT
    pizzas.size,
    COUNT(orders_detail.Quantity) AS Most_ordered_size
FROM
    orders_detail
    INNER JOIN
    pizzas ON orders_detail.Pizza_id = pizzas.pizza_id
GROUP BY pizzas.size
ORDER BY Most_ordered_size DESC;
```



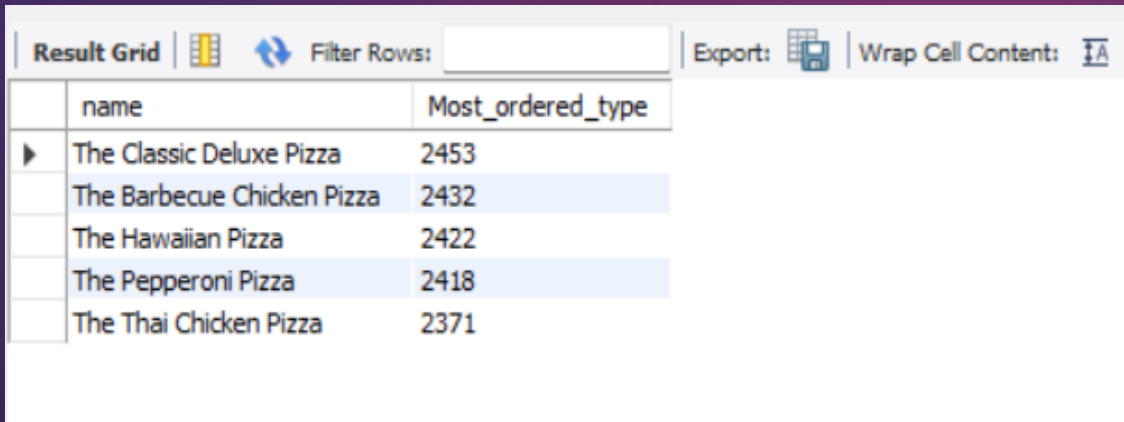
The screenshot shows a database result grid with the following data:

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

Question5 - List the top 5 most ordered pizza types along with their quantities.



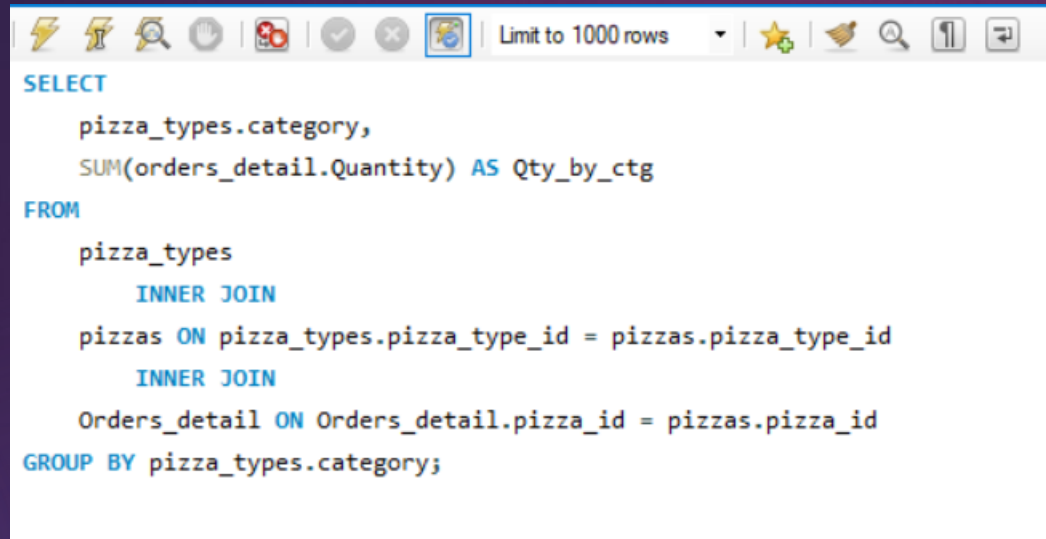
```
SELECT
    pizza_types.name,
    SUM(orders_detail.Quantity) AS Most_ordered_type
FROM
    pizza_types
    INNER JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
    INNER JOIN
    orders_detail ON orders_detail.Pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY Most_ordered_type DESC
LIMIT 5;
```



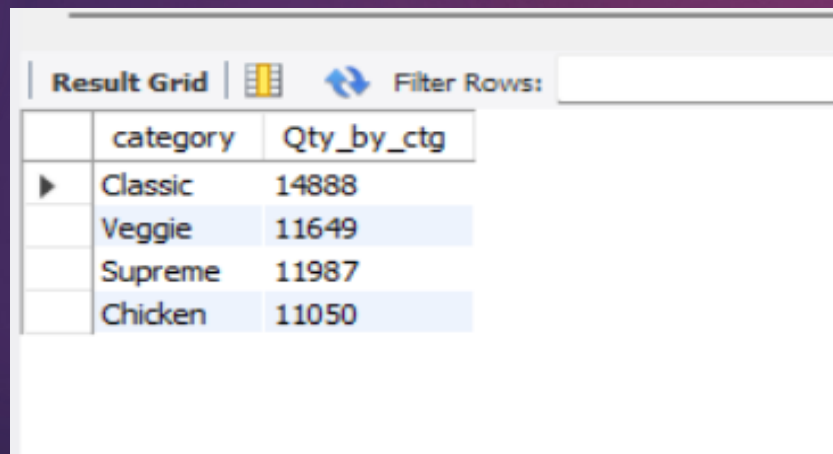
	name	Most_ordered_type
▶	The Classic Deluxe Pizza	2453
	The Barbecue Chicken Pizza	2432
	The Hawaiian Pizza	2422
	The Pepperoni Pizza	2418
	The Thai Chicken Pizza	2371



Question6 - Join the necessary tables to find the total quantity of each pizza category ordered.



```
SELECT
    pizza_types.category,
    SUM(orders_detail.Quantity) AS Qty_by_ctg
FROM
    pizza_types
    INNER JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
    INNER JOIN
    Orders_detail ON Orders_detail.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category;
```



	category	Qty_by_ctg
▶	Classic	14888
	Veggie	11649
	Supreme	11987
	Chicken	11050


Question7 -Determine the distribution of orders by hour of the day.

```
SELECT
    HOUR(Oder_time) AS Hours, COUNT(Order_id) AS No_of_Orders
FROM
    orders
GROUP BY HOUR(Oder_time);
```



	Hours	No_of_Orders
▶	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



Question8 - Join relevant tables to find the category-wise distribution of pizzas.



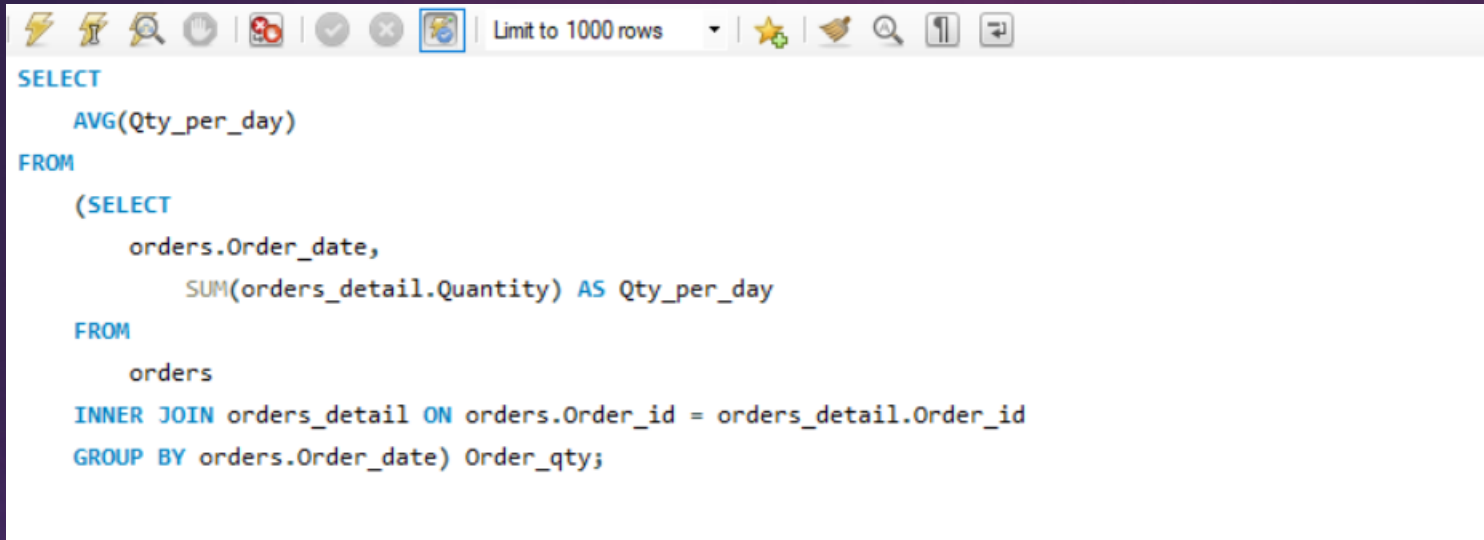
```
SELECT
    category, COUNT(name)
FROM
    pizza_types
GROUP BY category;
```

Result Grid |   Filter Rows:

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

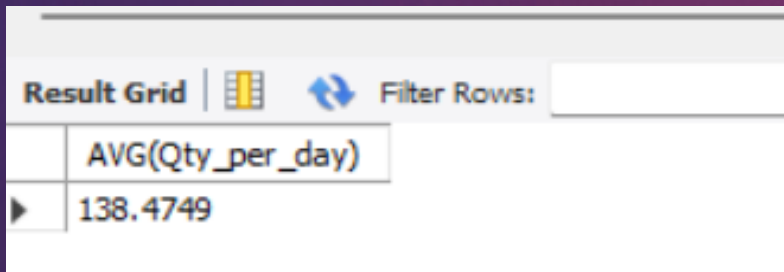


Question9 -Group the orders by date and calculate the average number of pizzas ordered per day.



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
SELECT
  AVG(Qty_per_day)
FROM
  (SELECT
    orders.Order_date,
    SUM(orders_detail.Quantity) AS Qty_per_day
  FROM
    orders
  INNER JOIN orders_detail ON orders.Order_id = orders_detail.Order_id
  GROUP BY orders.Order_date) Order_qty;
```



The screenshot shows a 'Result Grid' with a 'Filter Rows' input field. The grid contains one row with the following data:

	AVG(Qty_per_day)
▶	138.4749

## Question10 -Determine the top 3 most ordered pizza types based on revenue

```
SELECT
    pizza_types.name,
    ROUND(SUM(orders_detail.Quantity * pizzas.price),
          0) AS Revenue
FROM
    pizza_types
    INNER JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
    INNER JOIN
    orders_detail ON orders_detail.Pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```

	name	Revenue
▶	The Thai Chicken Pizza	43434
	The Barbecue Chicken Pizza	42768
	The California Chicken Pizza	41410



Question11 - Calculate the percentage contribution of each pizza type to total revenue.

```
Limit to 1000 rows
Select pizza_types.category,
round(sum(orders_detail.quantity*pizzas.price) / (Select round(sum(orders_detail.quantity*pizzas.price),2) As Total_Sales
From orders_detail
Join
  pizzas on pizzas.pizza_id = orders_detail.Pizza_id)*100,2) As revenue
from pizza_types join Pizzas
On pizza_types.pizza_type_id = pizzas.pizza_type_id
Join orders_detail
On orders_detail.Pizza_id = pizzas.pizza_id
group by pizza_types.category order by revenue desc;
```

Result Grid

Filter Rows:

	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68



Question 12 - Analyze the cumulative revenue generated over time.

```
Select order_date ,  
       sum(revenue) over (order by order_date ) as cum_revenue  
from  
(select orders.order_date, sum(orders_detail.quantity*pizzas.price) as revenue  
 from orders_detail join pizzas  
 on orders_detail.pizza_id = pizzas.pizza_id  
 Join orders on orders.order_id = orders_detail.order_id  
 group by orders.order_date ) as sales;
```

Result Grid			Filter Rows:	Export:	Wrap
	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	16560.7			



**THANKYOU**

