




Pizza Sales

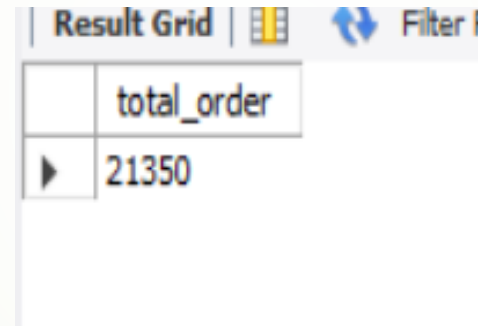
Problem queries

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 top 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 distribution of orders by hour of the day.

- 
8. Join 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 day.
 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. Analyze the cumulative revenue generated over time.
 13. Determine the top 3 most ordered pizza types based on revenue for each pizza category.

Retrieve the total number of orders placed

```
select count(order_id) as total_order from orders;
```

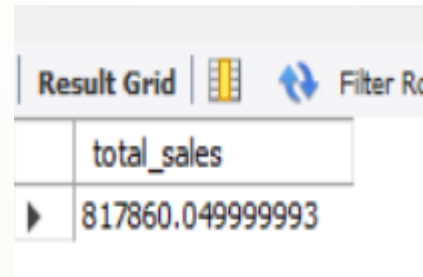


A screenshot of a database query result grid. The grid has a header row with the column name 'total_order' and a data row with the value '21350'. The grid is titled 'Result Grid' and has a 'Filter' button.

	total_order
▶	21350

Calculate the total revenue generated from pizza sales.

```
SELECT SUM(order_details.quantity * pizzas.price) AS  
total_sales FROM order_details JOIN pizzas ON  
pizzas.pizza_id = order_details.pizza_id;
```

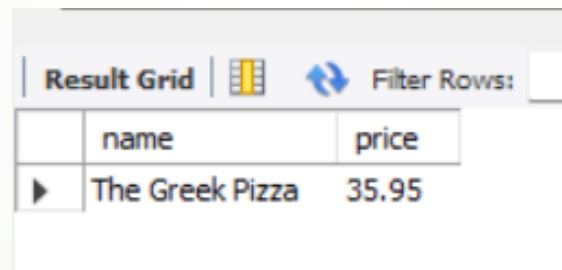


The screenshot shows a database query result grid. At the top, there is a header bar with the text "Result Grid" and a "Filter Rows" button. Below the header, the grid has two columns: "total_sales" and a value column. The value column contains the number "817860.0499999993".

	total_sales
▶	817860.0499999993

Identify the highest-priced pizza.

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

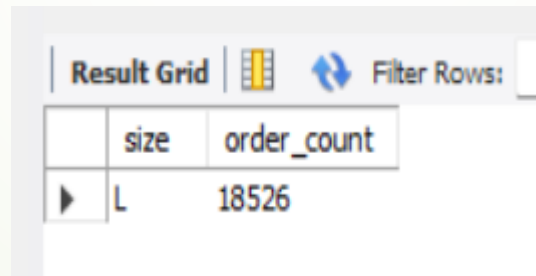


A screenshot of a database query result grid. The grid has two columns: 'name' and 'price'. The first row shows 'The Greek Pizza' with a price of 35.95. Above the grid, there are icons for 'Result Grid', a grid icon, a refresh icon, and a 'Filter Rows:' input field.

	name	price
▶	The Greek Pizza	35.95

Identify the most common pizza size ordered

```
SELECT pizzas.size, COUNT(order_details.order_details_id)  
AS order_count FROM pizzas JOIN order_details ON  
pizzas.pizza_id = order_details.pizza_id GROUP BY pizzas.size  
ORDER BY order_count DESC LIMIT 1;
```



The screenshot shows a database query result grid. At the top, there is a tab labeled 'Result Grid' and a 'Filter Rows:' section. Below this, a table displays the results of the query. The table has two columns: 'size' and 'order_count'. The first and only row shown is for size 'L' with an order count of 18526.

	size	order_count
▶	L	18526

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

```
SELECT pizza_types.name, SUM(order_details.quantity) AS  
quanties 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 quanties DESC  
LIMIT 5;
```

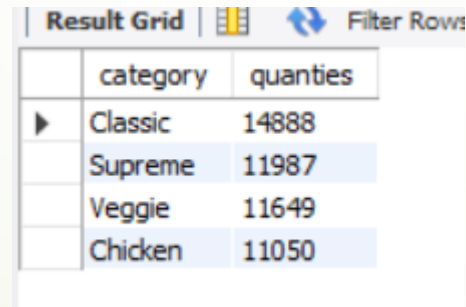


The screenshot shows a database interface with a 'Result Grid' tab. It displays a table with two columns: 'name' and 'quanties'. The table lists the top 5 most ordered pizza types based on their total quantity. The rows are: 'The Classic Deluxe Pizza' (2453), 'The Barbecue Chicken Pizza' (2432), 'The Hawaiian Pizza' (2422), 'The Pepperoni Pizza' (2418), and 'The Thai Chicken Pizza' (2371). The first row is highlighted with a blue arrow icon in the first column.

	name	quanties
▶	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 ordered.

```
SELECT pizza_types.category, SUM(order_details.quantity) AS  
quanties 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.category ORDER BY quanties DESC;
```





The screenshot shows a 'Result Grid' window with a 'Filter Rows' button. It contains a table with two columns: 'category' and 'quanties'. The data is sorted in descending order of quantity. The categories listed are Classic, Supreme, Veggie, and Chicken.

	category	quanties
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

Determine the distribution of orders by hour of the day.

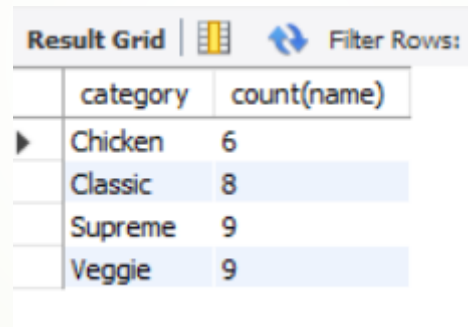
```
SELECT HOUR(order_time), COUNT(order_id)FROM  
orders GROUP BY HOUR(order_time);
```

Result Grid |   Filter Rows:

	HOUR(order_time)	COUNT(order_id)
▶	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

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

```
select category , count(name) from pizza_types group by category;
```

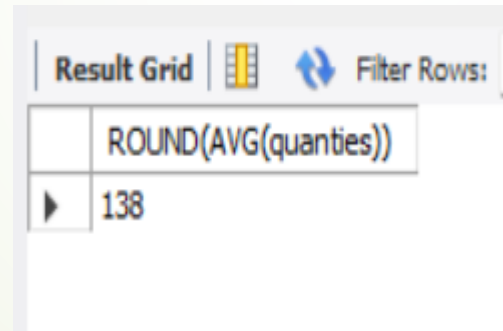


The screenshot shows a database interface with a 'Result Grid' tab. It displays a table with two columns: 'category' and 'count(name)'. The data is as follows:

category	count(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

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

```
SELECT ROUND(AVG(quanties))FROM(SELECT orders.order_date,  
SUM(order_details.quantity) AS quanties FROM orders JOIN  
order_details ON orders.order_id = order_details.order_id GROUP BY  
orders.order_date) AS order_quantity;
```

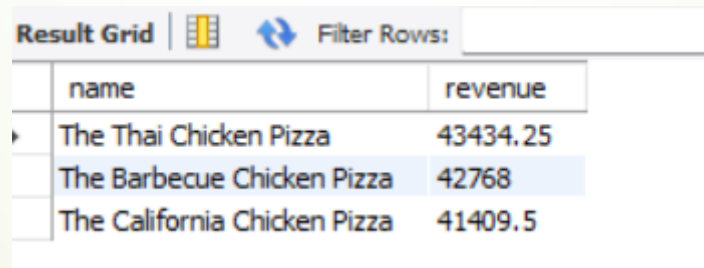


The screenshot shows a database interface with a 'Result Grid' tab. It contains a single row with the expression 'ROUND(AVG(quanties))' in the first column and the value '138' in the second column. There is a 'Filter Rows:' button with a blue double-headed arrow icon.

	ROUND(AVG(quanties))
▶	138

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

```
SELECT pizza_types.name, SUM(pizzas.price *  
order_details.quantity) AS revenue FROM pizza_types JOIN pizzas  
ON pizzas.pizza_type_id = pizza_types.pizza_type_id JOIN  
order_details ON pizzas.pizza_id = order_details.pizza_id GROUP BY  
pizza_types.name ORDER BY revenue DESC LIMIT 3;
```

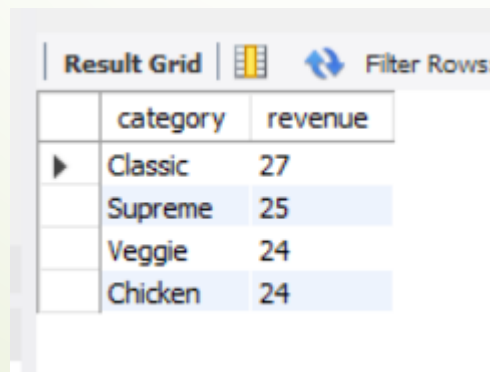


The screenshot shows a 'Result Grid' window with a 'Filter Rows' input field. Below the header, three rows of data are displayed, sorted by revenue in descending order. The first row is 'The Thai Chicken Pizza' with a revenue of 43434.25, the second is 'The Barbecue Chicken Pizza' with 42768, and the third is 'The California Chicken Pizza' with 41409.5.

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.

```
SELECT pizza_types.category, ROUND(SUM(pizzas.price *  
order_details.quantity) / (SELECT ROUND(SUM(pizzas.price *  
order_details.quantity)) AS total_sales FROM order_details JOIN  
pizzas ON pizzas.pizza_id = order_details.pizza_id) * 100) AS revenue  
FROM pizza_types JOIN pizzas ON pizzas.pizza_type_id =  
pizza_types.pizza_type_id JOIN order_details ON pizzas.pizza_id =  
order_details.pizza_id GROUP BY pizza_types.category ORDER BY  
revenue DESC;
```

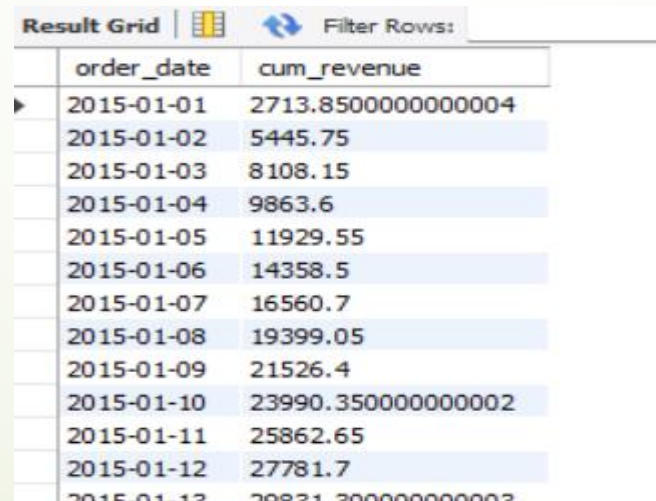


The screenshot shows a 'Result Grid' window with a 'Filter Rows' button. It displays a table with two columns: 'category' and 'revenue'. The data is sorted in descending order of revenue. The categories and their corresponding revenue values are: Classic (27), Supreme (25), Veggie (24), and Chicken (24).

	category	revenue
▶	Classic	27
	Supreme	25
	Veggie	24
	Chicken	24

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(order_details.quantity *
pizzas.price) AS revenue FROM order_details JOIN pizzas ON
order_details.pizza_id = pizzas.pizza_id JOIN orders ON
order_details.order_id = orders.order_id GROUP BY orders.order_date) as
sales;
```



The screenshot shows a 'Result Grid' window with a 'Filter Rows' button. It displays a table with two columns: 'order_date' and 'cum_revenue'. The data shows a steady increase in cumulative revenue from January 1st to January 12th, 2015, with a significant jump on January 10th.

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
2015-01-08	19399.05
2015-01-09	21526.4
2015-01-10	23990.350000000002
2015-01-11	25862.65
2015-01-12	27781.7
2015-01-13	29831.300000000003

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

```
select name,revenue,category from(select category,name, revenue,rank()  
over(partition by category order by revenue desc)as rn from(select  
pizza_types.category,pizza_types.name,sum(order_details.quantity*pizza  
s.price) as revenue from pizza_types join pizzas on  
pizza_types.pizza_type_id =pizzas.pizza_type_id join order_details on  
pizzas.pizza_id=order_details.pizza_id group by  
pizza_types.category,pizza_types.name) as a )as b where rn<=3;
```

name	revenue	category
The Thai Chicken Pizza	43434.25	Chicken
The Barbecue Chicken Pizza	42768	Chicken
The California Chicken Pizza	41409.5	Chicken
The Classic Deluxe Pizza	38180.5	Classic
The Hawaiian Pizza	32273.25	Classic
The Pepperoni Pizza	30161.75	Classic
The Spicy Italian Pizza	34831.25	Supreme
The Italian Supreme Pizza	33476.75	Supreme
The Sicilian Pizza	30940.5	Supreme
The Four Cheese Pizza	32265.70000000065	Veggie
The Mexicana Pizza	26780.75	Veggie
The Five Cheese Pizza	26066.5	Veggie