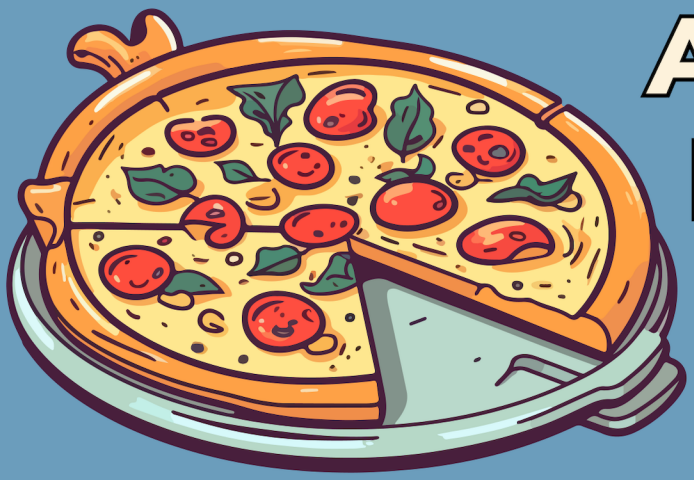
WELCOME TO THE PIZZA SALES ANALYSIS PROJECT



WELCOME TO THE PIZZA SALES ANALYSIS PROJECT!

THIS PROJECT IS DESIGNED TO PROVIDE INSIGHTS INTO THE SALES PERFORMANCE OF OUR PIZZA BUSINESS USING DATA STORED IN AN SQL DATABASE.

WE WILL LEVERAGE SQL TO ANALYZE SALES TRENDS, CUSTOMER PREFERENCES, AND OTHER KEY METRICS THAT CAN HELP US MAKE DATA-DRIVEN DECISIONS TO IMPROVE OUR BUSINESS.

PROJECT GOALS

THE PRIMARY GOALS OF THIS PROJECT ARE TO:

1. UNDERSTAND SALES TRENDS: IDENTIFY PEAK SALES PERIODS AND BEST-SELLING PIZZA VARIETIES.
2. CUSTOMER INSIGHTS: ANALYZE CUSTOMER PURCHASING PATTERNS TO ENHANCE MARKETING STRATEGIES.
3. INVENTORY MANAGEMENT: OPTIMIZE INVENTORY LEVELS BASED ON SALES DATA TO REDUCE WASTE AND ENSURE AVAILABILITY.
4. REVENUE ANALYSIS: ASSESS OVERALL REVENUE PERFORMANCE AND IDENTIFY AREAS FOR GROWTH.

DATA OVERVIEW

THE 4 DATASETS WE WILL BE WORKING WITH INCLUDES DETAILED RECORDS OF PIZZA SALES TRANSACTIONS.

HERE ARE SOME KEY ATTRIBUTES OF THE DATA:

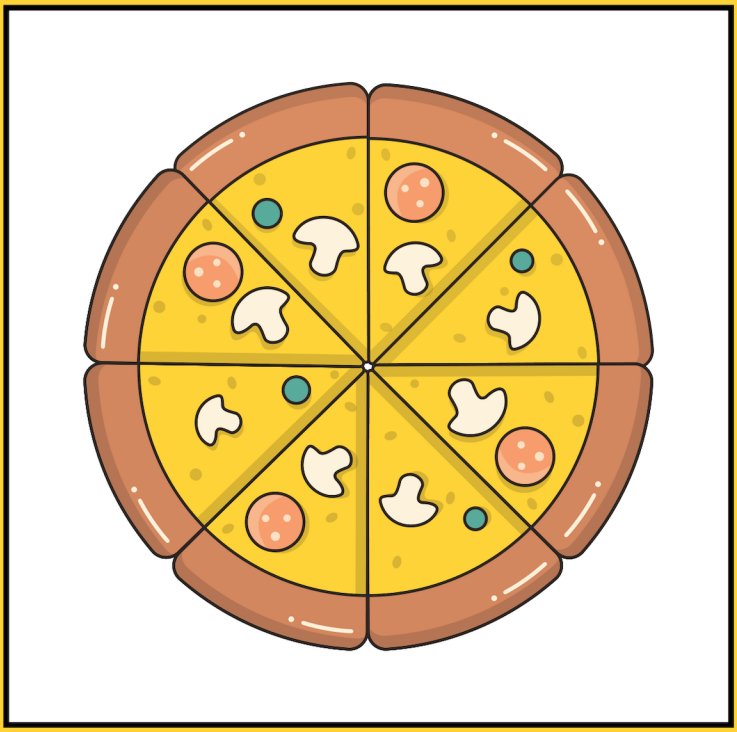
- TRANSACTION ID: UNIQUE IDENTIFIER FOR EACH SALE
- DATE AND TIME: TIMESTAMP OF THE SALE
- PIZZA TYPE: TYPE OF PIZZA SOLD (E.G., MARGHERITA, PEPPERONI, VEGGIE)
- SIZE: SIZE OF THE PIZZA (E.G., SMALL, MEDIUM, LARGE)
- QUANTITY: NUMBER OF PIZZAS SOLD IN THE TRANSACTION
- PRICE: PRICE OF THE PIZZA
- ORDER ID: UNIQUE IDENTIFIER FOR THE ORDER

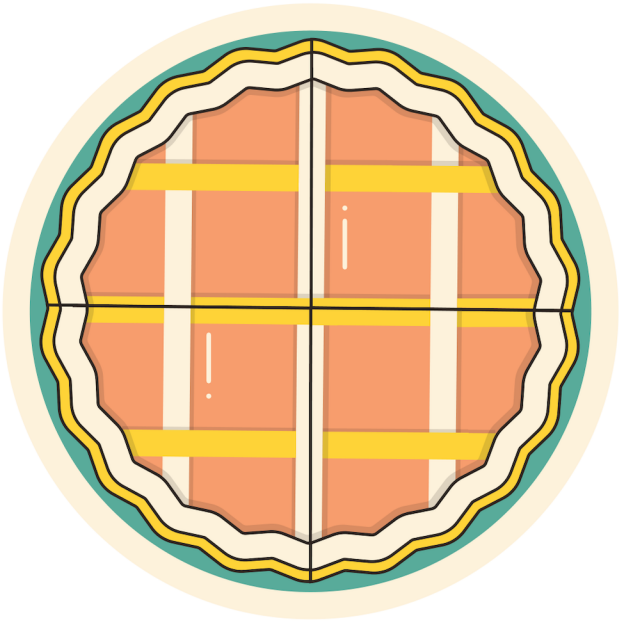
1

1)Creating database

Code:
Create database pizzahut;

Solution: Refresh schema,
A new database called
pizzahut will be available



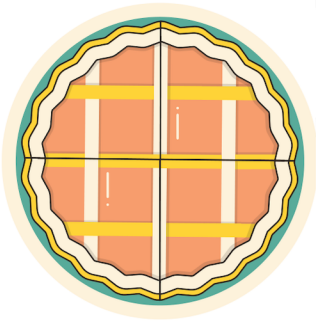


2

Retrieve the total number of orders placed.

ANSWER

select count(order_id) as total_orders from orders;



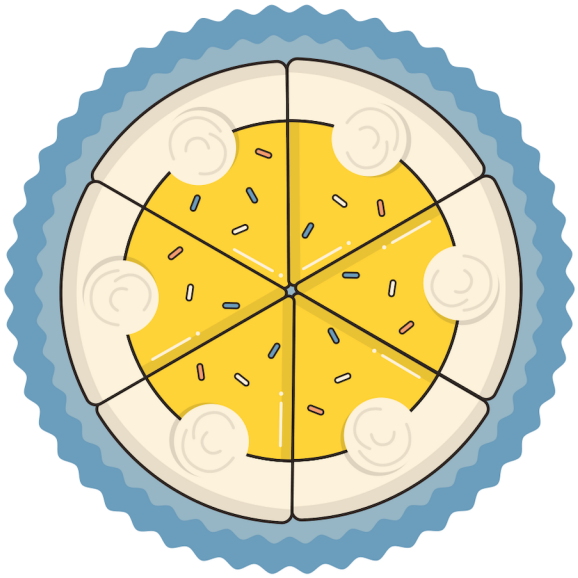
Query 1 | pizza_types | orders_details | SQL File 1*

Limit to 1000 rows

```
1   -- Basic:
2   -- 1) Retrieve the total number of orders placed.
3   select * from orders;
4   select count(order_id) as total_orders from orders;
5
```

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

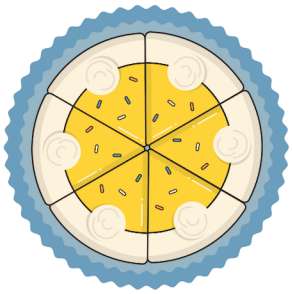
total_orders
21350



3

Calculate the total revenue generated from pizza sales.

ANSWER



Query 1 | pizza_types | orders_details | SQL File 1* | SQL File 2* | pizzas | orders_details

Limit to 1000 rows

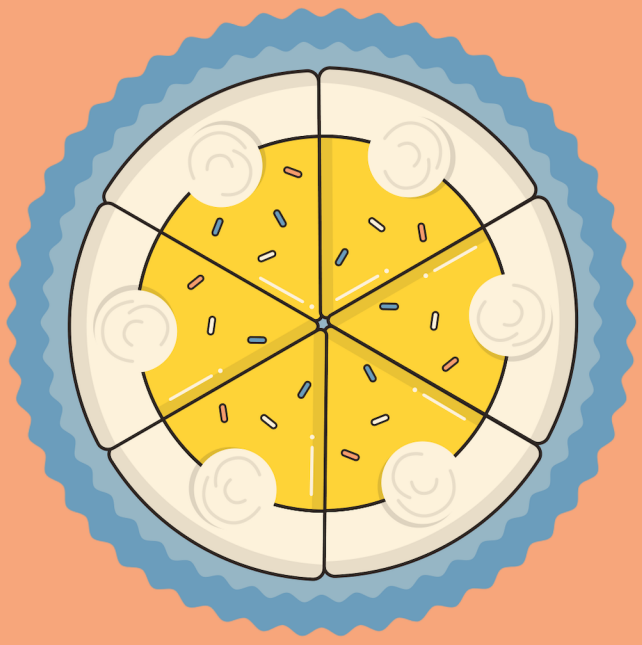
```
3   -- Calculate the total revenue generated from pizza sales.
4
5   SELECT
6       ROUND(SUM(orders_details.quantity * pizzas.price),2) AS total_sales
7   FROM
8       orders_details
9           JOIN
10      pizzas ON pizzas.pizza_id = orders_details.pizza_id
11
```

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

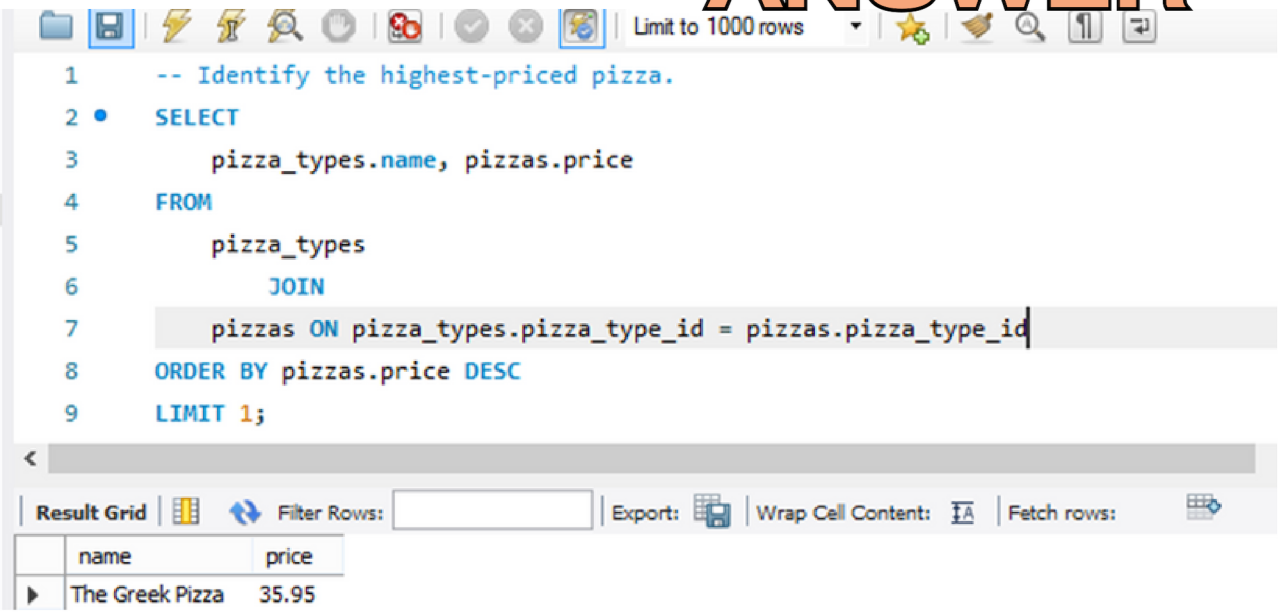
total_sales
115073.45

4

Identify the highest-priced pizza.

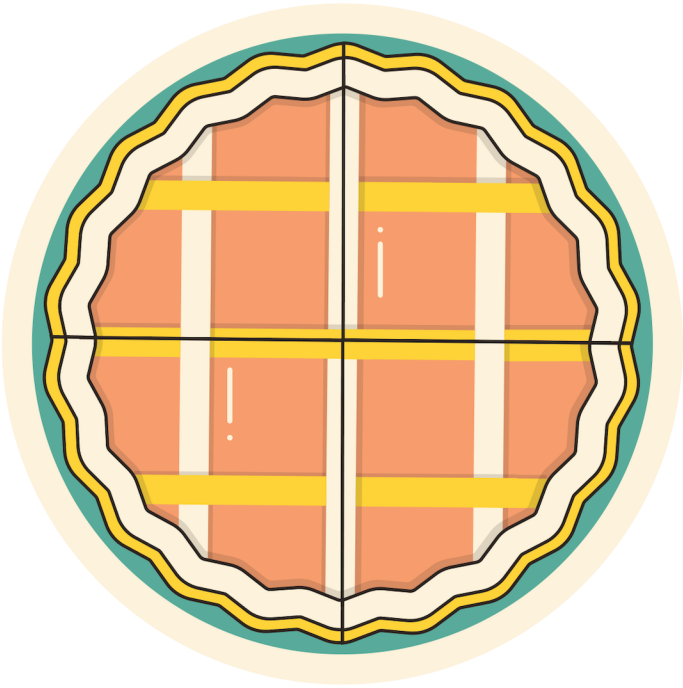


ANSWER


Limit to 1000 rows
-- Identify the highest-priced pizza.
SELECT
pizza_types.name, pizzas.price
FROM
pizza_types
JOIN
pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Fetch rows:
name
price
The Greek Pizza
35.95

5

Identify the most common pizza size ordered.

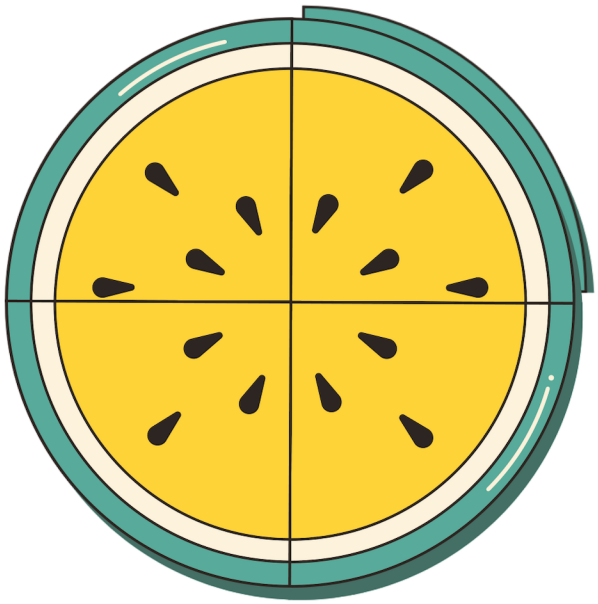


ANSWER

```sql
-- Identify the most common pizza size ordered.
SELECT
    pizzas.size,
    COUNT(orders_details.order_details_id) AS order_count
FROM
    pizzas
        JOIN
    orders_details ON pizzas.pizza_id - order_details_id
GROUP BY pizzas.size
ORDER BY order_count DESC;
```

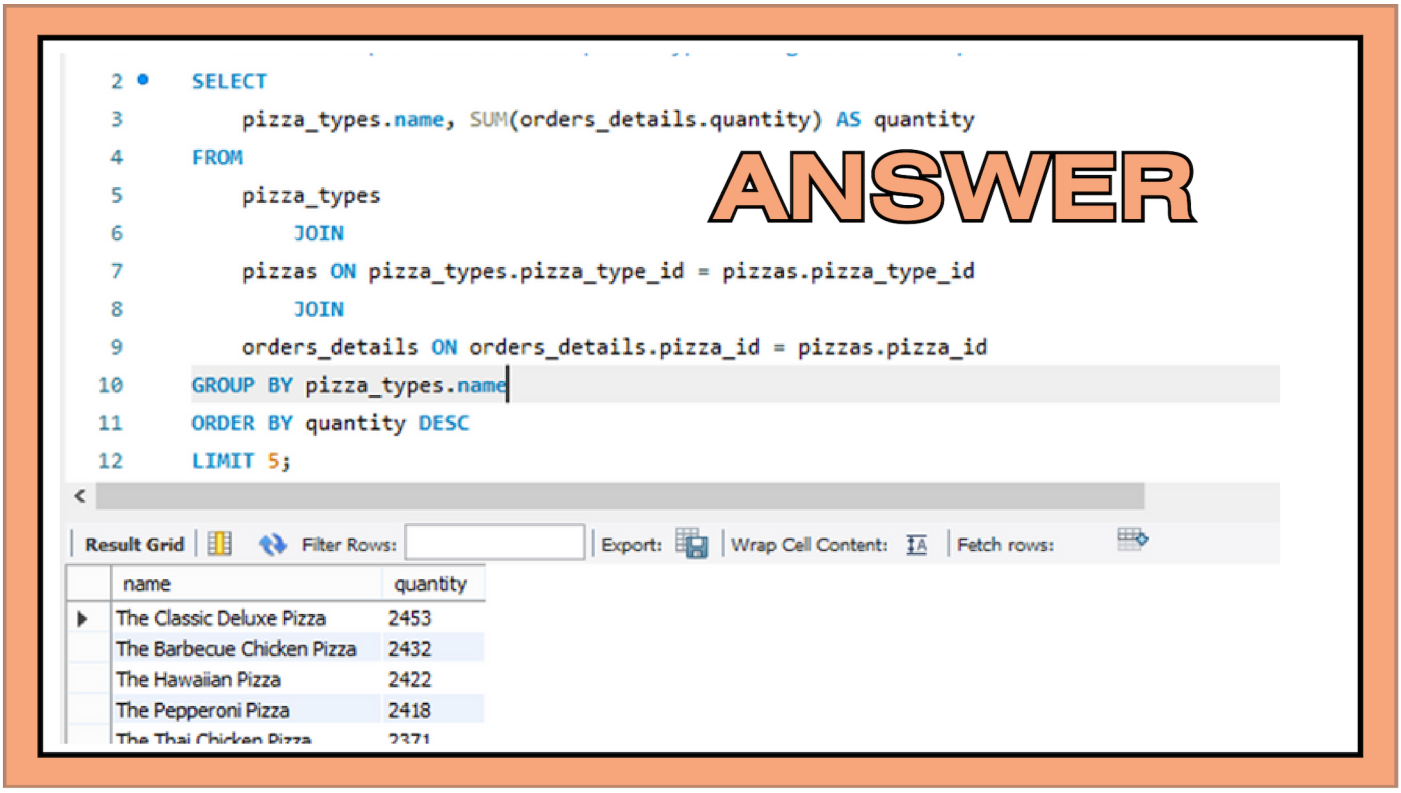
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

size	order_count
S	1529248
L	1481459
M	1481459
XXL	47789
XL	47789



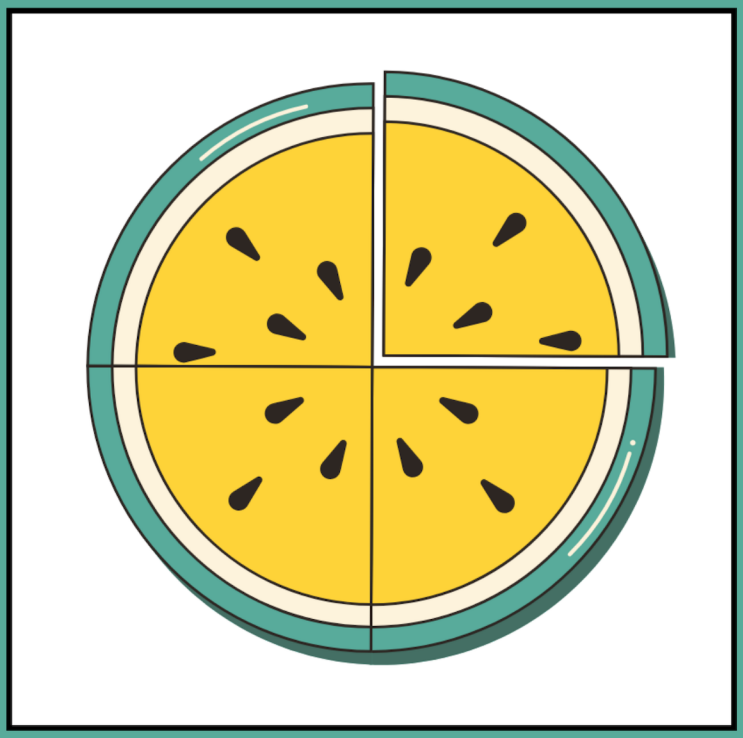
6

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


ANSWER
SELECT
pizza_types.name, SUM(orders_details.quantity) AS quantity
FROM
pizza_types
JOIN
pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN
orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY quantity DESC
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

7

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

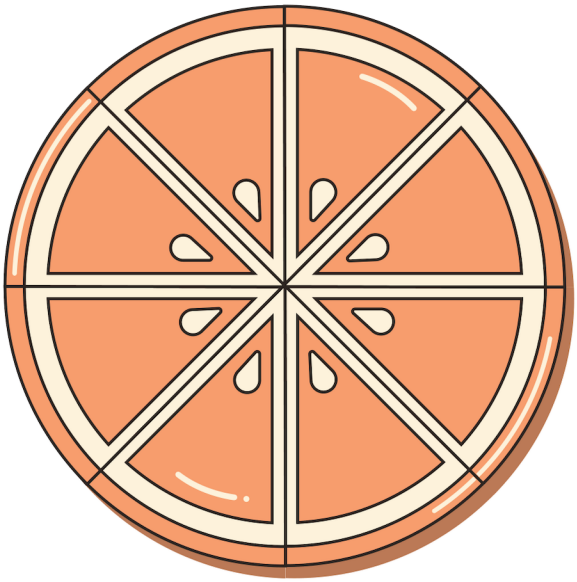


```sql
-- Join the necessary tables to find the total quantity of each pizza category ordered.
SELECT
    SUM(orders_details.quantity) AS Total_Quantity,
    pizza_types.category
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details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

ANSWER

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

Total_Quantity	category
14888	Classic
11987	Supreme
11649	Veggie



8

Determine the distribution of orders by hour of the day.

```sql
-- Determine the distribution of orders by hour of the day.

SELECT
    HOUR(order_time), COUNT(order_id) AS order_count
FROM
    orders
GROUP BY HOUR(order_time) order by hour(order_time) asc
```

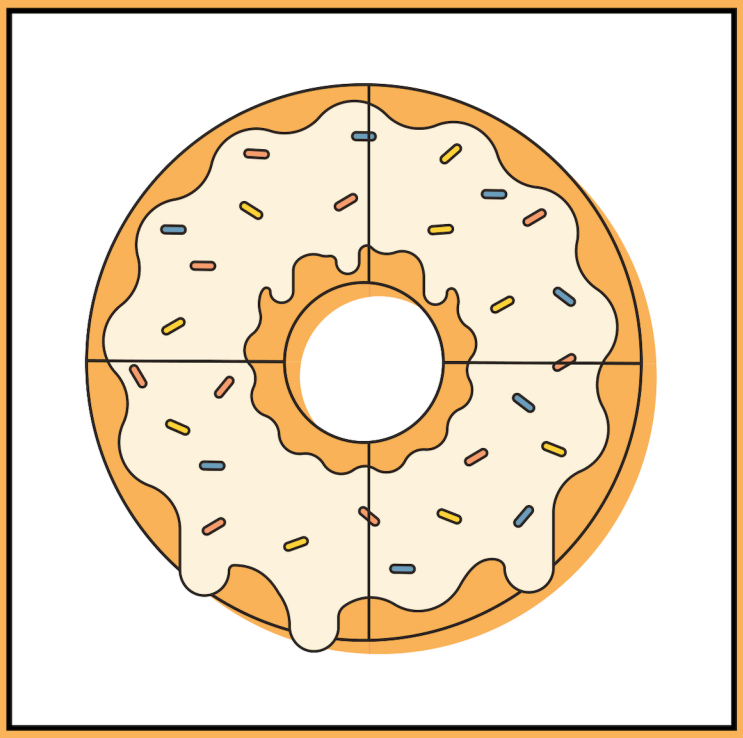
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

HOUR(order_time)	order_count
9	1
10	8
11	1231
12	2520
13	2455
14	1472
15	1468

ANSWER

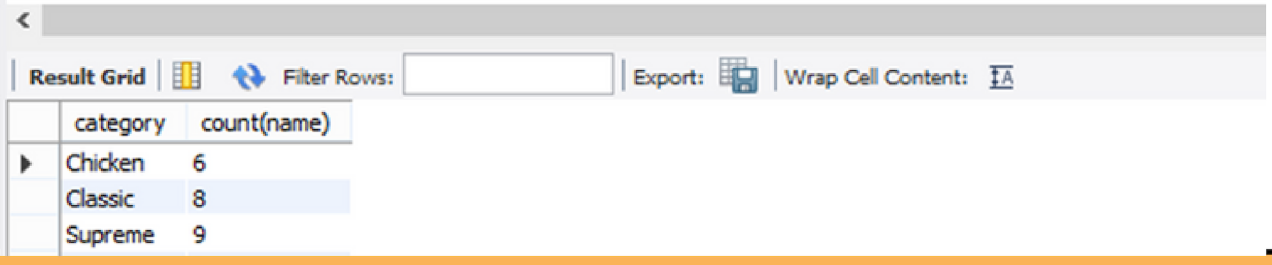
9

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



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

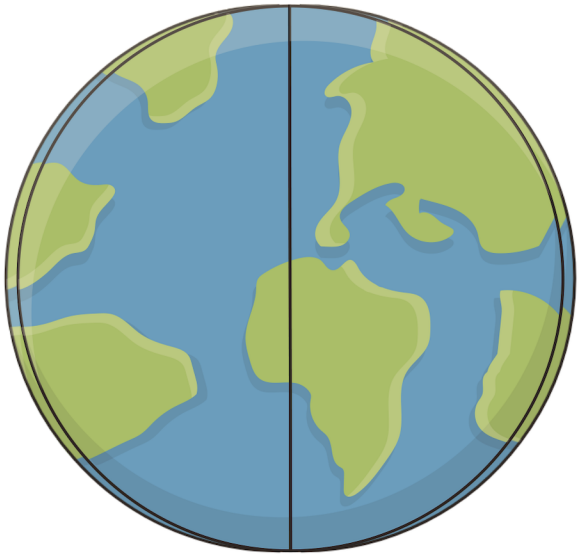
ANSWER



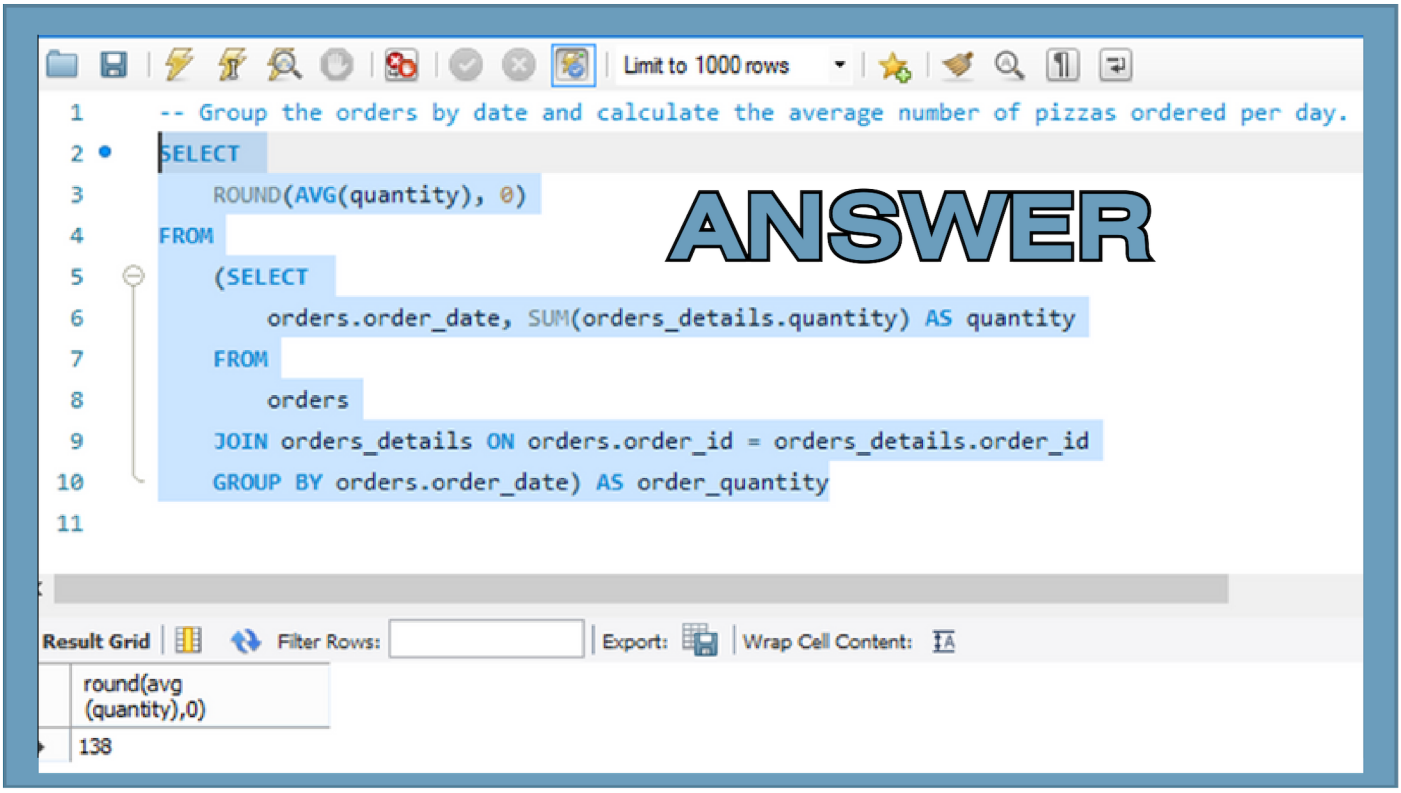
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category	count(name)
Chicken	6
Classic	8
Supreme	9

10

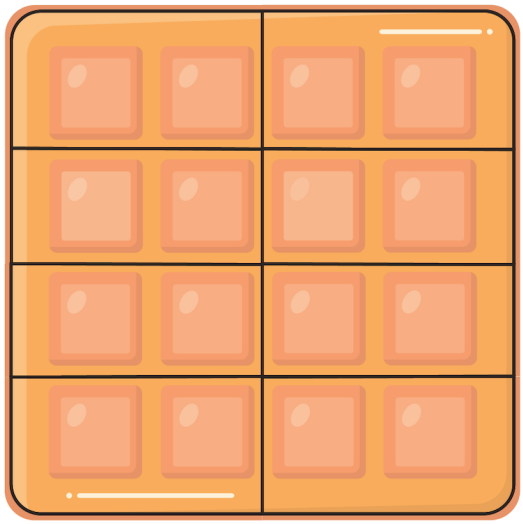


Group the orders by date and calculate the average number of pizzas ordered per day.


Limit to 1000 rows
-- Group the orders by date and calculate the average number of pizzas ordered per day.
SELECT
ROUND(AVG(quantity), 0)
FROM
(SELECT
orders.order_date, SUM(orders_details.quantity) AS quantity
FROM
orders
JOIN orders_details ON orders.order_id = orders_details.order_id
GROUP BY orders.order_date) AS order_quantity
ANSWER
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
round(avg (quantity),0)
138

11

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



```sql
-- Determine the top 3 most ordered pizza types based on revenue.
SELECT
    pizza_types.name,
    SUM(orders_details.quantity * pizzas.price) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3
```

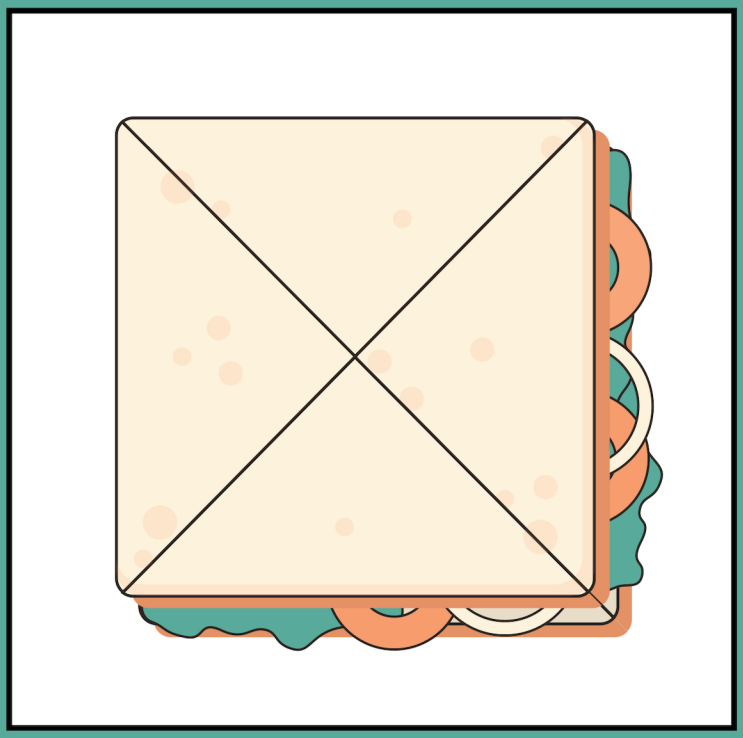
ANSWER

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

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

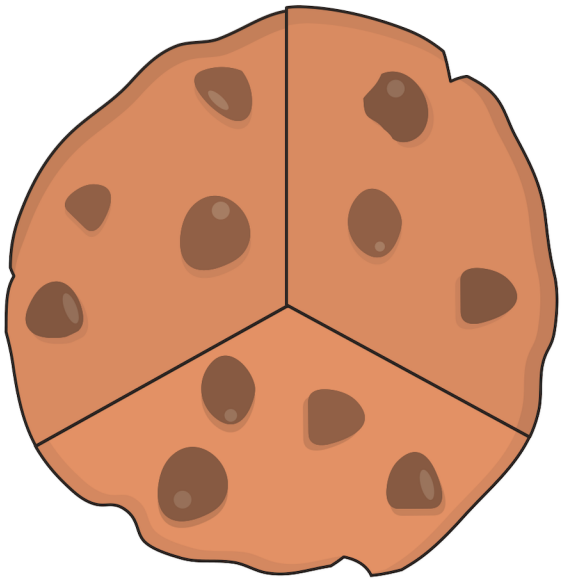
12

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



```sql
SELECT
    pizza_types.category,
    ROUND(SUM(orders_details.quantity * pizzas.price) / (SELECT
                    ROUND(SUM(orders_details.quantity * pizzas.price),
                                2) AS total_sales
                FROM
                    orders_details
                        JOIN
                    pizzas ON pizzas.pizza_id = orders_details.pizza_id) * 100,2)
                AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY category ORDER BY revenue DESC
```

ANSWER



13

Analyze the cumulative revenue generated over time.

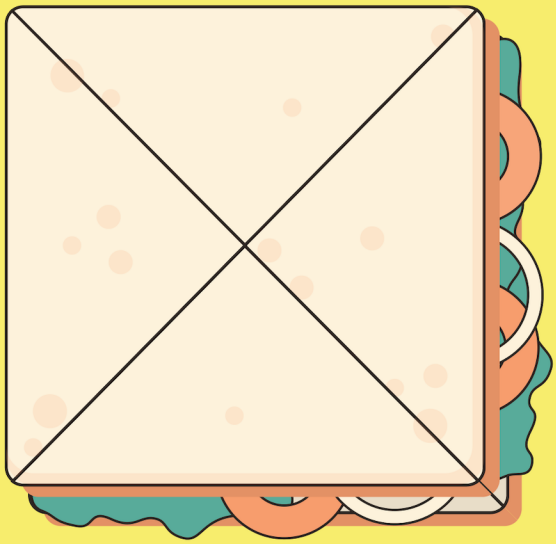
```sql
-- Analyze the cumulative revenue generated over time.

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

ANSWER

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

order_date	cum_revenue
2015-01-01	2713.8500000000004



14

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

```sql
-- Determine the top 3 most ordered pizza types based on revenue for each pizza category.
Select name, revenue 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((orders_details.quantity) * pizzas.price) as revenue
from pizza_types join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join orders_details
on orders_details.pizza_id = pizzas.pizza_id
group by pizza_types.category, pizza_types.name) as a) as b
where rn<=3;
```

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

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