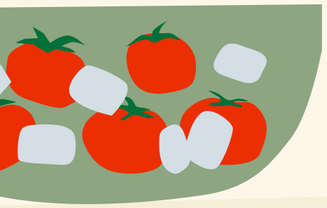
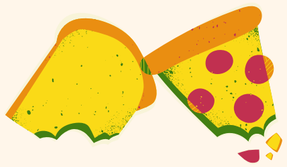

Pizza sales data analysis with sql

Overview

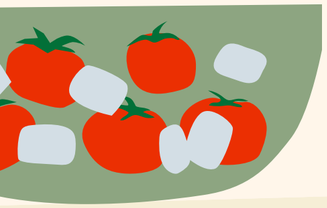
The Pizza Sales Project is a data analysis project aimed at analysing and extract sales data from a pizza restaurant chain. This project utilises SQL for data extraction. The goal of this project is to provide insights and actionable information to help the pizza restaurant chain optimise its operations, improve sales, and enhance customer satisfaction.



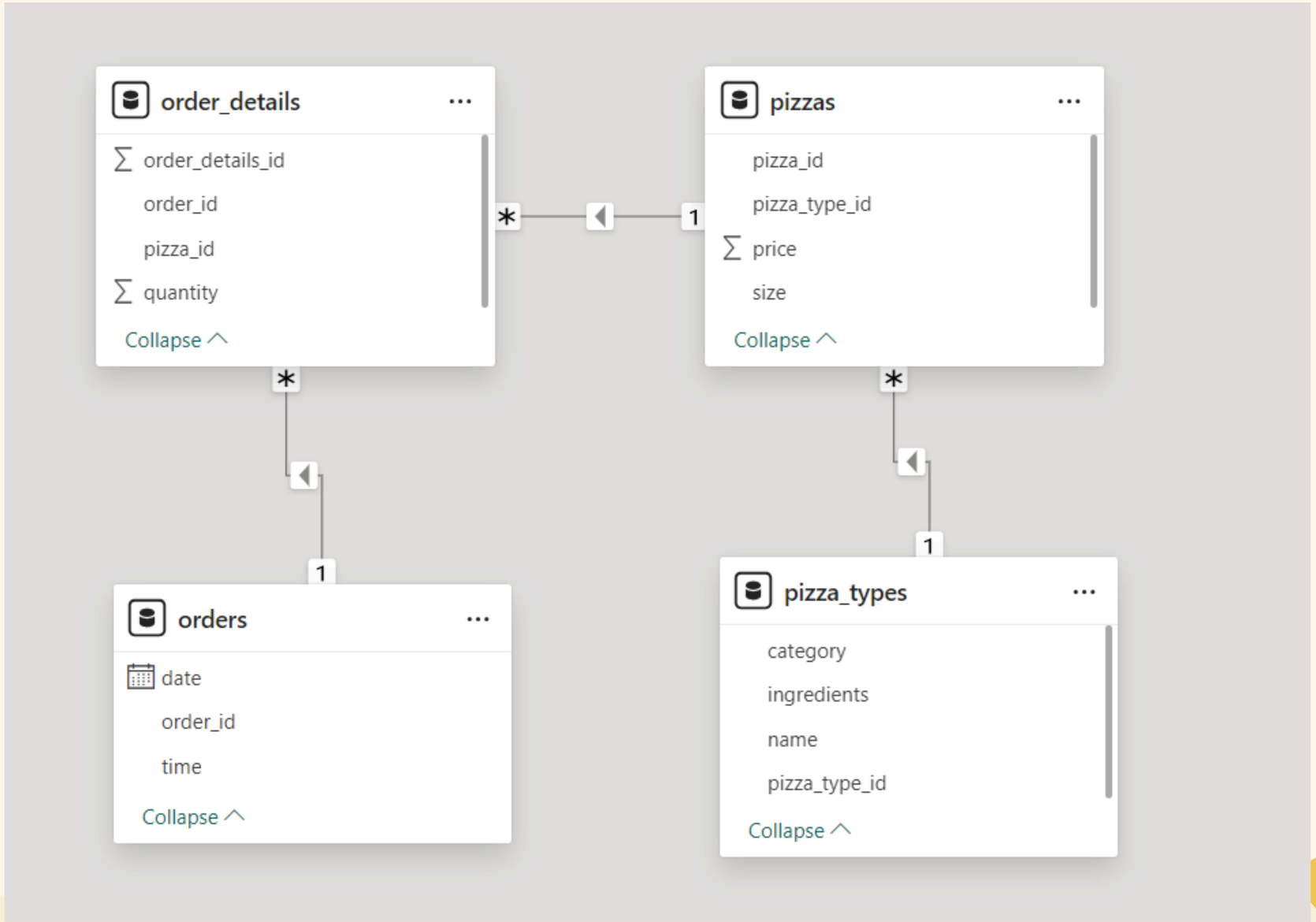
Questions



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

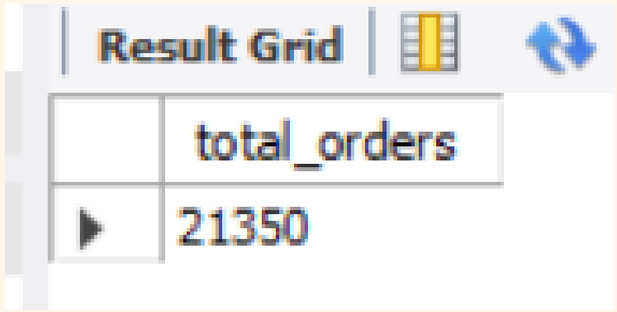


Schema


order_details
order_details_id
order_id
pizza_id
quantity
Collapse
pizzas
pizza_id
pizza_type_id
price
size
Collapse
orders
date
order_id
time
Collapse
pizza_types
category
ingredients
name
pizza_type_id
Collapse
*
1

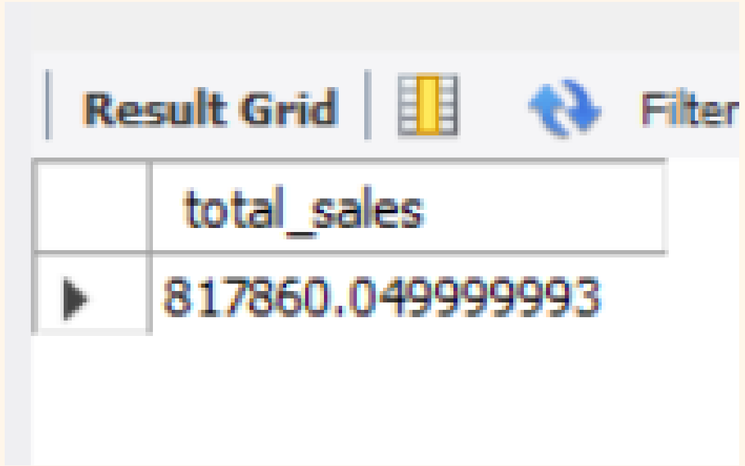
1. Retrieve the total number of orders placed.

```sql
select count(order_id) as total_orders from orders;
```



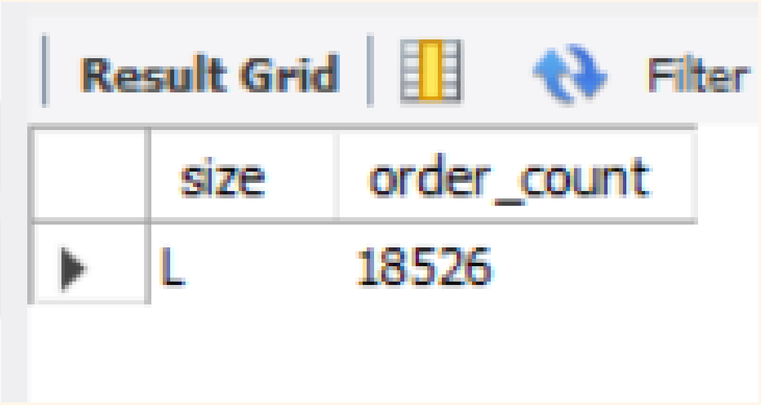
2. Calculate the total revenue generated from pizza sales.

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



3. Identify the most common pizza size ordered.

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



Result Grid | Filter

	size	order_count
▶	L	18526

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

```sql
SELECT
    pizza_types.name,
    SUM(orders_details.quantity) AS order_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 order_quantity DESC
LIMIT 5;
```

Result Grid | Filter Rows:

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

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

```sql
SELECT
    pizza_types.category,
    SUM(orders_details.quantity) AS category_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.category
ORDER BY category_quantity DESC;
```

Result Grid | Filter Rows:

category	category_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

6. Determine the distribution of orders by hour of the day.

```sql
SELECT
    HOUR(order_time) AS hour, COUNT(order_id) AS order_count
FROM
    orders
GROUP BY HOUR(order_time);
```

Result Grid | Filter

hour	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

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

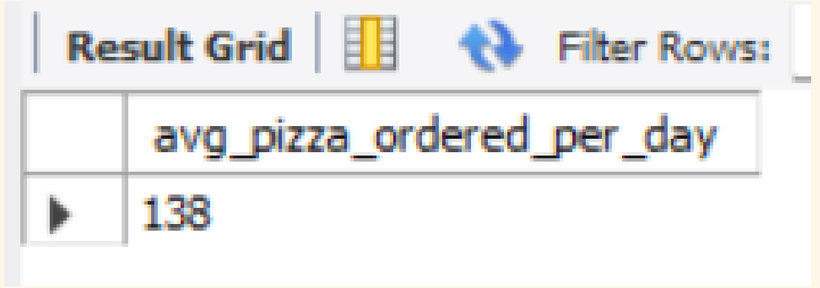
```
select category, count(name) from pizza_types
group by category;
```

Result Grid | Filter Ro

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

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

```sql
SELECT
    ROUND(AVG(quantity), 0) as avg_pizza_ordered_per_day
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;
```



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

```sql
SELECT
    pizza_types.name,
    SUM(ROUND((orders_details.quantity * pizzas.price),
            0)) 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.name
ORDER BY revenue DESC
LIMIT 3;
```

Result Grid | Filter Rows:

name	revenue
The Thai Chicken Pizza	44026
The Barbecue Chicken Pizza	43372
The California Chicken Pizza	42001

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

```sql
SELECT
    pizza_types.category,
    round(SUM(orders_details.quantity * pizzas.price) / (SELECT
    SUM(orders_details.quantity * pizzas.price) AS total_sales
FROM
    orders_details
        JOIN
    pizzas ON orders_details.pizza_id = pizzas.pizza_id) * 100, 3)
            AS contribution_of_each_pizza_type
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
ORDER BY contribution_of_each_pizza_type DESC;
```

Result Grid | Filter Rows:

category	contribution_of_each_pizza_type
Classic	26.906
Supreme	25.456
Chicken	23.955
Veggie	23.683

11. Analyze the cumulative revenue generated over time.

```sql
select order_date,
sum(revenue) over(order by order_date) as cumulative_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;
```

Result Grid | Filter Rows:

order_date	cumulative_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
2015-01-14	32358.700000000004
2015-01-15	34343.50000000001
2015-01-16	36937.65000000001
2015-01-17	39001.75000000001
2015-01-18	40978.600000000006
2015-01-19	43365.75000000001
2015-01-20	45763.65000000001

Result 1

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

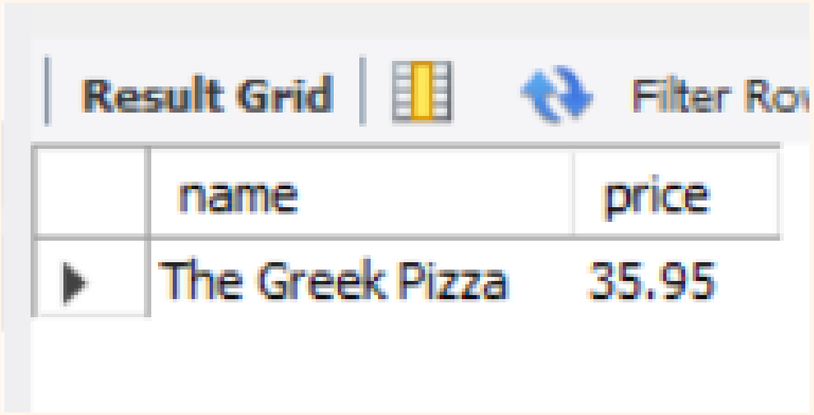
```sql
select name, category, revenue
from
(select category, name, revenue,
rank() over(partition by category order by revenue desc) as rnk
from
(select pizza_types.category, pizza_types.name,
sum((orders_details.quantity) * pizzas.price) as revenue
from pizza_types join pizzas
on pizza_types.pizza_type_id = 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 rnk <= 3;
```

Result Grid | Filter Rows: | Export:

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

13. 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;
```



Thank You