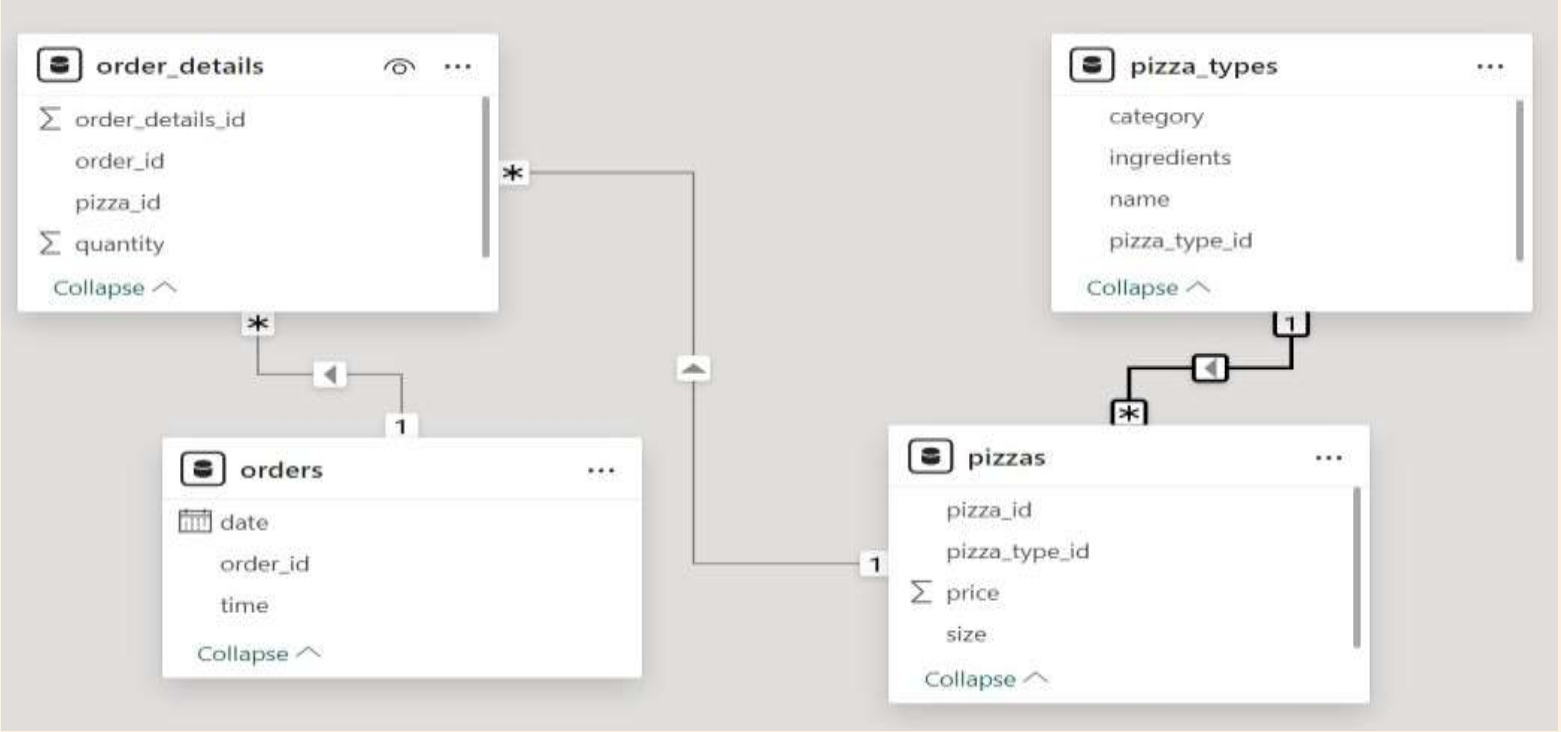
# Pizza
# Sales Analysis

In this project, I have utilized SQL to solve queries that are  related to Pizza sales.


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

# Basic

# Q1. Retrieve the total number of orders placed.

```
SELECT * FROM orders;
SELECT count(order_id) FROM orders;
```

| | count(order_id) |
|---|---|
| ▶ | 21350 |

# Q2. Calculate the total revenue generated from pizza sales.

```sql
SELECT * FROM pizzas;
SELECT * FROM order_details;
SELECT ROUND(SUM(price * quantity)) as Total_revenue
FROM pizzas p JOIN order_details od ON p.pizza_id = od.pizza_id;
```

| | Total_revenue |
|---|---|
| ▶ | 817860 |

# Q3. Identify the highest-priced pizza

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

| | name | price |
|---|---|---|
| ▶ | The Greek Pizza | 35.95 |

# Q4. Identify the most common pizza size ordered.

```
SELECT * FROM pizzas;
SELECT * FROM order_details;
SELECT size, COUNT(order_details_id) as order_count
FROM pizzas p JOIN order_details od ON p.pizza_id = od.pizza_id
GROUP BY size
ORDER BY order_count DESC;
```

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

# Q5. Find the category-wise distribution of pizzas.

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

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

# Intermediate

# Q6. Join the necessary tables to find the total quantity of each pizza category ordered.

```
SELECT category, SUM(quantity) as Total_quantity
FROM pizzas p JOIN order_details od ON p.pizza_id = od.pizza_id
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY category
ORDER BY Total_quantity DESC;
```

| | category | Total_quantity |
|---|---|---|
| ▶ | Classic | 14888 |
| | Supreme | 11987 |
| | Veggie | 11649 |
| | Chicken | 11050 |

# Q7. Determine the distribution of orders by hour of the day.

```
SELECT hour(time) as Hours, count(order_id) as Total_count
FROM orders
GROUP BY Hours
ORDER BY Total_count DESC;
```

| Hours | Total_count |
|---|---|
| 12 | 2520 |
| 13 | 2455 |
| 18 | 2399 |
| 17 | 2336 |
| 19 | 2009 |
| 16 | 1920 |

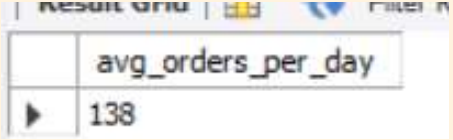
# Q8. List the top 5 most ordered pizza types along with their quantities

```
SELECT name, sum(quantity) as Total_Quantity
FROM pizzas p JOIN order_details od ON p.pizza_id = od.pizza_id
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY name
ORDER BY Total_quantity DESC
LIMIT 5;
```

| name | Total_Quantity |
|---|---|
| The Classic Deluxe Pizza | 2453 |
| The Barbecue Chicken Pizza | 2432 |
| The Hawaiian Pizza | 2422 |
| The Pepperoni Pizza | 2418 |
| The Thai Chicken Pizza | 2371 |

# Q9. Group the orders by date and calculate the average number of pizzas ordered per day.

```
SELECT Round(avg(Total_quantity),0) as avg_orders_per_day
FROM
(SELECT date, SUM(od.quantity) as Total_quantity
FROM orders o JOIN order_details od ON o.order_id = od.order_id
GROUP BY date) as order_quantity;
```



| | avg_orders_per_day |
|---|---|
| ▶ | 138 |

# Q10. Determine the top 3 most ordered pizza types based on revenue.

```sql
SELECT name, round(sum(quantity * price)) as revenue
FROM pizzas p JOIN order_details od ON p.pizza_id = od.pizza_id
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY name
ORDER BY revenue DESC
LIMIT 3;
```

| name | revenue |
|---|---|
| The Thai Chicken Pizza | 43434 |
| The Barbecue Chicken Pizza | 42768 |
| The California Chicken Pizza | 41410 |

# Advance

# Q11. Calculate the percentage contribution of each pizza type to total revenue.

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

| | category | revenue |
|---|---|---|
| ▶ | Classic | 27 |
| | Supreme | 25 |
| | Veggie | 24 |
| | Chicken | 24 |

# Q12. Analyze the cumulative revenue generated over time.

```
SELECT date, sum(revenue) OVER(ORDER BY date) AS cum_revenue
FROM
    (SELECT date,
    round(SUM(quantity * price)) as revenue
    FROM order_details od JOIN pizzas p
    ON od.pizza_id = p.pizza_id
    JOIN orders o
    ON o.order_id = od.order_id
    GROUP BY date) AS sales;
```

| date | cum_revenue |
|---|---|
| 2015-01-01 | 2714 |
| 2015-01-02 | 5446 |
| 2015-01-03 | 8108 |
| 2015-01-04 | 9863 |
| 2015-01-05 | 11929 |
| 2015-01-06 | 14358 |
| 2015-01-07 | 16560 |
| 2015-01-08 | 19398 |

# Q13. Determine the top 3 most ordered pizza types based on revenue for each pizza category.

```
SELECT name, category, revenue
FROM
    (SELECT name, category, revenue,
    RANK() OVER(partition by category ORDER BY revenue DESC) as rn
    FROM
        (SELECT category, name, round(SUM(quantity * price)) as revenue
        FROM pizza_types pt JOIN pizzas p ON pt.pizza_type_id = p.pizza_type_id
        JOIN order_details od ON p.pizza_id = od.pizza_id
        GROUP BY category, name) as a) as b
        WHERE rn<=3;
```

| name | category | revenue |
|---|---|---|
| The Thai Chicken Pizza | Chicken | 43434 |
| The Barbecue Chicken Pizza | Chicken | 42768 |
| The California Chicken Pizza | Chicken | 41410 |
| The Classic Deluxe Pizza | Classic | 38180 |
| The Hawaiian Pizza | Classic | 32273 |
| The Pepperoni Pizza | Classic | 30162 |
| The Spicy Italian Pizza | Supreme | 34831 |
| The Italian Supreme Pizza | Supreme | 33477 |
| The Sicilian Pizza | Supreme | 30940 |

# Thank You