# PIZZA SALES – SQL PROJECT

Author – GAURAV PRAJAPATI

Project Link - <u>Link</u>

GitHub - Link

LinkedIn – <u>Link</u>

# **Objective:**

To perform comprehensive data analysis on pizza sales using structured SQL queries to derive insights on order patterns, revenue generation, and customer preferences.

# **Scope of Work:**

# **Basic Level Analysis:**

- Retrieved the **total number of orders** placed in the dataset.
- Calculated **total revenue** from pizza sales.
- Identified the highest-priced pizza.
- Found the **most common pizza size** ordered.
- Listed the top 5 most ordered pizza types by quantity.

### **Intermediate Level Analysis:**

- Joined multiple tables to compute **total quantity ordered** per pizza category.
- Analyzed order distribution by hour, identifying peak times.
- Performed a category-wise pizza distribution using joins.
- Grouped data by date to calculate the average number of pizzas ordered per day.
- Identified top 3 pizza types by revenue.

#### **Advanced Level Analysis:**

- Calculated percentage revenue contribution of each pizza type.
- Analyzed cumulative revenue over time for trend observation.
- Found top 3 revenue-generating pizza types within each category.

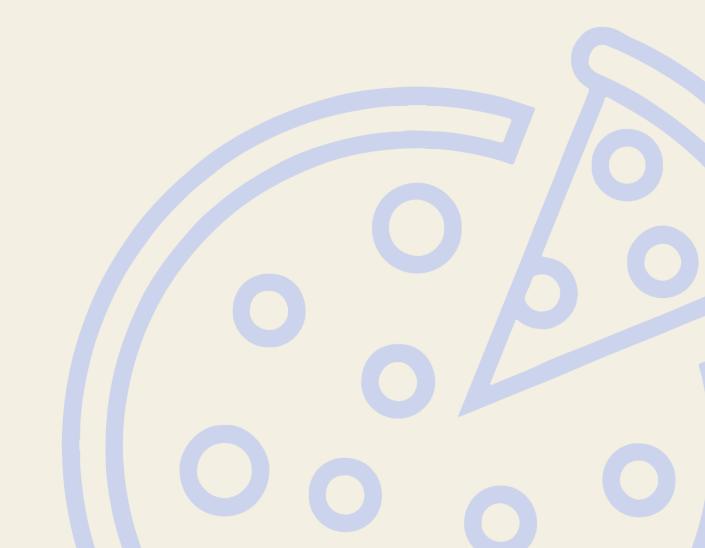
### **Key Learnings:**

• Strengthened my understanding of **SQL fundamentals** such as SELECT, GROUP BY, ORDER BY, JOIN, and AGGREGATE FUNCTIONS.

- Learned to write **complex queries** using multiple table joins and subqueries.
- Gained experience in data grouping, date/time manipulation, and revenue-based ranking.
- Understood how to derive actionable insights from raw data for business decision-making.
- Improved analytical thinking by designing solutions to real-world business questions.

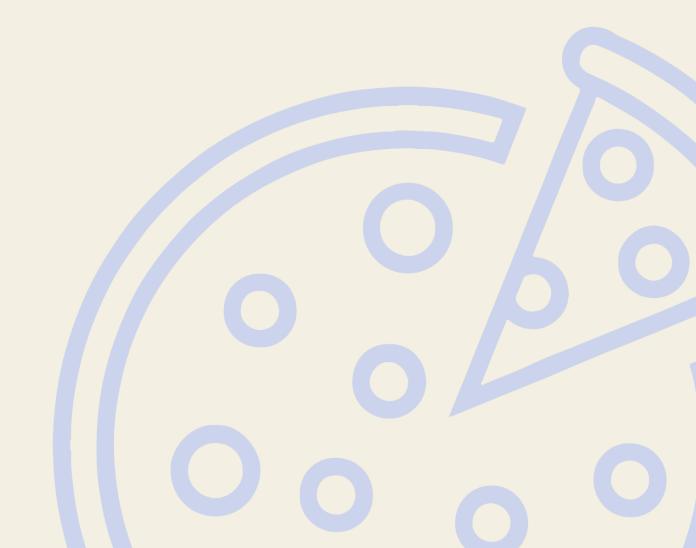
#### **Conclusion:**

This project provided a strong foundation in using **SQL** for data analysis. By working through different complexity levels, I developed both technical and problem-solving skills essential for data-centric roles. The project simulated a real-world business analytics scenario, and I'm now confident in handling SQL-based analytical tasks independently.



#### Problem Statements –

- 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.



1. Retrieve the total number of orders placed.

```
-- 1. Retrieve the total number of orders placed.

SELECT

COUNT(order_id) AS Total_Orders

FROM

orders;
```

2. Calculate the total revenue generated from pizza sales.

```
-- 2. Calculate the total revenue generated from pizza sales

SELECT

ROUND(SUM(orders_details.quantiy * pizzas.price),

2) AS total_sales

FROM

orders_details

JOIN

pizzas ON pizzas.pizza_id = orders_details.pizza_id
```

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.

```
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;
```

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

```
SELECT
    pizza_types.category,
    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.category
ORDER BY quantity DESC;
```

7. 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);
```

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

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

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

```
SELECT
   ROUND(AVG(quantity), 0) AS 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;
```

10. 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;
```

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

```
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
   orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY revenue DESC;
```

12. Analyze the cumulative revenue generated over time.

```
select order_date ,
round(sum(revenue) over (order by order_date),2) as cumm_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;
```

13.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 pizza_ranks
    from
( select pizza_types.category , pizza_types.name ,
    round(sum((orders_details.quantity)*pizzas.price),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 pizza_types.category , pizza_types.name ) as a) as b
    where pizza_ranks <= 3 ;</pre>
```