# ABOUT OUR PIZZA RESTO
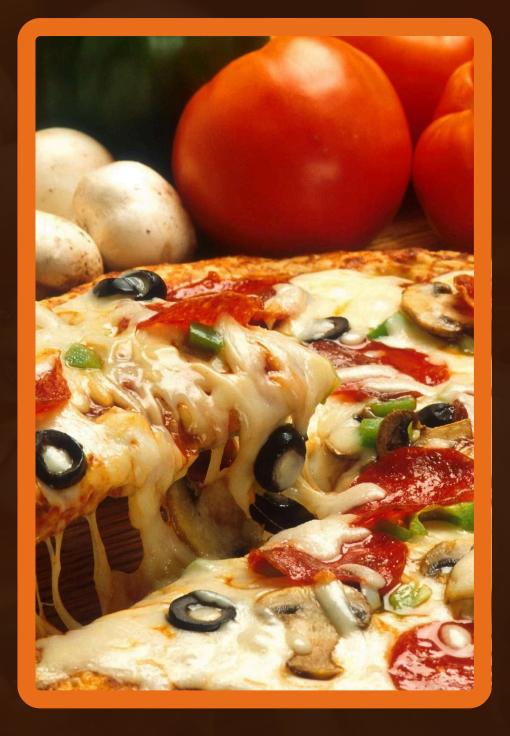
## Our Passion for Pizza

This analysis focuses on the Pizzahut database, examining key insights related to orders, revenue, and pizza sales performance. The dataset includes tables such as orders, order_details, pizzas, and pizza_types to derive meaningful business metrics.
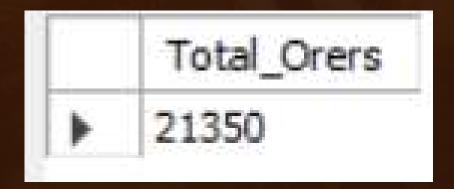
Key highlights:

- Total Orders: Calculation of the number of orders placed.
- Revenue Analysis: Aggregation of total revenue generated from pizza sales.
- Top-Selling & Highest-Priced Pizzas: Identifying the most popular and expensive pizzas.
- Order Trends: Analyzing order frequency over time.

# Retrieve the total number of orders placed.

This SQL query calculates the total number of orders placed in the orders table, providing an overall count of transactions.

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

| | Total_Orers |
|---|---|
| ▶ | 21350 |

# Identify the highest-priced pizza.

This SQL query calculates the total revenue generated from pizza sales by multiplying the quantity of each pizza ordered by its price and summing the results, rounding to two decimal places.

```sql
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;
```

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

# Calculate the total revenue generated from pizza sales.

This query calculates the total revenue from pizza sales by summing the product of pizza quantity and price, ensuring the result is rounded to two decimal places.

```sql
select
    round(sum(order_details.quantity * pizzas.price),
            2) as total_revenue
from
    order_details
        join
    pizzas ON pizzas.pizza_id = order_details.pizza_id;
```

| total_revenue |
|---|
| 817860.05 |

# Identify the most common pizza size ordered.

This query counts the number of orders for each pizza size, grouping by size and sorting the results in descending order to identify the most ordered pizza size.

```sql
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;
```

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

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

This query identifies the top 5 most ordered pizza types by summing the total quantity ordered for each type and sorting the results in descending order.

```sql
    pizza_types.name, sum(order_details.quantity) as quantity
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.name
order by quantity desc
limit 5;
```

| name | quantity |
|------|----------|
| 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.

This query calculates the total quantity of pizzas ordered for each category, grouping by category and sorting the results in descending order to show the most popular categories.

```sql
select
    pizza_types.category, sum(order_details.quantity) as quantity
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
order by quantity desc;
```

| category | quantity |
|----------|----------|
| Classic  | 14888    |
| Supreme  | 11987    |
| Veggie   | 11649    |
| Chicken  | 11050    |

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

This query counts the number of orders placed each hour, grouping by hour to analyze order distribution throughout the day.

```sql
select
    hour(order_time) as hour, count(order_id) as order_count
from
    orders
group by hour;
```

| 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           |

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

This query counts the number of pizza types in each category, grouping by category to show how many different pizzas belong to each type.

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

| 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.

This query calculates the average number of pizzas ordered per day, by first summing the total pizzas ordered each day and then averaging that value across all days, rounding to the nearest whole number.

```sql
select
    round(avg(quantity), 0) as average_pizza_ordered_per_day
from
    (select
        orders.order_date, sum(order_details.quantity) as quantity
    from
        orders
    join order_details ON orders.order_id = order_details.order_id
    group by orders.order_date) as order_quantity;
```

| average_pizza_ordered_per_day |
|---|
| 138 |

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

This query identifies the top 3 highest revenue-generating pizza types by calculating total sales for each pizza type and sorting them in descending order.

```sql
select pizza_types.name,
sum(order_details.quantity * pizzas.price) as revenue
from pizza_types
join pizzas on pizzas.pizza_type_id = pizza_types.pizza_type_id
join order_details on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.name order by revenue desc limit 3;
```

| 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.

This query calculates the percentage of total revenue contributed by each pizza category, sorting the results in descending order to show the highest-earning categories.

```sql
select pizza_types.category,
round(sum(order_details.quantity * pizzas.price) / (select
    round(sum(order_details.quantity * pizzas.price),
        2) as total_revenue
from
    order_details
        join
    pizzas ON pizzas.pizza_id = order_details.pizza_id) *100,2) as revenue
from pizza_types
join pizzas on pizzas.pizza_type_id = pizza_types.pizza_type_id
join order_details on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.category order by revenue desc;
```

| category | revenue |
|----------|---------|
| Classic | 26.91 |
| Supreme | 25.46 |
| Chicken | 23.96 |
| Veggie | 23.68 |

# Analyze the cumulative revenue generated over time.

This query calculates the cumulative revenue over time, summing daily revenue in chronological order to track the total earnings growth.

```sql
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 orders.order_id = order_details.order_id
group by orders.order_date) as sales;
```

| 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 |

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

This query identifies the top 3 highest revenue-generating pizzas in each category, ranking them within their respective categories based on total revenue.

```sql
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((order_details.quantity) * pizzas.price) as revenue

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, pizza_types.name) as a) as b

where rn <= 3;
```

| 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 |
| The Pepperoni Pizza | 30161.75 |
| The Spicy Italian Pizza | 34831.25 |
| The Italian Supreme Pizza | 33476.75 |
| The Sicilian Pizza | 30940.5 |
| The Four Cheese Pizza | 32265.7000 |
| The Mexicana Pizza | 26780.75 |
| The Five Cheese Pizza | 26066.5 |

# "Pizza Hut Sales & Revenue Analysis: Key Insights & Trends"

## Pizza Hut Sales & Revenue Analysis

This analysis provides insights into total orders, revenue generation, top-selling pizzas, and category-wise performance. It identifies the most profitable pizzas, peak ordering times, and revenue contributions by category. Additionally, cumulative revenue trends help track business growth over time. These insights can assist in optimizing pricing, improving inventory management, and enhancing overall sales strategies.

Furthermore, the analysis highlights customer preferences based on pizza size, type, and category, helping to identify high-demand products. By understanding order patterns and revenue distribution, businesses can make data-driven decisions to enhance marketing strategies, streamline operations, and maximize profitability.

Pizza Resto Presentation

# THANK YOU FOR ATTENTION

Jeevan.choudhary2111@gmail.com

https://github.com/jeevan2111