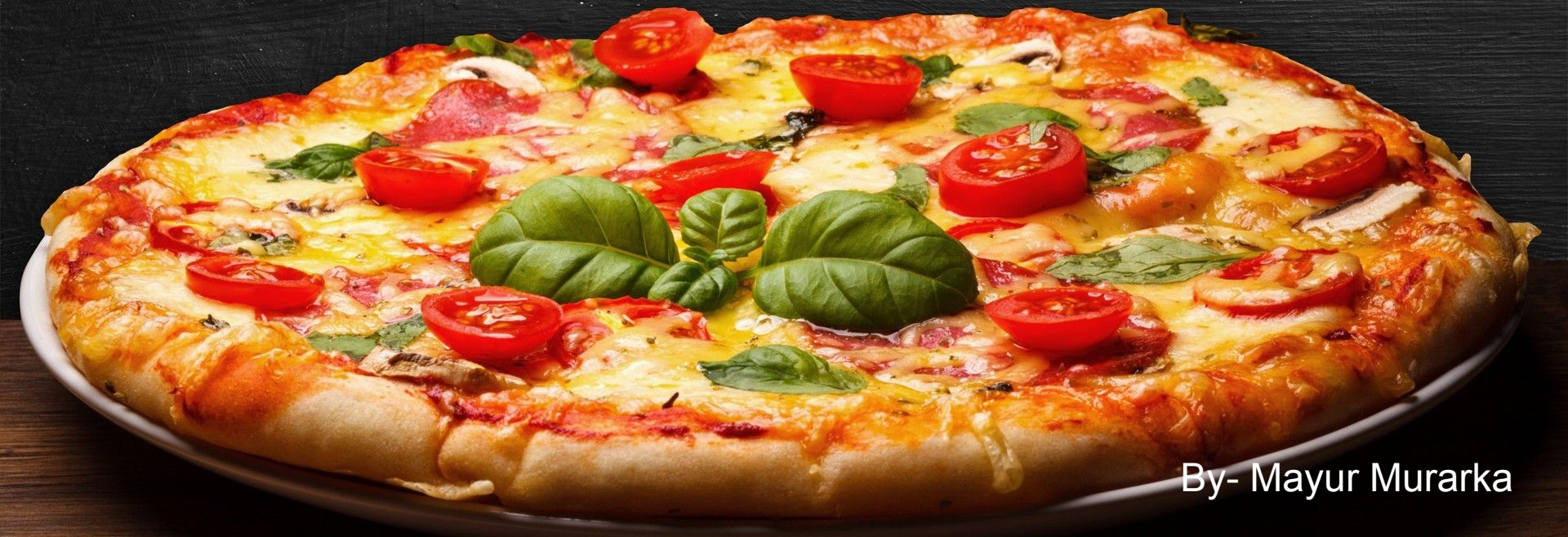
PIZZAHUT

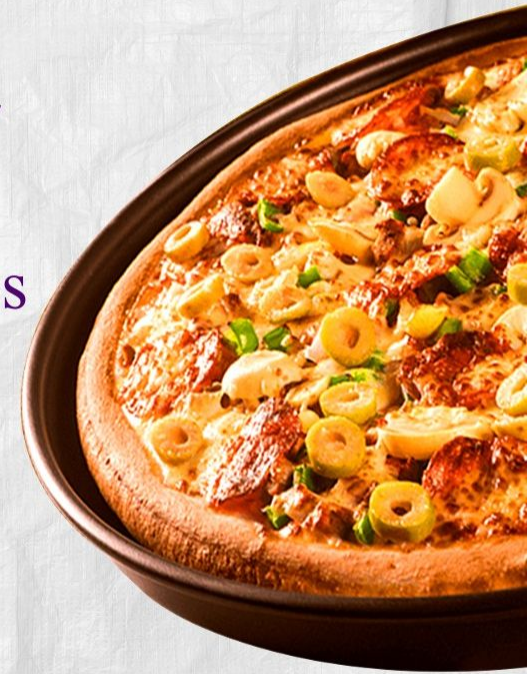
# SQL PROJECT ON PIZZA SALES



By- Mayur Murarka

# Project Overview

- **Objective:** Analyze pizza sales data using SQL
- **Goal**: Generate insights that help understand performance, trends, and customer preferences
- **Tools Used:** SQL(My SQL)



# Dataset Description

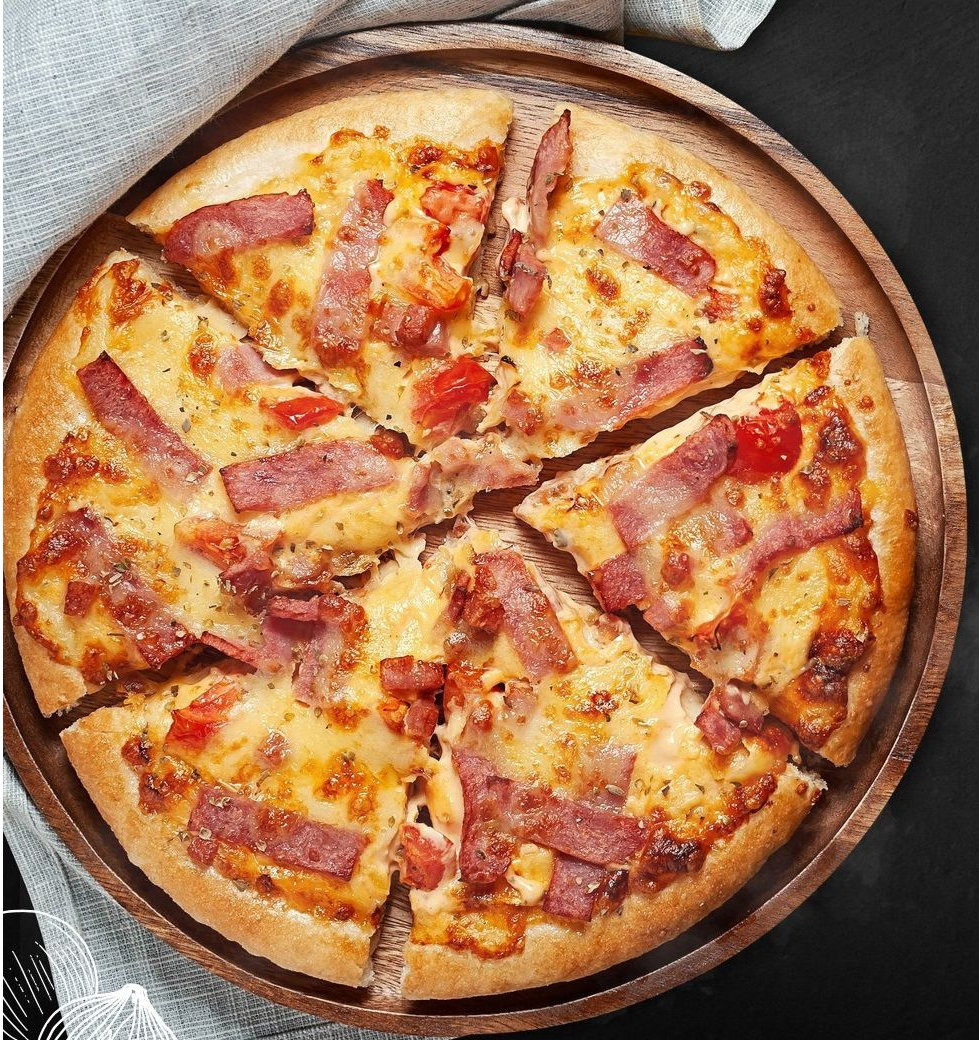


- **Tables Used:** Orders, Order Details, Pizzas, Pizza Types
- **Rows:** 48620 (from order_details.csv)
- **Columns:** i) **orders.csv** – Order ID, date, time

  ii) **order_details.csv** – Order ID, Pizza ID, quantity

  iii) **pizzas.csv** – Pizza ID, name, size, price

  iii) **pizza_types.csv** – Pizza name, category, ingredients



# QUESTION

PIZZAHUT

## BASIC:

- Retrieve the total number of orders placed.
- Calculate the total revenue generated from pizza sales.
- Identify the highest-priced pizza.
- Identify the most common pizza size ordered.
- List the top 5 most ordered pizza types along with their quantities.

## INTERMEDIATE:

- Join the necessary tables to find the total quantity of each pizza category ordered.
- Determine the distribution of orders by hour of the day.
- Join relevant tables to find the category-wise distribution of pizzas.
- Group the orders by date and calculate the average number of pizzas ordered per day.
- Determine the top 3 most ordered pizza types based on revenue.

## ADVANCE;

- Calculate the percentage contribution of each pizza type to total revenue.
- Analyze the cumulative revenue generated over time.
- Determine the top 3 most ordered pizza types based on revenue for each pizza category.

# RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

```sql
SELECT
    COUNT(order_id) AS total_orders
FROM
    orders;
```

Result Grid

| | total_orders |
|---|---|
| ▶ | 21350 |

# CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

```sql
SELECT
    ROUND(SUM(quantity * price), 2) AS total_revenue
FROM
    pizzas AS p
        INNER JOIN
    order_details od ON p.pizza_id = od.pizza_id;
```

Result Grid

| | total_revenue |
|---|---|
| ▶ | 817860.05 |

# IDENTIFY THE HIGHEST-PRICED PIZZA.

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

Result Grid | Filter Rows:

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

# IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

```sql
SELECT
    p.size,
    COUNT(od.order_details_id) AS most_common_pizza_size_ordered
FROM
    pizzas p
        INNER JOIN
    order_details od ON od.pizza_id = p.pizza_id
GROUP BY p.size
ORDER BY most_common_pizza_size_ordered DESC
LIMIT 1;
```

| size | most_common_pizza_size_ordered |
|---|---|
| L | 18526 |

# LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.

```sql
SELECT
    pt.name, SUM(od.quantity) AS quantity
FROM
    pizza_types pt
        INNER JOIN
    pizzas p ON p.pizza_type_id = pt.pizza_type_id
        INNER JOIN
    order_details od ON od.pizza_id = p.pizza_id
GROUP BY pt.name
ORDER BY quantity DESC
LIMIT 5;
```

Result Grid | Filter Rows:

| 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.

```sql
SELECT
    pt.category, SUM(od.quantity) AS quantity
FROM
    pizza_types pt
        INNER JOIN
    pizzas p ON p.pizza_type_id = pt.pizza_type_id
        INNER JOIN
    order_details od ON od.pizza_id = p.pizza_id
GROUP BY pt.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.

```
SELECT
    HOUR(order_time), COUNT(order_id) AS no_of_orders
FROM
    orders
GROUP BY HOUR(order_time)
ORDER BY no_of_orders DESC;
```

| | HOUR(order_time) | no_of_orders |
|---|---|---|
| ▶ | 12 | 2520 |
| | 13 | 2455 |
| | 18 | 2399 |
| | 17 | 2336 |
| | 19 | 2009 |
| | 16 | 1920 |
| | 20 | 1642 |
| | 14 | 1472 |
| | 15 | 1468 |
| | 11 | 1231 |
| | 21 | 1198 |
| | 22 | 663 |
| | 23 | 28 |
| | 10 | 8 |
| | 9 | 1 |

# JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

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

| | category | COUNT(pizza_type_id) |
|---|---|---|
| ▶ | Chicken | 6 |
| | Classic | 8 |
| | Supreme | 9 |
| | Veggie | 9 |

# GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

```sql
SELECT
    ROUND(AVG(quantity), 0) AS avg_pizza_per_day
FROM
    (SELECT
        o.order_date, SUM(od.quantity) AS quantity
    FROM
        orders o
    INNER JOIN order_details od ON o.order_id = od.order_id
    GROUP BY o.order_date) AS order_quantity;
```

Result Grid

| | avg_pizza_per_day |
|---|---|
| ▶ | 138 |

# DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```
SELECT
    pt.name, SUM(od.quantity * p.price) AS price
FROM
    pizza_types pt
        INNER JOIN
    pizzas p ON p.pizza_type_id = pt.pizza_type_id
        INNER JOIN
    order_details od ON p.pizza_id = od.pizza_id
GROUP BY pt.name
ORDER BY price DESC
LIMIT 3
;
```

| name | price |
|---|---|
| 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.

```
SELECT
    pt.category,
    ROUND(SUM(od.quantity * p.price) / (SELECT
                    ROUND(SUM(quantity * price), 2) AS total_revenue
                FROM
                    pizzas AS p
                        INNER JOIN
                    order_details od ON p.pizza_id = od.pizza_id) * 100,
            2) AS revenue_percentage
FROM
    pizza_types pt
        INNER JOIN
    pizzas p ON p.pizza_type_id = pt.pizza_type_id
        INNER JOIN
    order_details od ON p.pizza_id = od.pizza_id
GROUP BY pt.category
ORDER BY revenue_percentage DESC
;
```

| | category | revenue_percentage |
|---|---|---|
| ▶ | Classic | 26.91 |
| | Supreme | 25.46 |
| | Chicken | 23.96 |
| | Veggie | 23.68 |

# ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```
select date , sum(revenue) over(order by date) as cum_revenue
from
(select o.order_date as date,round(sum(p.price*od.quantity),2) as revenue from orders o
inner join order_details od
on od.order_id=o.order_id
inner join pizzas p
on p.pizza_id=od.pizza_id
group by date) as sales;
```

| date | cum_revenue |
|---|---|
| 2015-01-01 | 2713.85 |
| 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 | 21526.399999999998 |
| 2015-01-10 | 23990.35 |
| 2015-01-11 | 25862.649999999998 |

# DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.

```
select name,revenue,rn from
(select category,name,revenue,
rank() over (partition by category order by revenue desc) as rn
from
(select pt.category,pt.name,sum(od.quantity*p.price) as revenue from pizza_types pt
inner join pizzas p
on p.pizza_type_id=pt.pizza_type_id
inner join order_details od
on od.pizza_id=p.pizza_id
group by pt.category,pt.name) as a) as ranked_pizza
where rn<=3;
```

| name | revenue | rn |
|---|---|---|
| The Thai Chicken Pizza | 43434.25 | 1 |
| The Barbecue Chicken Pizza | 42768 | 2 |
| The California Chicken Pizza | 41409.5 | 3 |
| The Classic Deluxe Pizza | 38180.5 | 1 |
| The Hawaiian Pizza | 32273.25 | 2 |
| The Pepperoni Pizza | 30161.75 | 3 |
| The Spicy Italian Pizza | 34831.25 | 1 |
| The Italian Supreme Pizza | 33476.75 | 2 |
| The Sicilian Pizza | 30940.5 | 3 |
| The Four Cheese Pizza | 32265.70000000065 | 1 |
| The Mexicana Pizza | 26780.75 | 2 |
| The Five Cheese Pizza | 26066.5 | 3 |