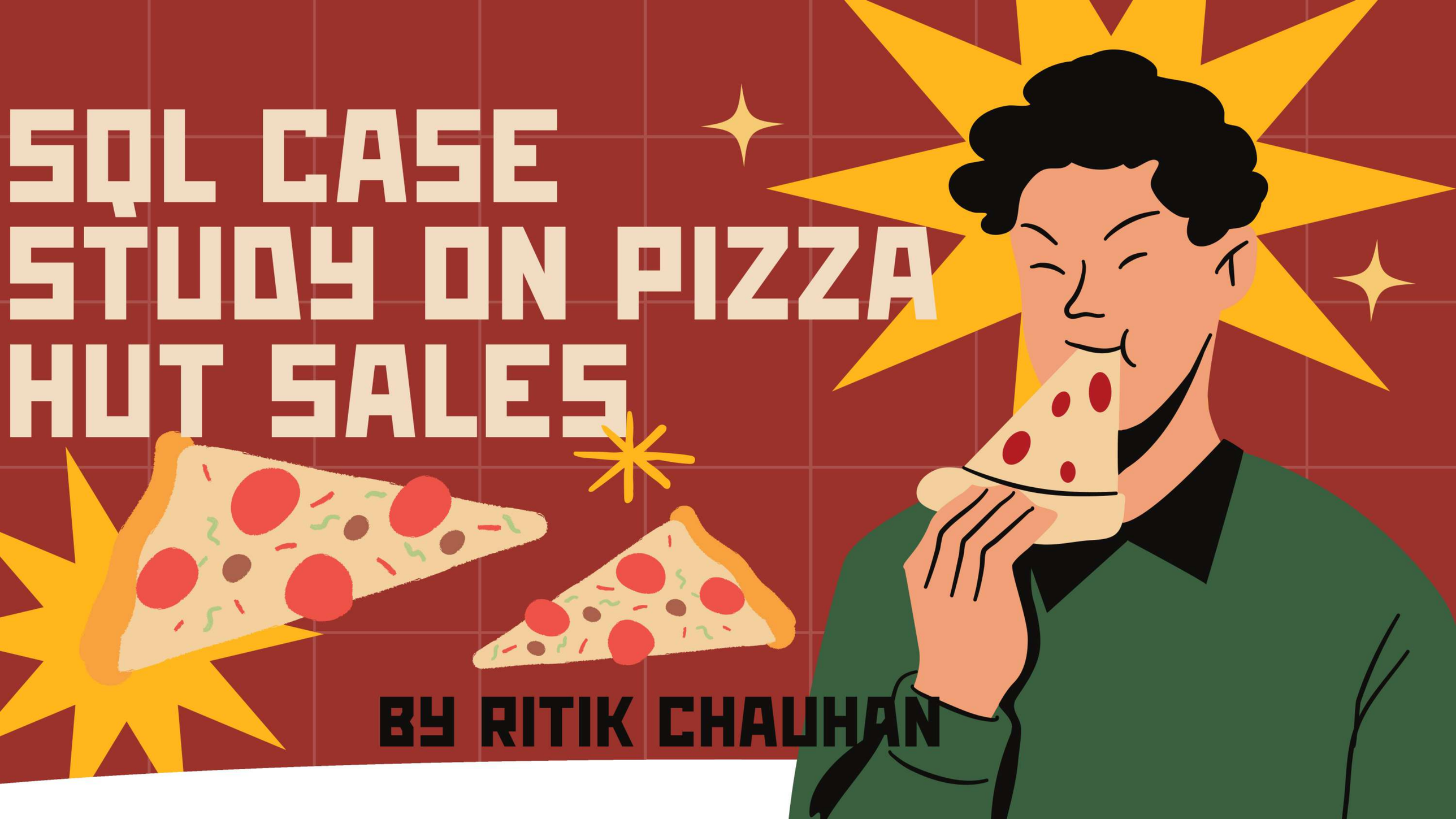
# SQL CASE STUDY ON PIZZA HUT SALES



BY RITIK CHAUHAN

# HELLO !

My name is Ritik Chauhan, and in this analytical study, I have employed SQL queries to conduct a comprehensive examination of Pizza Hut's sales performance.

The primary objective of this analysis is to derive actionable insights from the sales data, with the goal of supporting strategic business decisions.

# DATEBASE AND DATASET

This analysis is based on a dataset obtained from **WsCubeTech.com**, which encompasses detailed sales data for Pizza Hut. The dataset is segmented into three primary components:

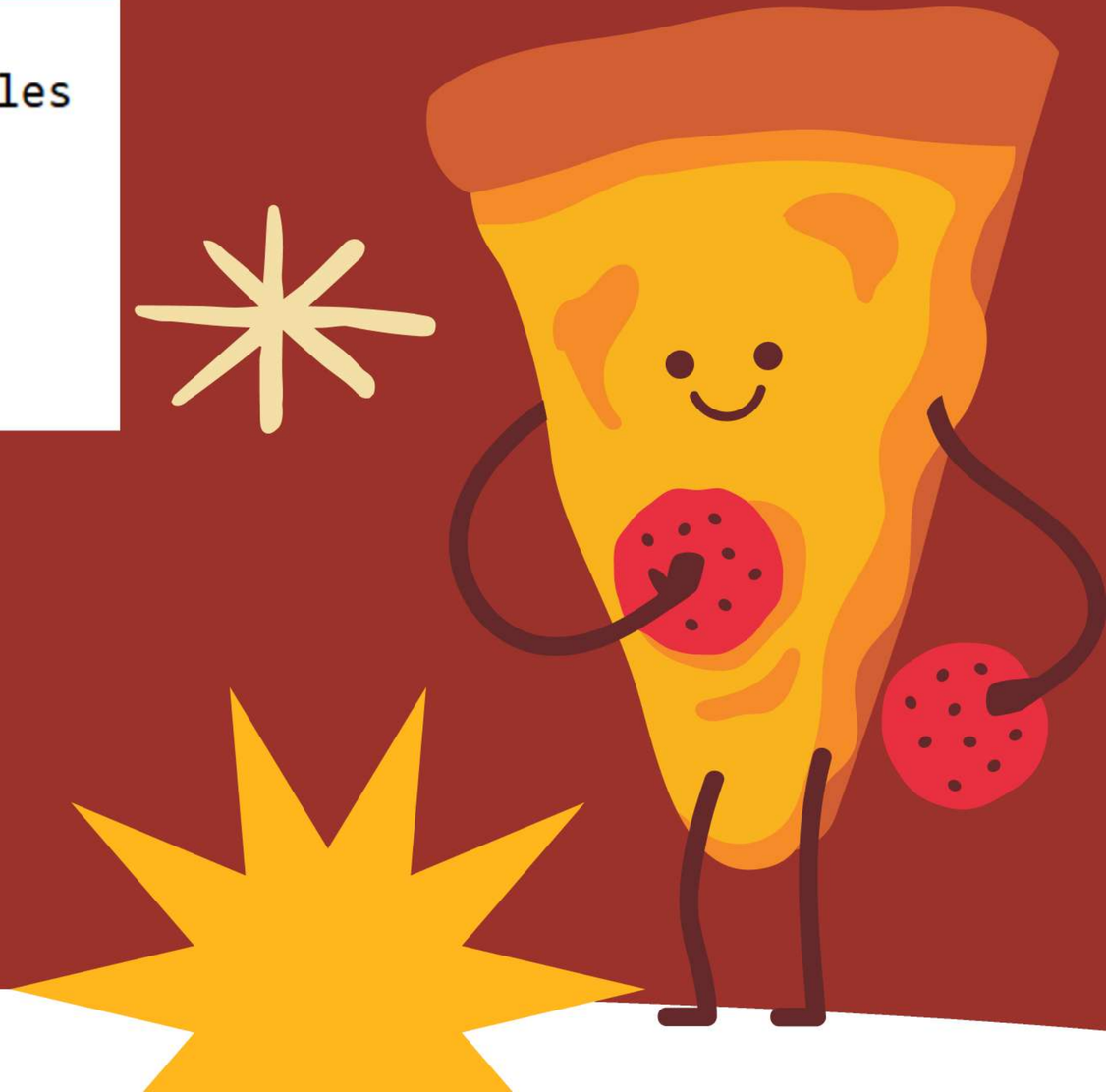
1. Orders Dataset: Contains records of over 21,000 orders placed at Pizza Hut. Each entry is uniquely identified by an order_id, and includes the date and time when the order was placed. This dataset provides the foundational timeline for analyzing sales trends and patterns.
2. Order Details Dataset: Comprises over 48,000 entries, detailing the specific pizzas ordered. Each record is linked to an order_id and includes the type of pizza ordered along with the quantity. This dataset allows for a granular analysis of customer preferences and order volumes.
3. Pizza Types Dataset: Contains information on 32 different pizza types available at Pizza Hut. Each pizza is categorized by its unique pizza_type_id and includes details such as the name, category (e.g., Chicken, Vegetarian), and ingredients. This dataset is crucial for understanding product diversity and categorization.

# 1. CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

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

Result Grid

| | Total_sales |
|---|---|
| ▶ | 817860.05 |

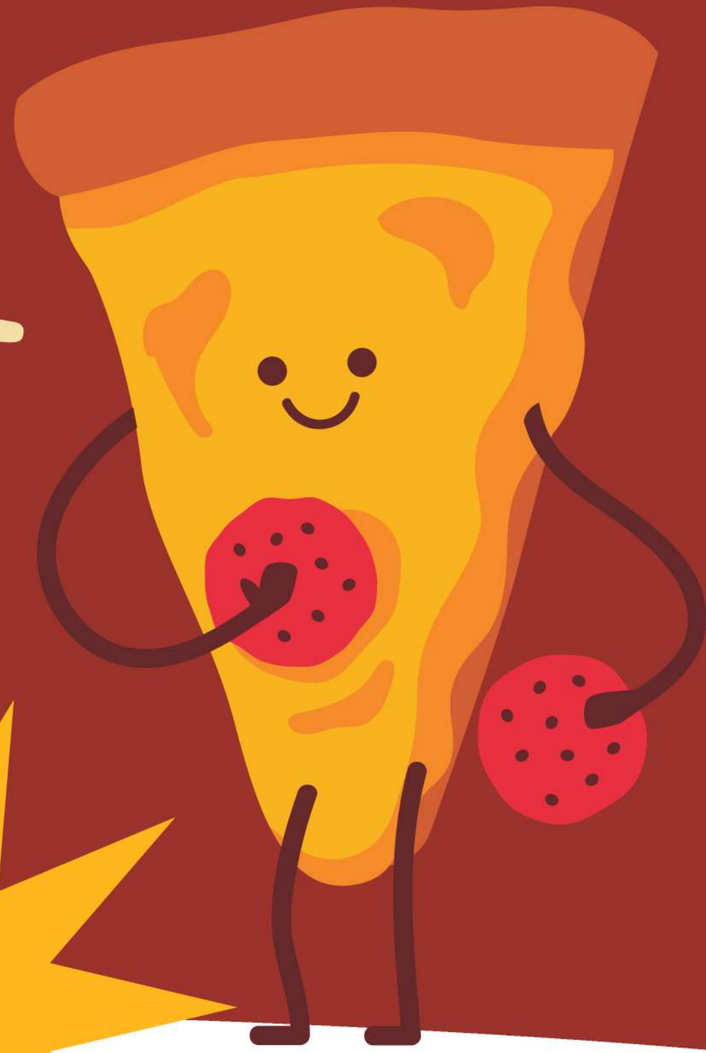


# 2.RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

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

Result Grid | Filter Rows:

| COUNT(order_id) |
|---|
| 21350 |

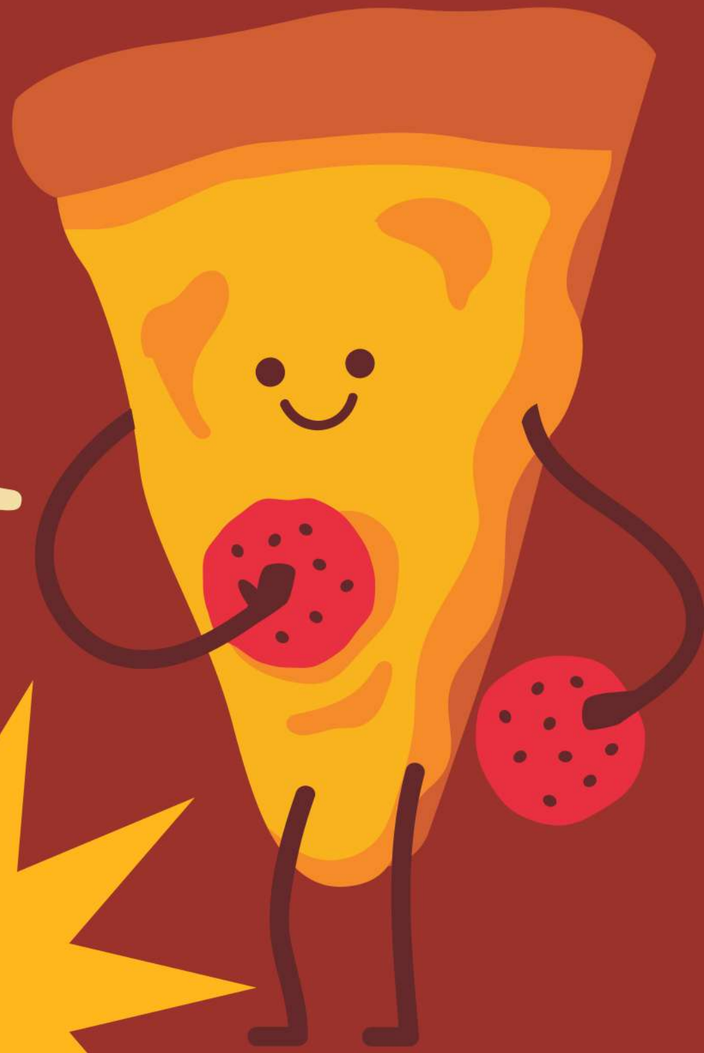


# 3.IDENTIFY THE HIGHEST-PRICED PIZZA.

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

Result Grid | Filter Rows:

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



# 4.IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

```sql
SELECT
    p.size as SIZE , COUNT(p.size) AS TOTAL_ORDER
FROM
    order_details AS od
        JOIN
    pizzas AS p ON od.pizza_id = p.pizza_id
GROUP BY p.size
ORDER BY TOTAL_ORDER DESC;
```

Result Grid  Filter Rows:

| SIZE | TOTAL_ORDER |
|---|---|
| L | 18526 |
| M | 15385 |
| S | 14137 |
| XL | 544 |
| XXL | 28 |

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

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

Result Grid | Filter Rows: | Exp

| 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 |

# 6. 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 AS pt
        JOIN
    pizzas AS pp ON pt.pizza_type_id = pp.pizza_type_id
        JOIN
    order_details AS od ON od.pizza_id = pp.pizza_id
GROUP BY pt.category
ORDER BY quantity DESC;
```

Result Grid | Filter Rows:

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

# 7.DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

```sql
SELECT
    HOUR(time) AS hhour, COUNT(HOUR(time)) as ORDER_COUNT
FROM
    orders
GROUP BY hhour;
```

Result Grid | Filter Rows:

| hhour | 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 |

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

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

Result Grid  Filter Rows:

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

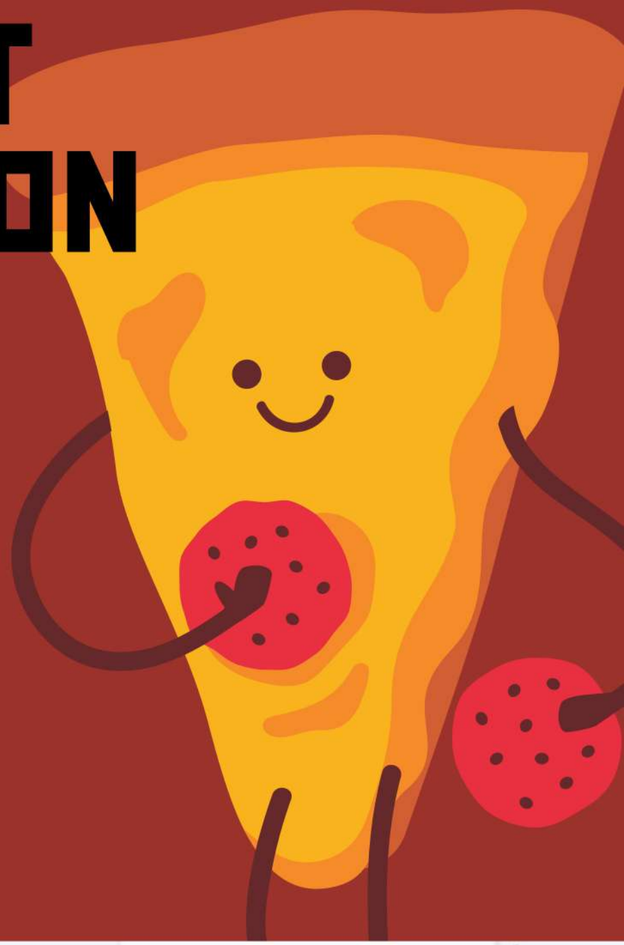
# 9. GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

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

Result Grid | Filter Rows:

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

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



```sql
SELECT
    pt.name, ROUND(SUM(od.quantity * p.price), 2) AS Total_sales
FROM
    order_details AS od
        JOIN
    pizzas AS p ON od.pizza_id = p.pizza_id
        JOIN
    pizza_types AS pt ON pt.pizza_type_id = p.pizza_type_id
GROUP BY pt.name
ORDER BY Total_sales DESC
LIMIT 3;
```

Result Grid | Filter Rows: | Expo

| name | Total_sale |
|---|---|
| The Thai Chicken Pizza | 43434.25 |
| The Barbecue Chicken Pizza | 42768 |
| The California Chicken Pizza | 41409.5 |

# 11. CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

```
SELECT
    pizza_types.category,
    ROUND((SUM(order_details.quantity * pizzas.price) / (SELECT
                    ROUND(SUM(order_details.quantity * pizzas.price),
                                2) AS total_sales
                FROM
                    order_details
                        JOIN
                    pizzas ON pizzas.pizza_id = order_details.pizza_id)) * 100,
            2) 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
ORDER BY revenue DESC;
```

Result Grid | Filter Row

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

# 12. ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```sql
select date,
sum(revenue) over(order by date) as cum_revenue
from (select orders.date,
sum(order_details.quantity * pizzas.price) as revenue
from order_details join pizzas
on pizzas.pizza_id = order_details.pizza_id
join orders
on orders.order_id = order_details.order_id
group by orders.date) as sales;
```

Result Grid | Filter Rows:

| 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 |
| 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 |

# 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 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;
```

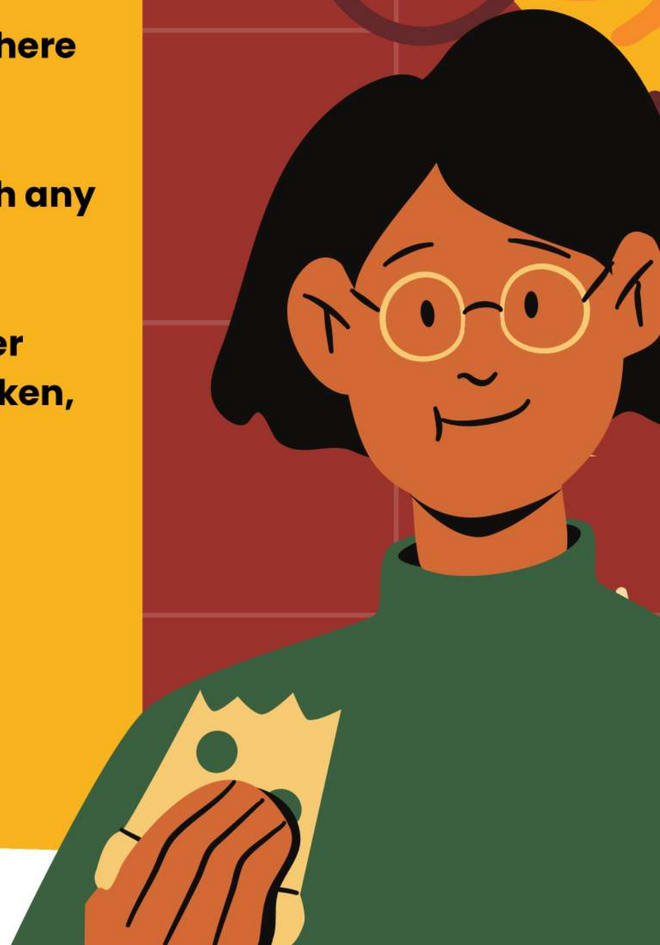
Result Grid | Filter Rows: | Export: | Wrap C

| 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 | 32265.70000000065 |
| The Mexicana Pizza | 26780.75 |
| The Five Cheese Pizza | 26066.5 |

# CONCLUSION

To conclude the analysis, focus on summarizing the key findings of the sales data. For instance:

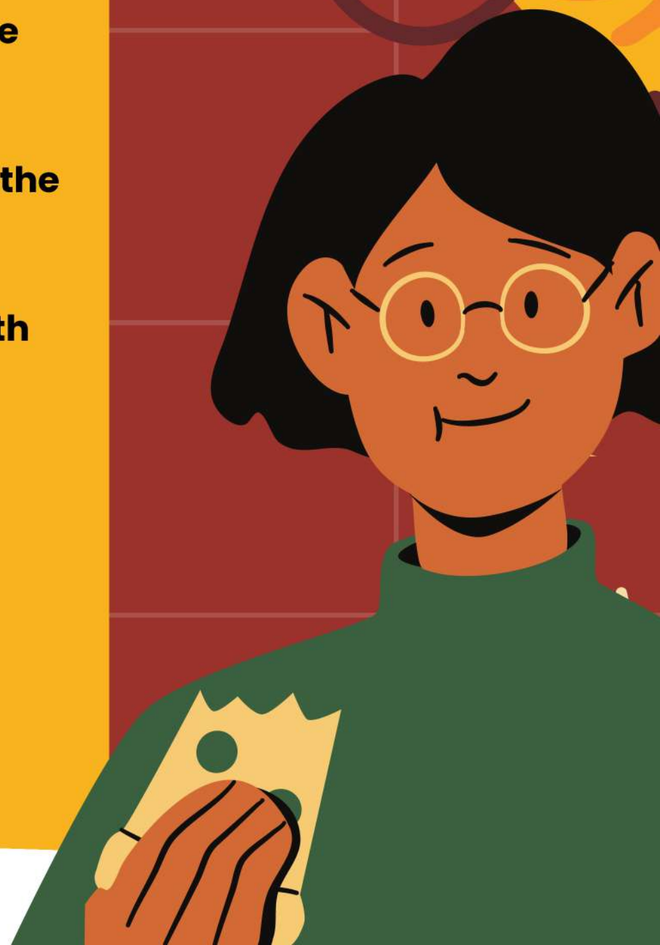
- Sales Trends: Highlight the overall trends in sales over time. Were there periods of significant growth or decline?
- Popular Pizzas: Identify the most and least popular pizzas, along with any seasonal variations.
- Customer Preferences: Discuss any notable patterns in customer preferences, such as a preference for certain categories (e.g., Chicken, Vegetarian).

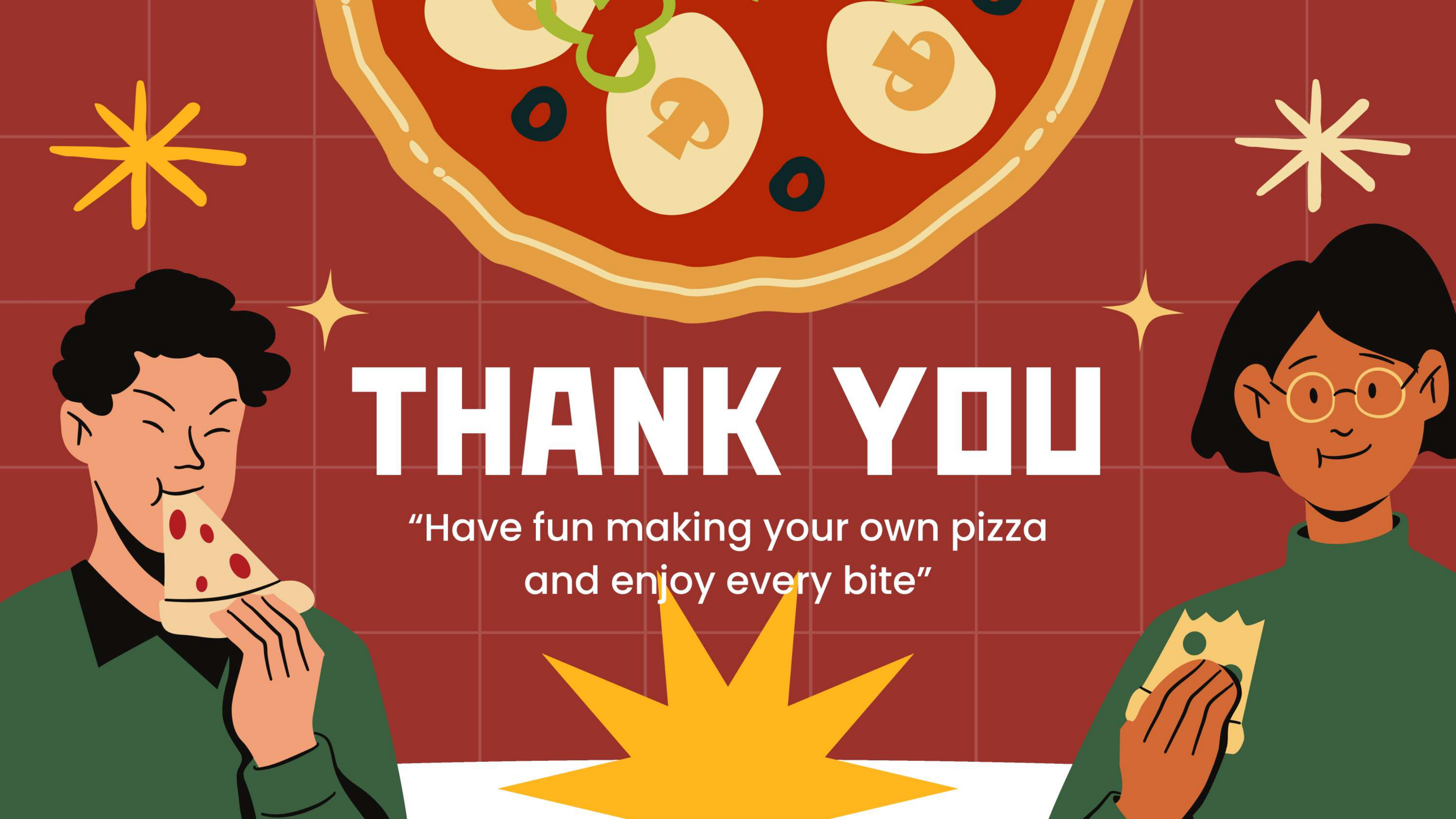


# IMPLICATIONS

**Based on the findings, outline the strategic implications:**

- **Marketing Focus: Suggest which products should be promoted more aggressively based on their popularity or profitability.**
- **Inventory Management: Recommend changes to inventory based on the demand for certain pizzas.**
- **Customer Engagement: Propose initiatives to engage customers with targeted promotions or new product offerings.**




THANK YOU
“Have fun making your own pizza
and enjoy every bite”