

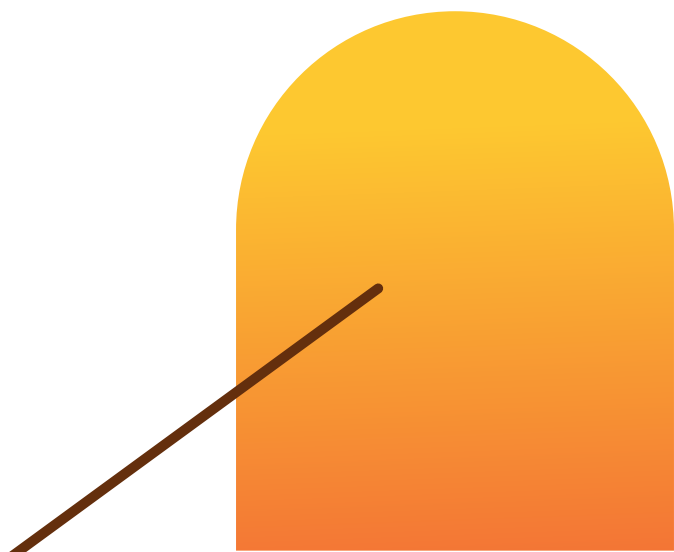
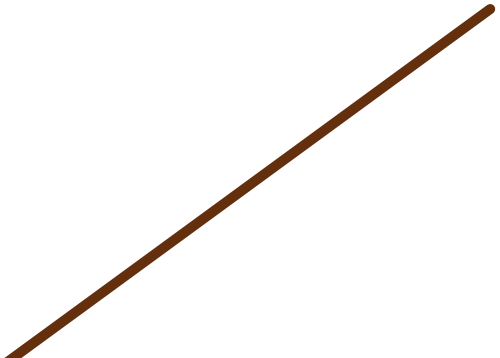

The background features a collection of abstract geometric shapes. Large, overlapping circles in shades of orange and yellow are positioned in the corners. Thin, dark brown lines intersect the scene at various angles. Numerous small, solid-colored circles are scattered throughout the white space.

Pizza Sales Report



Overview

This report presents a comprehensive analysis of pizza sales performance across multiple key metrics. Leveraging datasets encompassing various pizza-related information, the analysis delves into critical indicators such as total sales revenue, order count, average order value, and revenue contribution from different pizza categories. The report reveals valuable insights into the pizza sales landscape, shedding light on prevailing trends and patterns. One of the notable findings highlights the significant revenue generation attributed to non-vegetarian pizza offerings. This specific category emerged as the top contributor to overall sales revenue, suggesting a strong consumer preference for meat-based pizza varieties.



Methodology

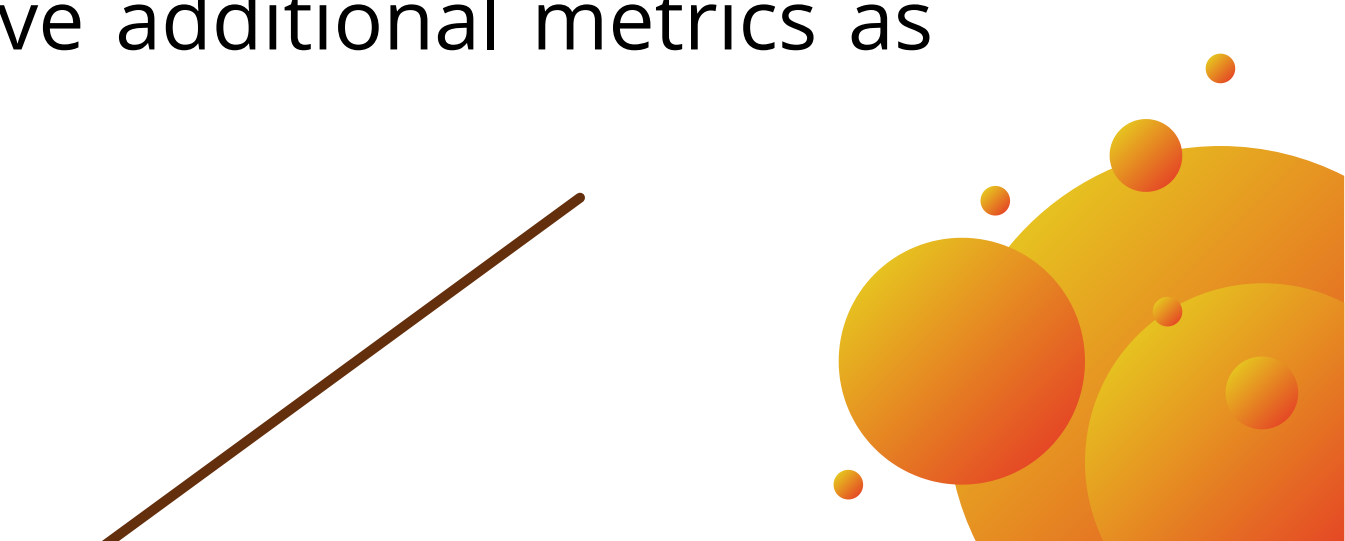
The datasets included information on sales transactions, product offerings, and customer demographics.

Data Preparation:

- The raw data was initially processed to ensure data quality and consistency. This involved identifying and handling any missing or duplicate records, as well as standardizing data formats.
- Relevant tables were joined using appropriate keys to establish relationships between different data entities, enabling comprehensive analysis.



Data Analysis:

- MySQL was employed as the primary tool for querying and analyzing the data. SQL queries were constructed to calculate key performance indicators (KPIs) such as total sales revenue, order count, average order value, and revenue contribution by pizza category.
 - Aggregate functions (SUM, COUNT, AVG) were used to summarize data at various levels, facilitating analysis across different dimensions (e.g., time periods, product categories)
 - Advanced SQL functions, such as window functions and subqueries, were utilized to perform complex calculations and derive additional metrics as needed.
- 

1. Retrieve the total number of orders placed

```
select * from orders;  
select count(order_id) as total_order from orders;
```

Result Grid	
	total_order
▶	21350

2. Calculate the total revenue generated from pizza sales.

- ```
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
```

| Result Grid |             |
|-------------|-------------|
|             | Total_sales |
| ▶           | 817860.05   |

### 3. Identify the highest-priced pizza.

```
4 • SELECT
5 pizza_types.name, pizzas.price
6 FROM
7 pizza_types
8 JOIN
9 pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
10 ORDER BY pizzas.price DESC
11 LIMIT 1;
```

| Result Grid |                 |       | Filter Rows |
|-------------|-----------------|-------|-------------|
|             | name            | price |             |
| ▶           | The Greek Pizza | 35.95 |             |

## 4. Identify the most common pizza size ordered.



```
4 • SELECT
5 pizzas.size,
6 COUNT(order_details.order_details_id) AS order_count
7 FROM
8 pizzas
9 JOIN
10 order_details ON pizzas.pizza_id = order_details.pizza_id
11 GROUP BY pizzas.size
12 ORDER BY order_count DESC;
```

| Result Grid |      |             | Filter |
|-------------|------|-------------|--------|
|             | size | order_count |        |
| ▶           | L    | 18526       |        |
|             | M    | 15385       |        |
|             | S    | 14137       |        |
|             | XL   | 544         |        |
|             | XXL  | 28          |        |



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

```
SELECT
 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 order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY quantity DESC
LIMIT 5;
```

| Result Grid     Filter Rows: <input type="text"/> |                            |          |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|----------|
|                                                                                                                                                                                                                         | 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.

```
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 order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity DESC
LIMIT 5;
```

| Result Grid |          |          | Filter |
|-------------|----------|----------|--------|
|             | category | quantity |        |
| ▶           | Classic  | 14888    |        |
|             | Supreme  | 11987    |        |
|             | Veggie   | 11649    |        |
|             | Chicken  | 11050    |        |



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

```
SELECT
 HOUR(order_time) AS hour, COUNT(order_id) AS order_count
FROM
 orders
group by hour(order_time);
```

| Result Grid |      |             | Filter |
|-------------|------|-------------|--------|
|             | hour | order_count |        |
| ▶           | 11   | 1231        |        |
|             | 12   | 2520        |        |
|             | 13   | 2455        |        |
|             | 14   | 1472        |        |
|             | 15   | 1468        |        |

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

```
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.

```
SELECT
 ROUND(AVG(quantity), 0) as avg_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;
```

| Result Grid |                           | Filter Rows: |
|-------------|---------------------------|--------------|
|             | avg_pizza_ordered_per_day |              |
| ▶           | 138                       |              |

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

```
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;
```

| Result Grid |                              |          | Filter Rows: |
|-------------|------------------------------|----------|--------------|
|             | name                         | revenue  |              |
| ▶           | 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 R |
|-------------|----------|---------|----------|
|             | category | revenue |          |
| ▶           | Classic  | 26.91   |          |
|             | Supreme  | 25.46   |          |
|             | Chicken  | 23.96   |          |
|             | Veggie   | 23.68   |          |

# 12 Analyze the cumulative revenue generated over time.



```
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;
```

| Result Grid     Filter Rows: <input type="text"/> |            |                    |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|--------------------|
|                                                                                                                                                                                                                         | 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           |
|                                                                                                                                                                                                                         | 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  |



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

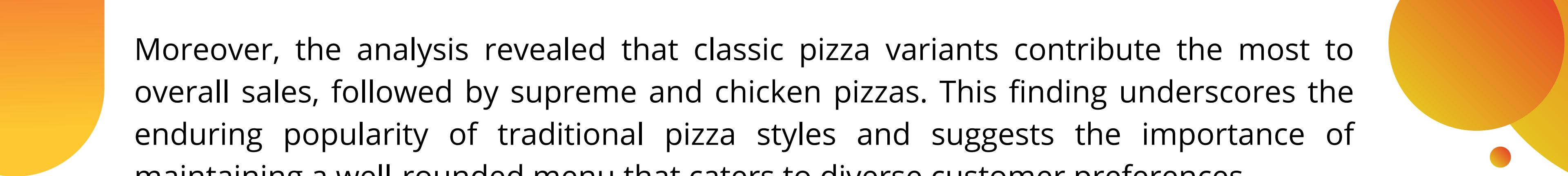
```
select name, revenue from
(select category, name, revenue,
rank() over(partition by category order by revenue desc) as rn
from
(select pizza_types.category, pizza_types.name,
sum((order_details.quantity) * (pizzas.price)) as revenue
from pizza_types join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
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: <input type="text"/> |                              |          |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------|----------|
|                                                                                                                                                                                                                             | name                         | revenue  |
| ▶                                                                                                                                                                                                                           | The Thai Chicken Pizza       | 43434.25 |
|                                                                                                                                                                                                                             | The Barbecue Chicken Pizza   | 42768    |
|                                                                                                                                                                                                                             | The California Chicken Pizza | 41409.5  |

# Conclusion :-

The comprehensive analysis of pizza sales data has yielded valuable insights that can inform strategic decision-making and drive revenue growth for the business. Among the key findings, it is evident that non-vegetarian pizza offerings are significantly preferred by customers, generating the highest revenue contributions.

Specifically, three non-vegetarian pizza variants – Thai Chicken Pizza, Barbecue Pizza, and California Pizza – emerged as the top revenue generators, highlighting their popularity and customer appeal. This insight presents an opportunity to further optimize and promote these successful menu items, potentially through targeted marketing campaigns or special promotions.



Moreover, the analysis revealed that classic pizza variants contribute the most to overall sales, followed by supreme and chicken pizzas. This finding underscores the enduring popularity of traditional pizza styles and suggests the importance of maintaining a well-rounded menu that caters to diverse customer preferences.

Quantitative metrics offer additional perspectives on customer behavior and operational performance. The data indicates that an average of 138 pizzas are ordered per day, with large-sized pizzas being the most popular choice, followed by medium and small sizes. This information can guide inventory management, staffing decisions, and resource allocation to ensure efficient operations and meet customer demand effectively

By leveraging the insights gained from this analysis, the business can make informed decisions to enhance product offerings, refine marketing strategies, and optimize operational processes. Embracing a data-driven approach will be crucial in staying competitive within the dynamic pizza industry and delivering superior customer experiences.

