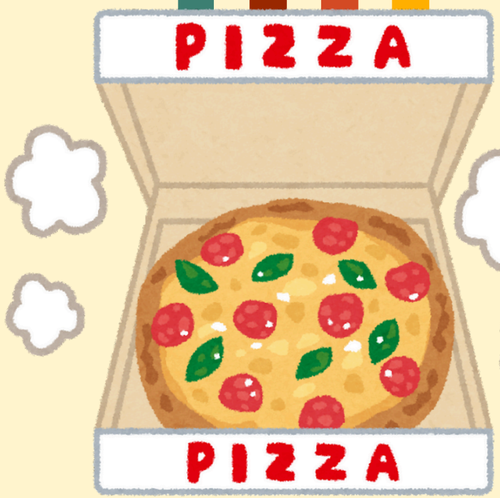
DATA ANALYSIS

PizzaHut

SALES ANALYSIS PROJECT

Using MySQL



10 August, 2024

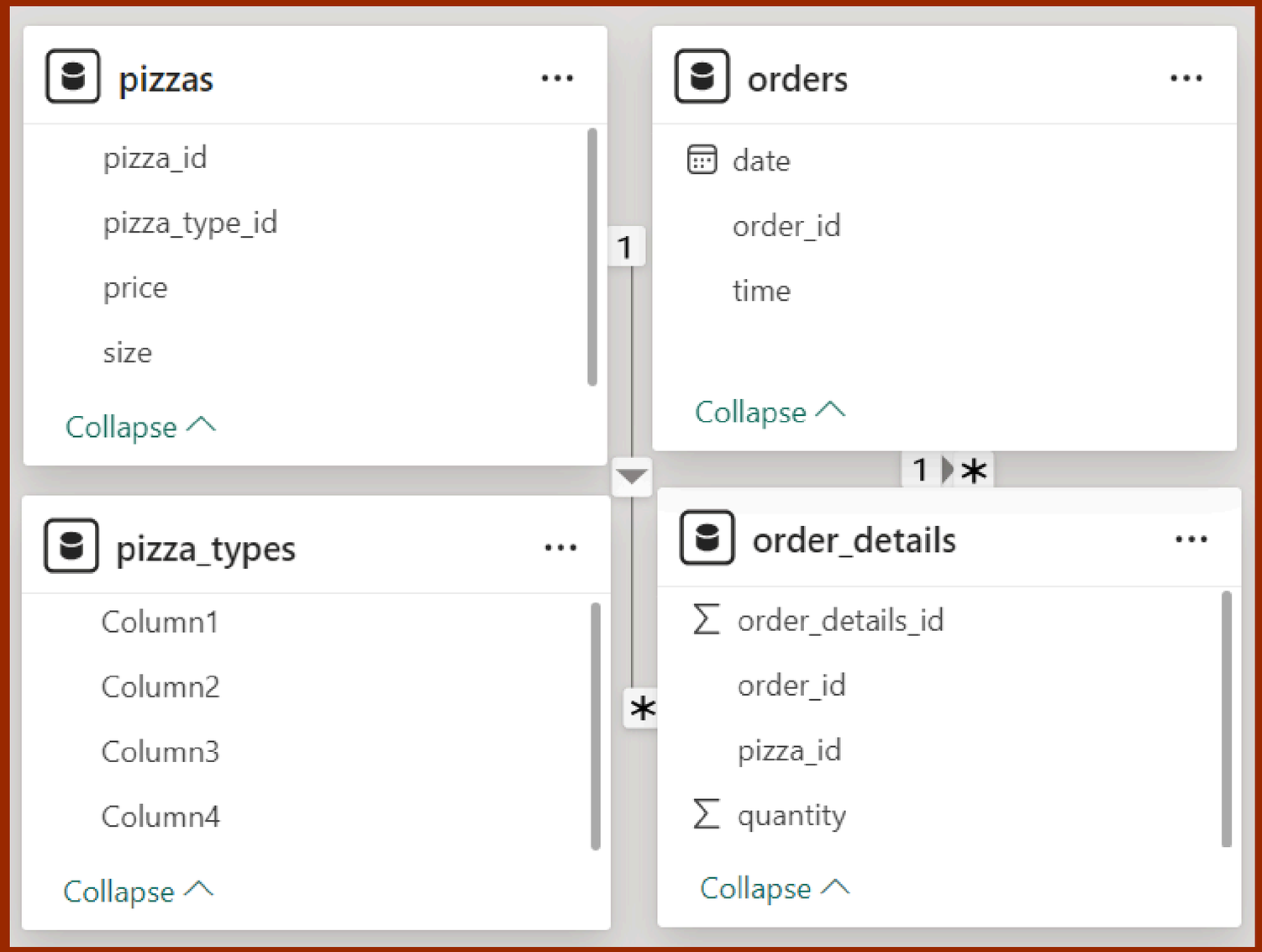
IMRAN MANSHA

Introduction

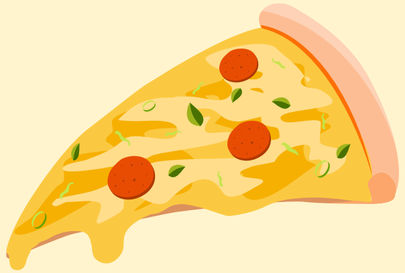
This project focuses on analyzing pizza sales data using MySQL to uncover valuable insights and drive data-driven decisions. By examining various aspects of pizza sales, including revenue, order patterns, and pizza popularity, the project aims to provide actionable information for optimizing business strategies. Key analyses include identifying top-performing pizza types, understanding category-wise revenue contributions, and visualizing cumulative sales trends over time. The results offer a comprehensive view of sales dynamics and support strategic decision-making in the pizza industry.

Imran Mansha

Data Modelling



Queries



1. Retrieve the total number of orders placed.

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

total_orders
21350

2. Calculate the total revenue generated from pizza sales.

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

	total_sales
▶	817860.05

3. Identify the highest-priced pizza.

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

name	price
The Greek Pizza	35.95

4. Identify the most common pizza size ordered.

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

size	order_count
L	18526

5. Identify the most common pizza size ordered.

```sql
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;
```

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 total_quantity
FROM pizzas p
JOIN order_details od ON p.pizza_id = od.pizza_id
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.category
ORDER BY total_quantity DESC;
```

category	total_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

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

```sql
SELECT EXTRACT(HOUR FROM order_time) AS hour_of_day, COUNT(*) AS total_orders
FROM orders
GROUP BY hour_of_day
ORDER BY hour_of_day;
```

hour_of_day	total_orders
9	1
10	8
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

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

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

category	total_pizzas
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 ROUND(AVG(daily_total),0 ) AS avg_pizzas_per_day
FROM (
    SELECT o.order_date, SUM(od.quantity) AS daily_total
    FROM orders o
    JOIN order_details od ON o.order_id = od.order_id
    GROUP BY o.order_date
) AS daily_totals;
```

avg_pizzas_per_day
138

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

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

name	total_revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

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

```sql
WITH TotalRevenue AS (
    SELECT SUM(od.quantity * p.price) AS total_revenue
    FROM pizzas p
    JOIN order_details od ON p.pizza_id = od.pizza_id
),
CategoryRevenue AS (
    SELECT pt.category, SUM(od.quantity * p.price) AS category_revenue
    FROM pizzas p
    JOIN order_details od ON p.pizza_id = od.pizza_id
    JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
    GROUP BY pt.category
)
SELECT cr.category,
       ROUND((cr.category_revenue / (SELECT total_revenue FROM TotalRevenue) * 100), 2) AS percentage_contribution
FROM CategoryRevenue cr
ORDER BY percentage_contribution DESC;
```

category	percentage_contribution
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

12. Analyze the cumulative revenue generated over time.

```sql
WITH DailyRevenue AS (
    SELECT o.order_date,
           SUM(od.quantity * p.price) AS daily_revenue
    FROM orders o
    JOIN order_details od ON o.order_id = od.order_id
    JOIN pizzas p ON od.pizza_id = p.pizza_id
    GROUP BY o.order_date
),
CumulativeRevenue AS (
    SELECT order_date,
           daily_revenue,
           SUM(daily_revenue) OVER (ORDER BY order_date) AS cumulative_revenue
    FROM DailyRevenue
)
SELECT order_date,
       round((cumulative_revenue),2)
FROM CumulativeRevenue
ORDER BY order_date;
```

order_date	round((cumulative_revenue),2)
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	21526.4
2015-01-10	23990.35

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

1

```sql
WITH PizzaRevenue AS (
    SELECT pt.category,
           pt.name,
           SUM(od.quantity * p.price) AS pizza_revenue
    FROM pizzas p
    JOIN order_details od ON p.pizza_id = od.pizza_id
    JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
    GROUP BY pt.category, pt.name
),
RankedPizzaRevenue AS (
    SELECT category,
           name,
           pizza_revenue,
           ROW_NUMBER() OVER (PARTITION BY category ORDER BY pizza_revenue DESC) AS rnk
    FROM PizzaRevenue
)
```

2

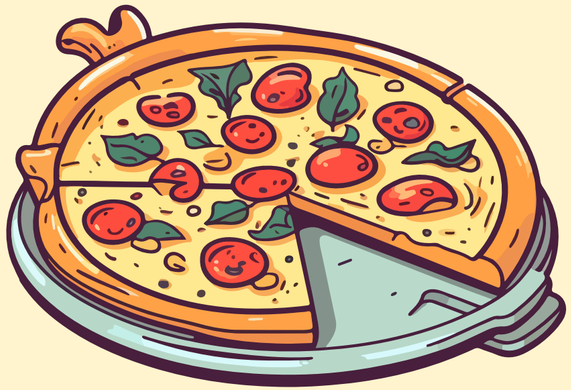
```sql
SELECT name,
       pizza_revenue
FROM RankedPizzaRevenue
WHERE rnk <= 3
ORDER BY MIN(category) OVER (ORDER BY rnk), pizza_revenue DESC;

WITH PizzaRevenue AS (
    SELECT pt.category,
           pt.name,
           SUM(od.quantity * p.price) AS pizza_revenue
    FROM pizzas p
    JOIN order_details od ON p.pizza_id = od.pizza_id
    JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
    GROUP BY pt.category, pt.name
),
```

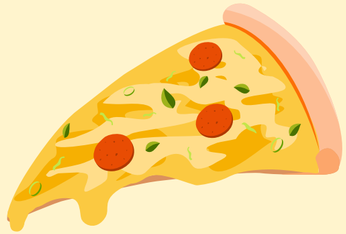
3

```sql
RankedPizzaRevenue AS (
    SELECT category,
           name,
           pizza_revenue,
           ROW_NUMBER() OVER (PARTITION BY category ORDER BY pizza_revenue DESC) AS rnk
    FROM PizzaRevenue
)
SELECT category,
       name,
       pizza_revenue
FROM RankedPizzaRevenue
WHERE rnk <= 3
ORDER BY category, pizza_revenue DESC;
```

category	name	pizza_revenue
Chicken	The Thai Chicken Pizza	43434.25
Chicken	The Barbecue Chicken Pizza	42768
Chicken	The California Chicken Pizza	41409.5
Classic	The Classic Deluxe Pizza	38180.5
Classic	The Hawaiian Pizza	32273.25
Classic	The Pepperoni Pizza	30161.75
Supreme	The Spicy Italian Pizza	34831.25
Supreme	The Italian Supreme Pizza	33476.75
Supreme	The Sicilian Pizza	30940.5
Veggie	The Four Cheese Pizza	32265.70000000065



Objectives



Project Objectives

Identify High-Performing Pizza Types:

- Determine which pizza types generate the most revenue and are most frequently ordered to inform menu and marketing strategies.

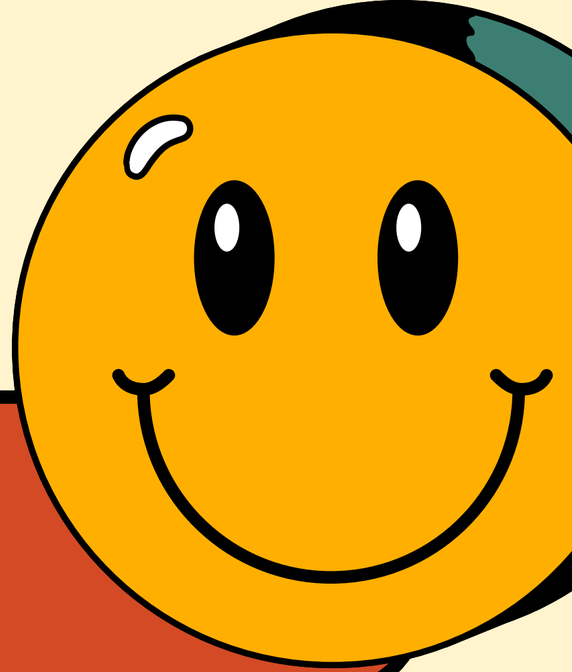
Analyze Revenue Distribution by Category:

- Examine how different pizza categories contribute to total revenue to identify which categories are the most profitable.

Visualize Sales Trends Over Time:

- Track and analyze cumulative revenue trends over time to understand sales patterns and seasonal variations.

Expected Outcomes



Top-Performing Pizza Insights:

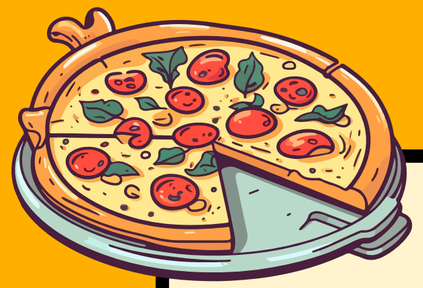
- A ranked list of the top 3 most ordered pizza types based on revenue, providing insights into consumer preferences and successful menu items.

Revenue Contribution Breakdown:

- A clear understanding of the percentage contribution of each pizza category to total revenue, highlighting the most and least profitable categories.

Sales Trend Visualization:

- Visual representations of cumulative revenue over time, revealing trends, peaks, and troughs in sales, which can guide inventory and promotional strategies.



Conclusion

The project effectively leveraged MySQL to analyze pizza sales data, revealing key insights into top-performing pizza types and revenue distribution by category. It provided a clear view of cumulative sales trends over time, facilitating informed decision-making.

Imran Mansha

Thank You