PIZZA SALES ANALYSIS

Using MySQL

By: SWASTIK SAMADDER

CONTENT

INTRODUCTION

This project leveraged MySQL's analytical capabilities and strategic SQL operations (including joins), combined with critical thinking, to extract insights for solving business problems within a pizza dataset. The core analysis centered on understanding sales performance in relation to category, revenue generated, customer choices and preferences, and changes over time.

KEY FEATURES:

1. TIME BASED ORDER ANALYSIS.
2. ORDERS MANAGEMENT.
3. REVENUE ANALYSIS.
4. MOST POPULAR PIZZAS BASED ON PRICE, ORDER, CATEGORY.

PROJECT GOALS

- **Understand Customer Behaviour:** Identify popular pizza types, preferred sizes, and peak ordering times to better cater to customer preferences.
- **Optimize Sales Strategies:** Provide data-driven recommendations to enhance sales during off-peak hours and boost the sales of popular and high-revenue products.
- **Improve Operational Efficiency:** Gain insights into order patterns to optimize staffing, inventory management, and production planning.
- **Drive Revenue Growth:** Identify opportunities to increase overall revenue through targeted promotions, product portfolio adjustments, and enhanced customer satisfaction.

DATA DESCRIPTION

This document outlines the schema for the tables used in a pizza ordering system database. The database includes 4 csv format datasheets- order_details, orders, pizzas. Pizza_types. The data description is given below-

1. order_details:

- **Purpose:** This table stores information about each pizza included within a customer's order.
- **Columns:**
  - order_details_id (INT, PRIMARY KEY): A unique identifier assigned to each individual item within an order. This serves as the primary key for this table.
  - order_id (INT, FOREIGN KEY): A foreign key that references the order_id in the orders table. This links each order detail back to the specific order it belongs to.
  - pizza_id (VARCHAR, FOREIGN KEY): A foreign key that references the pizza_id in the pizzas table. This identifies the specific type and size of pizza ordered.
  - quantity (INT): The number of units of the specified pizza_id included in this order detail.

2. orders

- Purpose: This table stores header-level information for each customer order placed. Each row represents a single, complete order.
- Columns:
  - order_id (INT, PRIMARY KEY): A unique identifier assigned to each customer order. This serves as the primary key for this table.
  - date (DATE): The date on which the order was placed.
  - time (TIME): The time at which the order was placed.

3. pizzas

- Purpose: This table contains specific details about the individual pizza products offered, including their size and price.
- Columns:
  - pizza_id (VARCHAR, PRIMARY KEY): A unique identifier for each specific pizza offering (considering both type and size). This serves as the primary key for this table.
  - pizza_type_id (VARCHAR, FOREIGN KEY): A foreign key that references the pizza_type_id in the pizza_types table. This links each pizza to its general type (e.g., Margherita).
  - size (VARCHAR): The size of the pizza (e.g., 'Small', 'Medium', 'Large', 'XL', 'XXL').
  - price (DECIMAL): The price of the pizza.

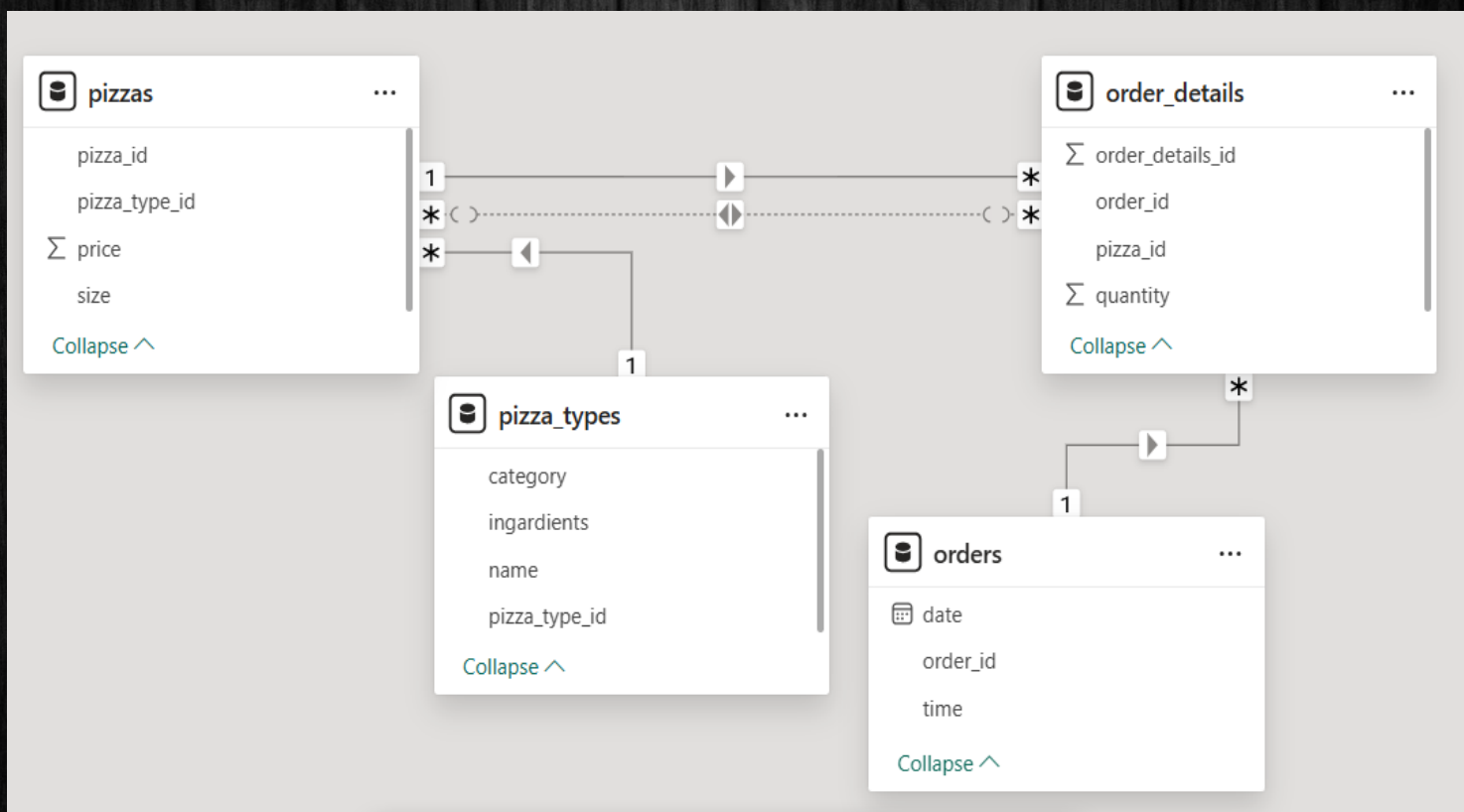
4. pizza_types

- Purpose: This table defines the different categories and names of pizza types available, along with their ingredients.
- Columns:
  - pizza_type_id (VARCHAR, PRIMARY KEY): A unique identifier for each general pizza type (e.g., 'MARG', 'PPR', 'SUP'). This serves as the primary key for this table.
  - name (VARCHAR): The descriptive name of the pizza type (e.g., 'Margherita Pizza', 'Pepperoni Pizza', 'Supreme Pizza').
  - category (VARCHAR): The broader category the pizza type belongs to (e.g., 'Veggie', 'Chicken', 'Supreme', 'Classic').
  - ingredients (VARCHAR): The ingredients included in this pizza type (e.g., 'cheese, tomatoes', 'cheese, pepperoni', 'cheese, pepperoni, olives, mushrooms').

This structured format clearly outlines the purpose of each table and describes the columns they contain, including their data types and any primary or foreign key constraints. This information is essential for understanding the database structure and formulating SQL queries for analysis.

DATA SCHEMA

A relational database with four tables managing pizza orders, items, and types.
Relationships: One order to many order details; one pizza type to many pizza sizes. Foreign keys ensure data links and integrity. Data schema is prepared in data model view of Power BI.



KEY PERFORMANCE INDICATORS

- **Total Revenue:** Overall sales generated within the analysis period.
- **Number of Orders:** Total count of pizza orders placed.
- **Average Order Value:** Revenue generated per order.
- **Most Ordered Pizza Types (by Quantity):** Identifying the most popular products based on the number of orders.
- **Top Revenue-Generating Pizza Types :** Identifying the products contributing the most to the total revenue.
- **Order Distribution by Hour:** Understanding when the highest order volumes occur.
- **Most Popular Pizza Size:** Determining the preferred size among customers.
- **Category-wise Revenue Contribution:** Assessing the performance of different pizza categories (e.g., Classic, Supreme, Chicken, Veggie).
- **Cumulative Revenue Over Time:** Tracking the trend of revenue growth

DATA ANALYSIS

1. Retrieve the total number of orders placed.

```
-- 1. Retrieve the total number of orders placed.
select count(order_id) as total_orders from orders;
```

RESULTS

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	total_orders
▶	21350

2. Calculate the total revenue generated from pizza sales

```sql
-- 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;
```

RESULTS

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_sales
817860.05

3. Identify the highest priced pizza

```sql
-- 3. Identify the highest-priced pizza.

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;
```

RESULTS

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

name	price
The Greek Pizza	35.95

4. Identify the most common pizza size ordered

```sql
-- 4.Identify the most common pizza size ordered.

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
LIMIT 1
;
```

RESULTS

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

size	order_count
L	18526

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

```sql
-- 5.List the top 5 most ordered pizza types along with their quantities.

SELECT
    pizza_types.name,
    SUM(order_details.quantity) AS order_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 order_quantity DESC
LIMIT 5;
```

RESULTS

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

name	order_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
-- 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;
```

RESULTS

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

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

```sql
-- 7.Determine the distribution of orders by hour of the day.

SELECT
    HOUR(order_time) AS hour, COUNT(order_id) AS oder_count
FROM
    orders
GROUP BY HOUR(order_time);
```

RESULTS

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

hour	oder_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

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

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

RESULTS

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category	count(name)
Supreme	9
Veggie	9
Classic	8
Chicken	6

9. Group the orders by date and calculate the average number of pizzas ordered per day

```sql
-- 9.Group the orders by date and calculate the average number of pizzas ordered per day
SELECT
    ROUND(AVG(quantity), 0) AS average_order_quantity
FROM
    (SELECT
        orders.order_dates, SUM(order_details.quantity) AS quantity
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY orders.order_dates) AS order_quantity;
```

RESULTS

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

average_order_quantity
138

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

```sql
-- 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;
```

RESULTS

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch 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

```sql
-- 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;
```

RESULTS

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

12. Analyze the cumulative revenue generated over time

```sql
-- 12.Analyze the cumulative revenue generated over time.

select order_dates,
sum(revenue) over(order by order_dates) as cumulative_revenue
from
(select orders.order_dates,
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 order_dates) as sales;
```

RESULTS

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_dates	cumulative_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

13. Top 3 most ordered pizza types based on revenue for each pizza category

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

select category,name,revenue
from
(select category, name, revenue,
rank() over(partition by category order by revenue desc) as ranking
from
(SELECT
    pizza_types.category,
    pizza_types.name,
    round(SUM(order_details.quantity * pizzas.price),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 , pizza_types.name) as a) as b
where ranking<=3;
```

RESULTS

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category	name	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	32265.7
Veggie	The Mexicana Pizza	26780.75
Veggie	The Five Cheese Pizza	26066.5

DATA VISUALIZATION (USING POWER BI)

PIZZA SALES REPORT

Jan/15 - Dec/15

Pizza Category: All

817.86K	49574	21350	38.31	2.32
Total Revenue	Total Pizzas Sold	Total Orders	Avg Order Value	Avg Pizzas Per Order

BEST SELLERS

REVENUE

The Thai Chicken Pizza Contributes to maximum Revenue.

QUANTITY

The Classic Deluxe Pizza Contributes to maximum Total Quantities

TOTAL ORDERS

The Classic Deluxe Pizza Contributes to maximum Total Orders

WORST SELLERS

REVENUE

The Brie Carre Pizza Contributes to minimum Revenue.

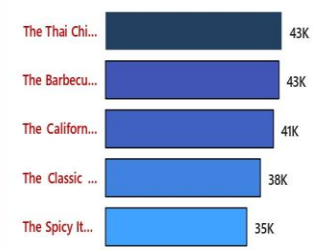
QUANTITY

The Brie Carre Pizza Contributes to minimum Total Quantities

TOTAL ORDERS

The Brie Carre Pizza Contributes to minimum Total Orders

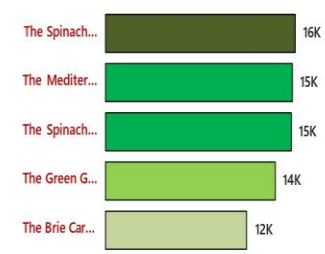
Top 5 Pizzas by Revenue



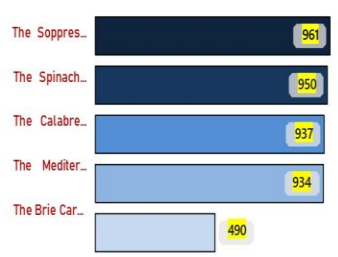
Top 5 Pizzas by order Quantity

Top 5 Pizzas by Total Orders

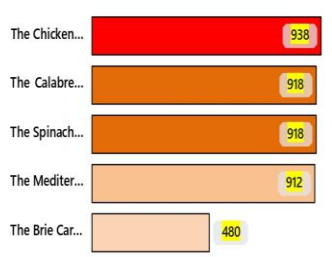
Bottom 5 Pizzas by Revenue



Bottom 5 Pizzas by Order Quantity



Bottom 5 Pizzas by Total Orders



DATA INSIGHTS

- · Strong Sales Volume: $817K revenue from 21K+ orders, indicating significant market demand.
- · Daily Average: 138 pizzas ordered daily, crucial for operational planning.
- · Peak Times: Lunch (12-1 PM) and early evening (4-7 PM) are peak ordering periods, vital for staffing. A late-night peak was also noted.
- · Popular Categories: Classic pizzas dominate (26.91% revenue, 14K+ units), followed by Supreme (25.46% revenue, 11K+ units).
- · Top Pizzas: Classic Deluxe is most ordered. Barbecue Chicken, Hawaiian, and Pepperoni are also popular. Thai Chicken and Barbecue Chicken are top revenue drivers.
- · Preferred Size: Large (L) is the most common size, guiding inventory. XL/XXL are less popular.
- · Category Revenue: Classic, Supreme, Chicken, and Veggie categories significantly contribute to revenue. Chicken, despite fewer options (around 6), shows growth potential.
- · Pricing: Greek Pizza is highest priced; The Brie Carre Pizza among the lowest.
- · Underperformance: XXL pizzas have low order numbers, suggesting a review.
- · Revenue Trend: Consistent growth indicates a healthy business.

RECOMMENDATIONS

- Optimize Inventory: Ensure sufficient stock of popular pizza types (especially Classic, Thai Chicken, and Barbecue Chicken) and the preferred large size.
- Strategic Promotions: Develop targeted promotions for off-peak hours and focus on popular categories like Classic and high-revenue generators like Thai Chicken and Barbecue Chicken. Consider boosting the visibility of the Chicken category.
- Staffing Optimization: Adjust staffing levels to meet the increased demand during peak lunch and evening hours.
- Product Portfolio Review: Consider discontinuing or re-evaluating underperforming products like XXL pizzas. Explore expanding the variety within the popular Chicken pizza category.
- Customer Engagement: Leverage the popularity of Classic pizzas and the Large size in marketing campaigns to attract and retain customers.

CONCLUSION

These data-driven insights, derived through rigorous SQL analysis and visualization, provide a solid foundation for informed decision-making and strategic initiatives aimed at enhancing pizza sales performance. By analyzing key metrics and relationships between orders, order details, specific pizzas, and pizza types, the project seeks to optimize sales strategies, improve operational efficiency, and enhance understanding of customer preferences within the pizza business.

THANK YOU

swastiksamadder011@gmail.com