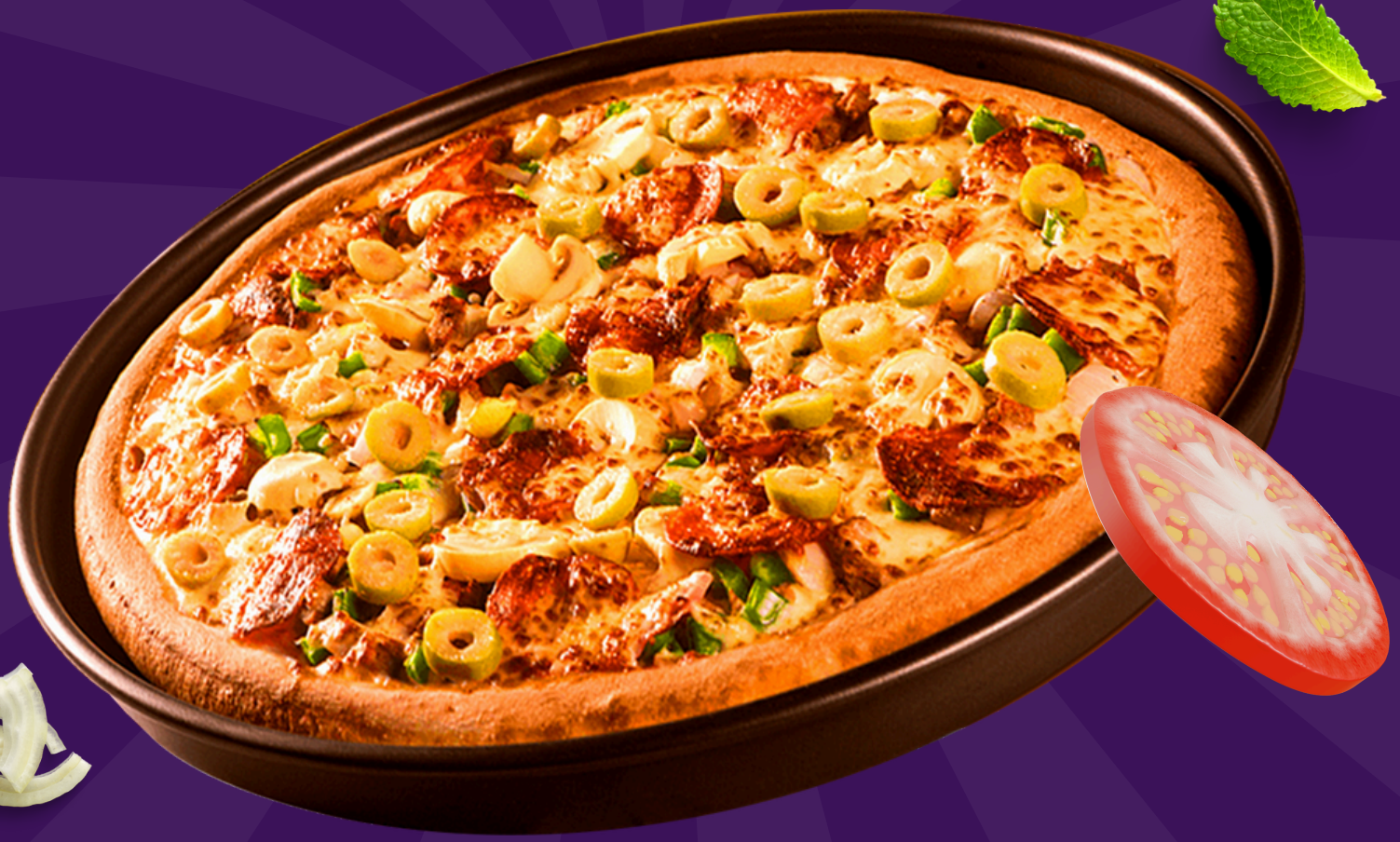
SQL PROJECT ON

PIZZA SALES



HELLO!

My name is Manasi Kulkarni and in this project i have utilized SQL queries to solve the questions related to Pizza Sales.

Welcome to my MySQL portfolio project on Pizza Sales Analysis. In this project, I have created a comprehensive database system to track and analyze sales data for a pizza business. The aim is to provide insights into various aspects of the business, such as sales trends, customer preferences and overall business performance.

This project involves the use of SQL queries, database design, and optimization techniques to effectively manage large volumes of sales data. By leveraging MySQL, I have built a robust relational database that allows for efficient data retrieval, reporting, and analysis.

Throughout this presentation, I will showcase how the system helps in making data-driven decisions, improving operational efficiency, and ultimately contributing to the growth of the pizza business.

Retrieve the total number of orders placed.

```
2 •    SELECT
3          COUNT(order_id) AS total_orders
4      FROM
5          orders;
```

Result Grid

	total_orders
▶	21350

Calculate the total revenue generated from pizza sales.

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

Result Grid

	total_revenue
▶	774579.55

Identify the highest-priced pizza.

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

Result Grid | Filter Rows:

	name	price
▶	The Greek Pizza	35.95

Identify the most common pizza size ordered.

```sql
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;
```

Result Grid | Filter Rows:

size	Order_count
L	17546
M	14584
S	13376
XL	515
XXL	27

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

```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;
```

Result Grid | Filter Rows:

name	quantity
The Classic Deluxe Pizza	2313
The Barbecue Chicken Pizza	2305
The Pepperoni Pizza	2300
The Hawaiian Pizza	2288
The California Chicken Pizza	2231

Join the necessary tables to find the total quantity of each pizza category ordered.

```
2  •  SELECT
3         SUM(order_details.quantity) AS Total_quantity,
4         pizza_types.category
5     FROM
6         order_details
7             JOIN
8         pizzas ON order_details.pizza_id = pizzas.pizza_id
9             JOIN
10        pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
11    GROUP BY pizza_types.category
12    ORDER BY Total_quantity DESC;
```

Result Grid | Filter Rows:

	Total_quantity	category
▶	14087	Classic
	11382	Supreme
	11030	Veggie
	10450	Chicken

Determine the distribution of orders by hour of the day.

```sql
SELECT
    HOUR(order_time) order_time_hour, COUNT(order_id) Orders
FROM
    orders
GROUP BY HOUR(order_time);
```

Result Grid | Filter Rows:

order_time_hour	Orders
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009
20	1642
21	1198

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

```
3 •    SELECT
4          category, COUNT(name) as Distribution
5      FROM
6          pizza_types
7      GROUP BY category;
```

Result Grid | Filter Rows:

	category	Distribution
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

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

```
3   SELECT
4       ROUND(AVG(quantity), 2) as Average
5   FROM
6       (SELECT
7           orders.order_date, SUM(order_details.quantity) quantity
8       FROM
9           orders
10      JOIN order_details ON orders.order_id = order_details.order_id
11      GROUP BY orders.order_date) AS order_quantity;
```

Result Grid

	Average
▶	138.90

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

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

Result Grid | Filter Rows:

Total_revenue	name
40783.75	The Thai Chicken Pizza
40508.75	The Barbecue Chicken Pizza
38985.25	The California Chicken Pizza

Calculate the percentage contribution of each pizza type to total revenue

```sql
SELECT
    pizza_types.category,
    ROUND(SUM(order_details.quantity * pizzas.price) / (SELECT
                    ROUND(SUM(order_details.quantity * pizzas.price),
                                2) AS Total_revenue
                FROM
                    order_details
                        JOIN
                    pizzas ON pizzas.pizza_id = order_details.pizza_id) * 100,
            2) AS Total_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 Total_revenue DESC;
```

Result Grid | Filter Row

category	Total_revenue
Classic	26.89
Supreme	25.52
Chicken	23.91
Veggie	23.68

Analyze the cumulative revenue generated over time.

```
3 •    select order_date,
4      sum(Revenue) over (order by order_date) as cum_revenue
5      from
6    (select orders.order_date, sum(order_details.quantity*pizzas.price)
7      as Revenue from order_details join pizzas
8      ON order_details.pizza_id= pizzas.pizza_id
9      join orders on orders.order_id= order_details.order_id
10     group by orders.order_date order by Revenue desc) as sales;
```

Result Grid | Filter Rows:

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

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

```sql
select name, revenue, category from
(select category,name,revenue,rank()
over(partition by category order by revenue desc) as ranking
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 ranking <=3;
```

Result Grid | Filter Rows: | Export:

name	revenue	category
The Thai Chicken Pizza	40783.75	Chicken
The Barbecue Chicken Pizza	40508.75	Chicken
The California Chicken Pizza	38985.25	Chicken
The Classic Deluxe Pizza	35977.5	Classic
The Hawaiian Pizza	30482.75	Classic
The Pepperoni Pizza	28695	Classic
The Spicy Italian Pizza	33200.75	Supreme
The Italian Supreme Pizza	31837	Supreme
The Sicilian Pizza	29202	Supreme
The Four Cheese Pizza	30599.400000000598	Veggie
The Mexicana Pizza	25623.5	Veggie

This project showcases my ability to design and optimize databases, write complex SQL queries, and generate actionable reports to support business growth. Moving forward, there are opportunities to expand this system with additional features, such as integrating real-time data or predictive analytics to forecast future sales.

Thank you for reviewing my project, and I look forward to any questions or feedback you may have!