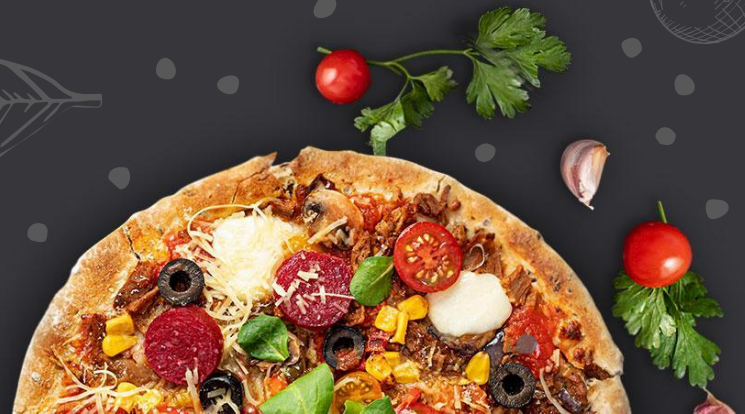
Pizza Sales Analysis

(SQL BASED PROJECT)



About the Project

1. This project focuses on analyzing pizza sales data using SQL to uncover valuable insights that can drive strategic business decisions.

2. The primary goal of this project is to leverage SQL queries to extract, analyze, and interpret key metrics such as total sales, customer preferences, popular pizza types, peak sales periods, and order trends. By analyzing this data, we aim to:

- Identify the most and least popular pizza types and sizes.
- Understand peak sales hours and days of the day to optimize staffing and inventory management.
- Analyze customer buying patterns to inform marketing strategies and promotional offers.
- Pizza types/categories distribution in revenue.

Database Structure and table schema

The database contains four tables Pizzas, Pizza_types, orders, order_details. Schema of the tables is as follows:

1. Pizzas

- Pizza_id
- Pizza_type_id
- Price

2. Pizza_types

- Pizza_type_id
- Name
- Category
- Ingerdients

3. Orders

- Order_id
- Order_date
- Order_time

4. Order_details

- Order_details_id
- Order_id
- Pizza_id
- Quantity

Let's Begin

1. Total No of orders placed this year.

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

2. Total Revenue generated this year pizza sales.

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

3.Total No of pizzas sold per month

```
SELECT
    MONTHNAME(orders.order_date) AS orders_placed,
    SUM(order_details.quantity) AS quantity_per_month
FROM
    orders
        JOIN
    order_details ON orders.order_id = order_details.order_id
GROUP BY orders_placed
ORDER BY quantity_per_month;
```

4. Identify the Highest Prized Pizza

```
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;
```

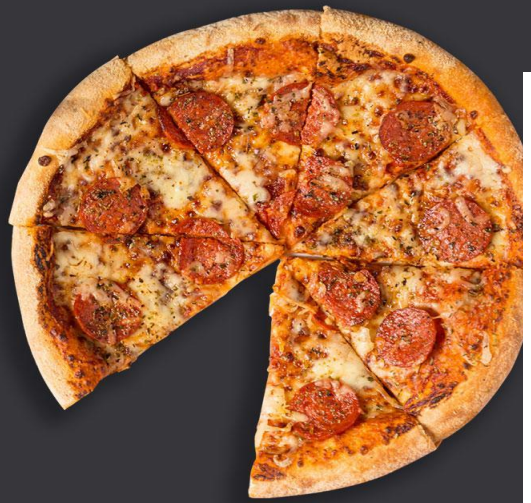
5. Identify the most common size pizza ordered.

```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
LIMIT 1;
```

6. Identify the highest order value made in the year.

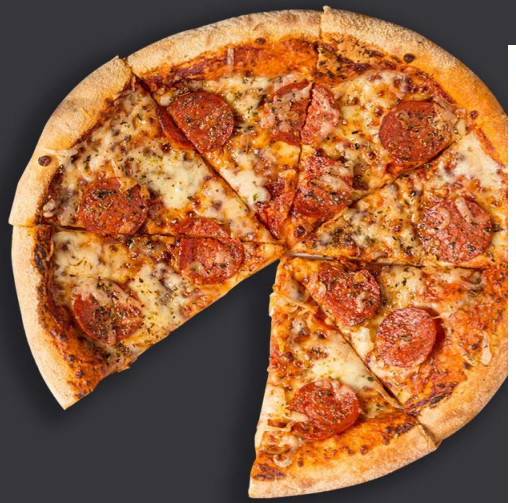
```sql
SELECT
    order_details.order_id,
    SUM(order_details.quantity * pizzas.price) AS order_value
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY order_details.order_id
ORDER BY order_value DESC
LIMIT 1;
```

7. List the top 5 most ordered pizzas along with their quantities



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

8. Find the total quantity of each pizza category ordered.



```sql
SELECT
    pizza_types.category AS pizza_category,
    SUM(order_details.quantity) AS total_quantity
FROM
    pizzas
        JOIN
    pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizza_category
ORDER BY total_quantity DESC;
```

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

```
SELECT
    HOUR(order_time) AS order_distribution,
    COUNT(order_id) AS orders_placed
FROM
    orders
GROUP BY order_distribution
ORDER BY orders_placed DESC;
```

10. Find the category-wise distribution of pizzas. i.e. How many types of pizza each category contain.

```
SELECT
    category, COUNT(name) AS no_of_pizzas
FROM
    pizza_types
GROUP BY category
ORDER BY no_of_pizzas DESC;
```

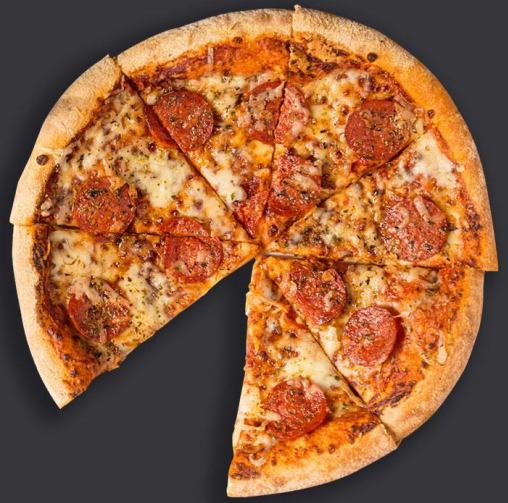
11. Calculate the average number of pizzas ordered per day.

```
SELECT
    ROUND(AVG(quantity_ordered), 0) AS avg_pizza_ordered_per_day
FROM
    (SELECT
        orders.order_date AS date,
            SUM(order_details.quantity) AS quantity_ordered
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY date
    ORDER BY quantity_ordered) AS total_order_per_day;
```

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

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

13. Determine the least 3 ordered pizza types based on revenue.



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

14. Calculate the percentage contribution of each pizza type/category to total revenue.

```
select pizza_types.category,
round(sum(order_details.quantity*pizzas.price)/(select round(sum(pizzas.price*order_details.quantity), 2) as revenue
from pizzas
join order_details
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 pizzas.pizza_id = order_details.pizza_id
group by pizza_types.category order by revenue desc;
```

15. Analyze the cumulative revenue generated over time.

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

16. Determine the top 3 most ordered pizza types(names) 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 pizzas.pizza_id = order_details.pizza_id
group by pizza_types.category, pizza_types.name) as a) as b
where rn<=3;
```

Thank You!