The SQLice of Pizza"
(Data Served Fresh!)

JYOTI BAWA

INTRODUCTION

This project explores pizza sales data using SQL to uncover key business insights. It involves analyzing orders, revenue, and customer preferences through joins, aggregations, window functions, and date-time analysis.

The goal was to identify top-selling pizzas, most profitable categories, and order trends — helping strengthen real-world SQL and data analytics skills.

RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

```sql
select count(order_id) as total_orders from orders;
```

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

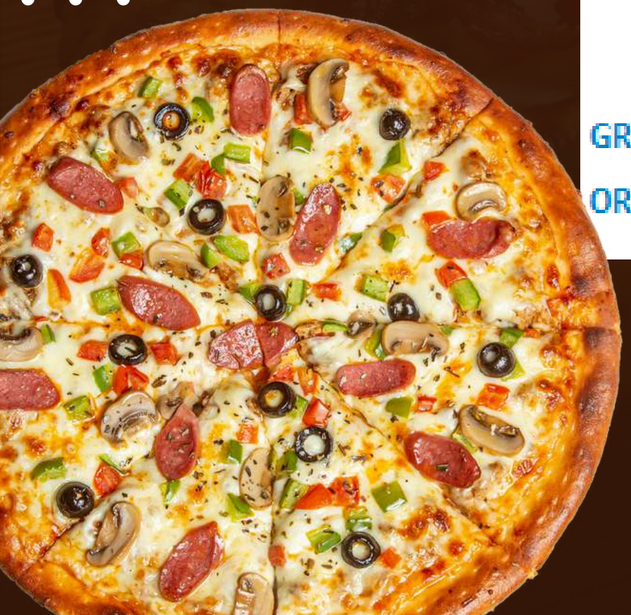
```sql
SELECT
    ROUND(SUM(order_details.quantity * pizzas.price))
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id;
```

IDENTIFY THE HIGHEST PRICED PIZZA.

```sql
 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;
```

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;
```



LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.

```sql
SELECT
    pizza_types.name, SUM(order_details.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 SUM(order_details.quantity) DESC
LIMIT 5;
```

JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

```sql
SELECT
    pizza_types.category, SUM(order_details.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 SUM(order_details.quantity) DESC;
```

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

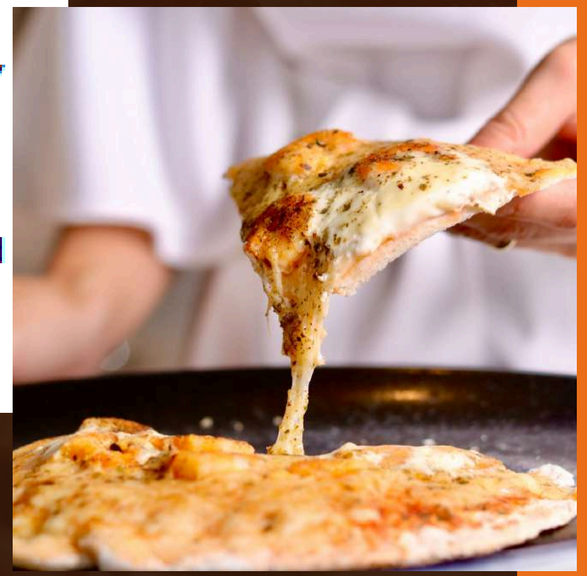
```sql
SELECT
    HOUR(order_time), COUNT(order_id)
FROM
    orders
GROUP BY HOUR(order_time);
```

JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

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

GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

```sql
SELECT
    ROUND(AVG(quantity), 0)
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;
```



DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```sql
SELECT
    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.name
ORDER BY revenue DESC
LIMIT 3;
```

CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

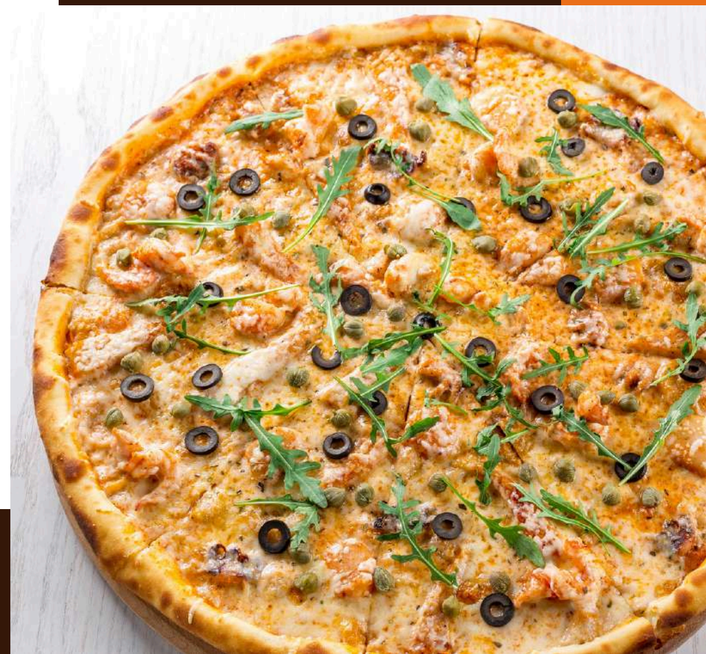
```sql
SELECT
    pizza_types.category,
    ROUND(SUM(order_details.quantity * pizzas.price) / (SELECT
                    ROUND(SUM(order_details.quantity * pizzas.price))
                FROM
                    order_details
                        JOIN
                    pizzas ON pizzas.pizza_id = 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;
```

ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```sql
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;
```

DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.

```sql
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;
```



CONCLUSION

This SQL project helped me analyze pizza sales data and extract valuable business insights using advanced SQL techniques.

- Applied real-world SQL queries on a structured sales dataset
- Identified best-selling pizzas and top revenue-generating categories
- Used JOINs, GROUP BY, ORDER BY, and aggregations
- Implemented CTEs and window functions like RANK() and SUM() OVER
- Explored sales trends by date, time, and pizza size

"Every slice of data tells a story — thank you for being a part of mine." 🍕

THANK YOU FOR ATTENTION