Pizza Sales Analysis

By: Tirth N. Sarswat

👨‍💻About

My name is **Tirth N. Sarswat**, and in this project, I have utilized SQL to analyze and solve a series of business-related questions based on pizza sales data. The queries cover a wide range of insights, including;

- Total orders,
- Revenue generation
- Top-performing pizzas
- Category-wise distribution, and
- Customer ordering behavior by time.

Each question is solved using structured and optimized SQL queries. This presentation is intended to serve as a clear and well-documented summary of the analytical findings

📦What, we find

We are working with a pizza sales database containing the following key tables:

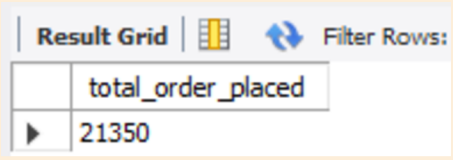
- **order_details:** order_details_id, order_id, pizza_id, quantity
- orders: order_id, order_date, order_time
- pizzas: pizza_id, pizza_type_id, size, price
- pizza_types: pizza_type_id, name, category, ingredients

🎯 What We Need to Do

- **Using SQL, we aim to answer the following questions:**
  - Retrieve the total number of orders placed.
  - Calculate the total revenue generated from pizza sales.
  - Identify the highest-priced pizza.
  - Identify the most common pizza size ordered.
  - List the top 5 most ordered pizza types along with their quantities.
  - Join the necessary tables to find the total quantity of each pizza category ordered.
  - Determine the distribution of orders by hour of the day.
  - Join relevant tables to find the category-wise distribution of pizzas.
  - Group the orders by date and calculate the average number of pizzas ordered per day.
  - Determine the top 3 most ordered pizza types based on revenue.
  - Calculate the percentage contribution of each pizza type to tota revenue.
  - Analyze the cumulative revenue generated over time.
  - Determine the top 3 most ordered pizza types based on revenue for each pizza category.

Retrieve the total number of unique orders placed.

```sql
SELECT
    COUNT(DISTINCT order_id) AS total_order_placed
FROM
    orders;
```



Result Grid | Filter Rows:

	total_order_placed
▶	21350

Calculate the total revenue generated from pizza sales.

```
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
817860.05

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;
```

Result Grid | Filter Rows:

name	price
The Greek Pizza	35.95

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;
```

Result Grid | Filter Rows:

size	order_count
L	18526

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

```
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	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

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;
```

Result Grid | Filter Rows:

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Determine the distribution of orders by hour of the day.

```
select Hour(order_time) as hours, count(order_id) as Order_Count
from orders
group by hours
order by Order_Count desc;
```

Result Grid | Filter Rows:

hours	Order_Count
12	2520
13	2455
18	2399
17	2336
19	2009
16	1920
20	1642
14	1472
15	1468
11	1231
21	1198
22	663
23	28
10	8
9	1

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

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

Result Grid | Filter Rows:

category	Category_Count
Supreme	9
Veggie	9
Classic	8
Chicken	6

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

```
SELECT
    ROUND(AVG(PizzaQuantity_Ordered), 0) AS avg_pizzas_ordered_per_day
FROM
    (SELECT
        orders.order_date,
            SUM(order_details.quantity) AS PizzaQuantity_Ordered
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY orders.order_date) AS Order_Quantity;
```

Result Grid | Filter Rows:

avg_pizzas_ordered_per_day
138

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;
```

Result Grid | Filter Rows:

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

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

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

Classic	26.91
Veggie	23.68
Supreme	25.46
Chicken	23.96

Analyze the cumulative revenue generated over time.

```
select order_date, round(sum(revenue) over (order by order_date),2) as Cum_Revenue
from (select orders.order_date,
round(sum(order_details.quantity * pizzas.price),2) as revenue
from
orders
join order_details
on orders.order_id = order_details.order_id
join pizzas
on order_details.pizza_id = pizzas.pizza_id
group by orders.order_date) as sales;
```

Result Grid | Filter Rows:

order_date	Cum_Revenue
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
2015-01-11	25862.65

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

```sql
select category, 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, 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 pizzas.pizza_id = order_details.pizza_id
group by pizza_types.category, pizza_types.name) as a) as b
where rn<=3;
```

Result Grid | Filter Rows:

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

✅ Project Notes

- All insights are based on structured SQL queries.
- Explored real-time data across multiple related tables.
- Gained valuable insights such as peak ordering times, best-selling pizzas, and high-value products.

Thank You