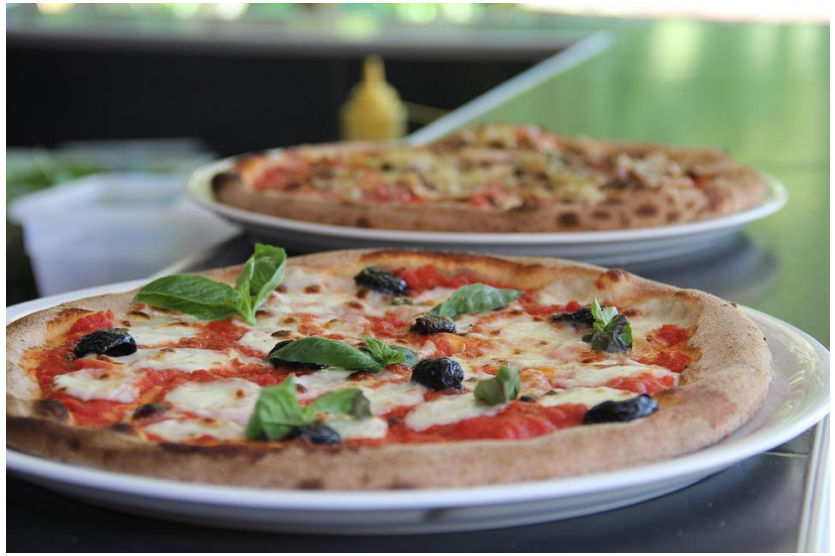
Pizza sales Analysis using SQL by : Ankita Rawat



This project analyzes pizza sales data using SQL to uncover key business insights. By leveraging **subqueries, GROUP By and Joins,** important patterns in sales and customer preferences were identified.

- Key Objectives & Insights

- Revenue Analysis: Calculated the total revenue generated and analyzed the cumulative revenue over time to track business growth.
- Product Performance: Determined the % contribution of each pizza type to total revenue and identified the highest-priced pizza to understand sales impact .
- Customer Preferences: Found the total quantity of each pizza category ordered, helping in optimizing stock levels.
- Sales Trends: Analyzed the distribution of orders by hour of the day to identify peak sales hours.

- *This project showcases how SQL-based data analysis can provide actionable insights for business strategy and decision-making.*

1. Retrieve the total number of orders placed.

```
-- Retrieve the total number of orders placed.

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

Result Grid

total_orders
21350

2. -- Calculate the total revenue generated from pizza sales.

```sql
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
```

Result Grid

	total_sales
▶	817860.05

3. -- Identify the highest-priced pizza.

```sql
SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
ORDER BY price DESC
LIMIT 1
```

Result Grid | Filter Rows:

name	price
The Greek Pizza	35.95

4. -- 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 order_details.pizza_id = pizzas.pizza_id
GROUP BY pizzas.size
ORDER BY COUNT(order_details.order_details_id) DESC
LIMIT 1
```

Result Grid | Filter Rows:

size	order_count
L	18526

5. -- 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 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 SUM(order_details.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

Result 7

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

```sql
SELECT
    SUM(order_details.quantity) AS quantity,
    pizza_types.category
FROM
    pizza_types
        JOIN
    pizzas 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.category
ORDER BY SUM(order_details.quantity)
```

Result Grid | Filter

quantity	category
11050	Chicken
11649	Veggie
11987	Supreme
14888	Classic

7. -- determine the distribution of orders by hour of the day.

```
SELECT
    HOUR(order_time) AS hour, COUNT(order_id) AS count
FROM
    orders
GROUP BY HOUR(order_time)
order by  COUNT(order_id) desc
```

Result Grid

hour	count
12	2520
13	2455
18	2399
17	2336
19	2009
16	1920

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

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

Result Grid | Filter Row:

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

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

```sql
SELECT
    ROUND(AVG(quantity), 2)
FROM
  (SELECT
        orders.order_date, SUM(order_details.quantity) AS quantity
    FROM
        orders
    JOIN order_details ON order_details.order_id = orders.order_id
    GROUP BY orders.order_date) AS order_quntity
```

	ROUND(AVG(quantity), 2)
▶	138.47

10. -- 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 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 SUM(order_details.quantity * pizzas.price) 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

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 pizzas.pizza_type_id = pizza_types.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 * pizzas.price) DESC
```

Result Grid Filte

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

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

```
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
```

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

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

Result Grid | Filter Rows:

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The Classic Deluxe Pizza	38180.5
The Hawaiian Pizza	32273.25
The Spicy Italian Pizza	34831.25
The Italian Supreme Pizza	33476.75

conclusion

- This project demonstrates how SQL can be used to analyze pizza sales data and extract key insights to optimize business strategies.
- By using subqueries, GROUP BY, and JOINs, the following insights were achieved:

- *Total revenue generated and cumulative revenue trends for performance tracking.*
- *% contribution of each pizza type to identify top-selling items.*
- *Insights into customer preferences through total quantity ordered by category.*
- *Analysis of order distribution by hour to optimize staffing.*

- These findings offer valuable data-driven strategies to enhance sales, improve inventory management, and better serve customers.