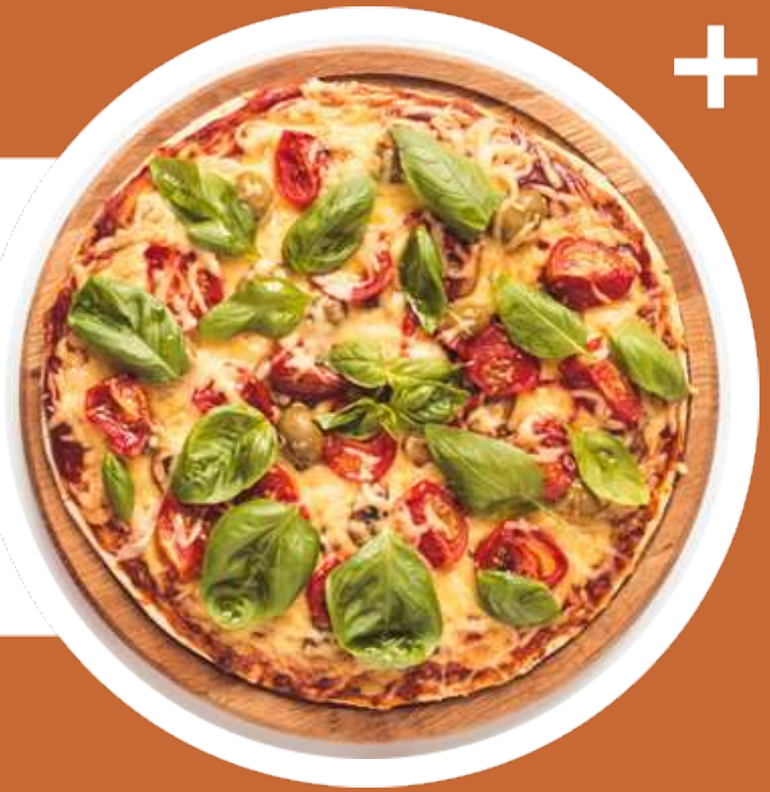
PIZZA SALES ANALYSIS

Uncovering Trends through Interactive Insights

ON SQL



In this project,I have utilized SQL queries to solve different questions related to pizza sales, showcasing the power of SQL in unlocking business intelligence from pizza sales data

Questions related to our project

1)Extract the total number of orders placed.
2)Calculate the total revenue generated from pizza sales.
3)Identify the highest-priced pizza.
4)Identify the most common pizza size ordered.
5)List the top 5 most ordered pizza types along with their quantities.
6)Join the necessary tables to find the total quantity of each pizza category ordered.
7)Determine the distribution of orders by hour of the day.
8)Join relevant tables to find the category-wise distribution of pizzas.
9)Group the orders by date and calculate the average number of pizzas ordered per day.
10)Determine the top 3 most ordered pizza types based on revenue.
11)Calculate the percentage contribution of each pizza type to total revenue.
12)Analyze the cumulative revenue generated over time.
13)Determine the top 3 most ordered pizza types based on revenue for each pizza category.

```sql
-- Extract the total number of orders placed

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

Result Grid

total_orders
21350

```sql
-- Calculate the total revenue generated from pizza sales

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

Result Grid

	total_revenue
▶	817860.05

```sql
-- 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

```sql
-- Identify the most common pizza size ordered

SELECT
    pizzas.size,
    COUNT(orders_details.order_details_id) AS order_count
FROM
    pizzas
        JOIN
    orders_details ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC
LIMIT 1;
```

Result Grid | Filter Rows:

size	order_count
L	18526

```sql
-- List the 5 most ordered pizza types along with their quantities

SELECT
    pizza_types.name, SUM(orders_details.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
        JOIN
    orders_details ON orders_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

```sql
-- Join the necessary tables to find the total quantity of each pizaa
-- category ordered

SELECT
    pizza_types.category, SUM(orders_details.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
        JOIN
    orders_details ON orders_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

```sql
-- Determine the distribution of orders by hour of the day

SELECT
    hour(order_time) AS Hours, COUNT(order_id)
FROM
    orders
GROUP BY Hours;
```

Result Grid | Filter Rows:

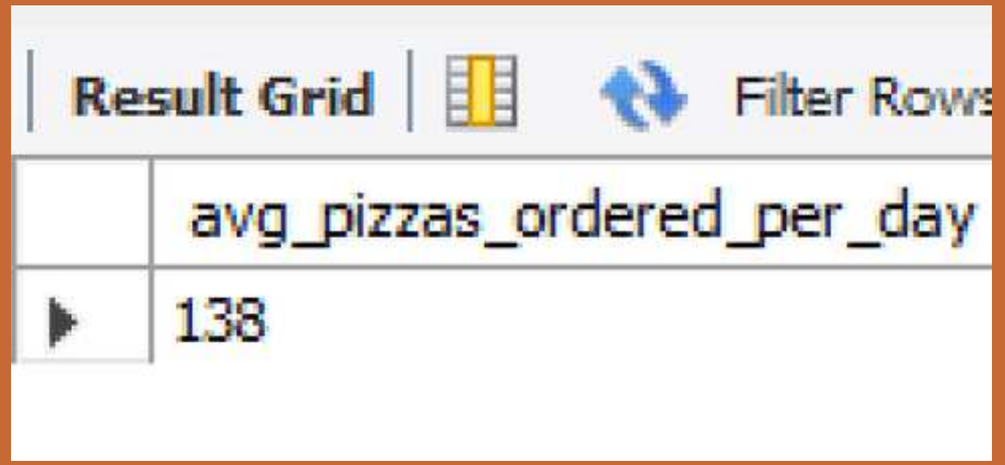
Hours	COUNT(order_id)
11	1231
12	2520
13	2455
14	1472
15	1468

```sql
-- Join relevant tables to find the category-wise pizza
-- distribution of pizzas

select category,count(name) from pizza_types

group by category;
```

Result Grid | Filter Rows:

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

```sql
-- Group the orders by date and calculate the average
-- number of pizzas order per day

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



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


SELECT
    pizza_types.name AS pizza_type,
    ROUND(SUM(orders_details.quantity * pizzas.price),
            0) AS revenue
FROM
    pizzas
        JOIN
    pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_i
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_type
ORDER BY revenue DESC
LIMIT 3;
```

Result Grid | Filter Rows:

pizza_type	revenue
The Thai Chicken Pizza	43434
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41410

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

SELECT
    pizza_types.category AS pizza_type,
    ROUND((SUM(orders_details.quantity * pizzas.price) / (SELECT
                    ROUND(SUM(orders_details.quantity * pizzas.price),
                                0) AS total_revenue
                FROM
                    orders_details
                        JOIN
                    pizzas ON orders_details.pizza_id = pizzas.pizza_id)) * 100,
            2) AS revenue
FROM
    pizzas
        JOIN
    pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_type
ORDER BY revenue DESC
```

Result Grid | Filter Rows:

pizza_type	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

```sql
-- Analyze the cumulative revenue generated over time.

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

```

Result Grid | Filter Rows:

order_date	cumulative_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

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


select name, category, revenue
from
(select category,name,revenue,
rank() over(partition by category order by revenue desc) as rn
from
(select pizza_types.category as category, pizza_types.name as name,
       sum(orders_details.quantity * pizzas.price) as revenue
from pizza_types join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join orders_details
on orders_details.pizza_id = pizzas.pizza_id
group by category, name) as a) as b
where rn <=3;
```

Result Grid | Filter Rows: | Export:

name	category	revenue
The Thai Chicken Pizza	Chicken	43434.25
The Barbecue Chicken Pizza	Chicken	42768
The California Chicken Pizza	Chicken	41409.5
The Classic Deluxe Pizza	Classic	38180.5
The Hawaiian Pizza	Classic	32273.25
The Pepperoni Pizza	Classic	30161.75
The Spicy Italian Pizza	Supreme	34831.25
The Italian Supreme Pizza	Supreme	33476.75
The Sicilian Pizza	Supreme	30940.5

This analysis summarizes the key findings and provides actionable recommendations to leverage the insights. We can suggest optimizing menus, focusing on high-demand pizzas during peak hours, or running targeted promotions. This data empowers informed decision-making to maximize profitability.