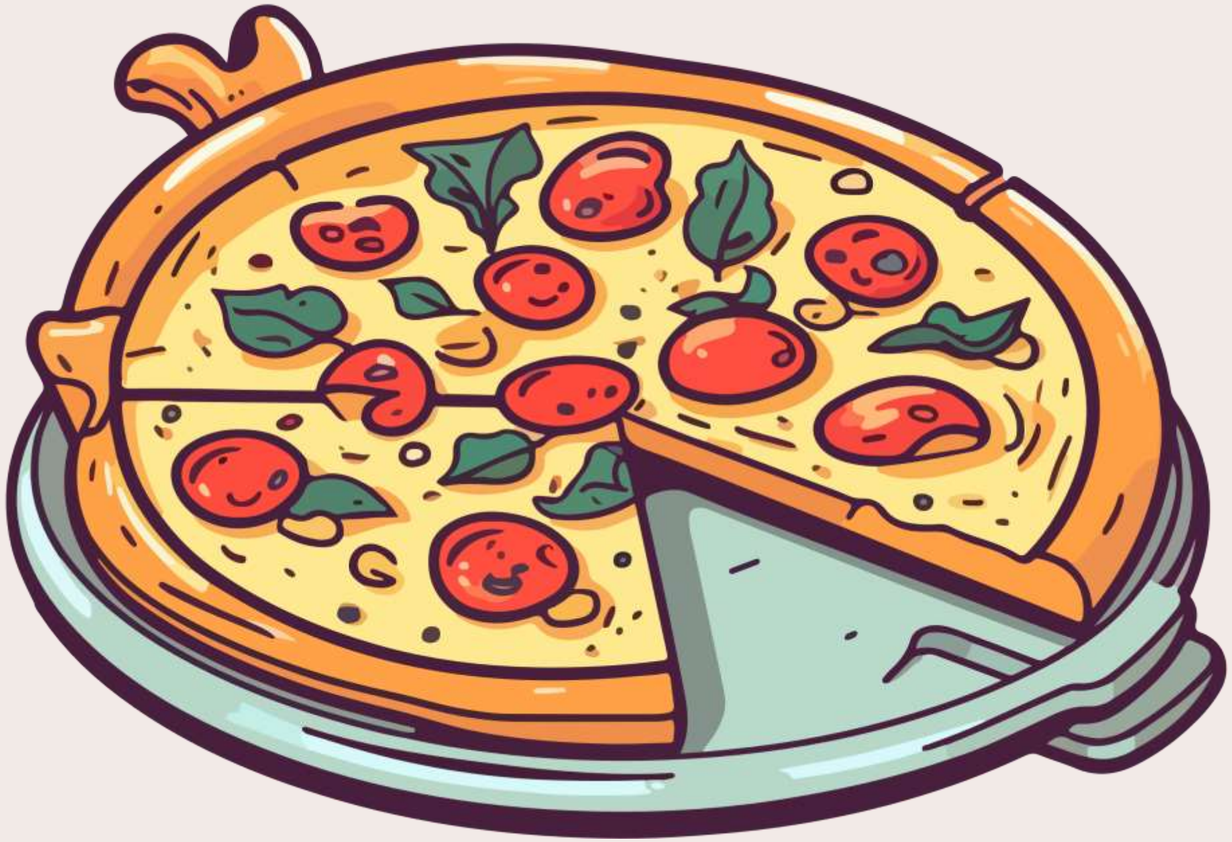
SQL Project for analyzing Pizza sale



By Anoop Kumar

HELLO

I am Anoop Kumar. In this project, I employed SQL queries to address questions pertaining to pizza sales. By analyzing the data, I was able to extract valuable insights and generate comprehensive reports to inform business decisions.

Questions

Level 1

- Calculate the total revenue generated from pizza sales.
- Identify the most common pizza size ordered.
- List the top 5 most ordered pizza types along with their quantities.
- Determine the distribution of orders by hour of the day.
- Group the orders by date and calculate the average number of pizzas ordered per day.

Level 2

- Join the necessary tables to find the total quantity of each pizza category ordered.
- Determine the top 3 most ordered pizza types based on revenue.
- Calculate the percentage contribution of each pizza type to total revenue.
- Analyze the cumulative revenue generated over time.

Here are the Input data in word file :

Orders details -

Order details

Orders -

Orders

Pizza Type -

Pizza type

Pizza List -

Pizza list

Calculate the total revenue generated from pizza sales

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

	Revenue
▶	817860.05

Identify the most common pizza size ordered

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

	size	Common_pizzas
▶	L	18526

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

```sql
select pizza_types.name, count(order_details.order_details_id) as Most_ordered
from pizzas join order_details
on pizzas.pizza_id = order_details.pizza_id
join pizza_types
on pizzas.pizza_type_id = pizza_types.pizza_type_id
group by pizza_types.name order by  count(order_details.order_details_id) desc
limit 5;
```

name	Most_ordered
The Classic Deluxe Pizza	2416
The Barbecue Chicken Pizza	2372
The Hawaiian Pizza	2370
The Pepperoni Pizza	2369
The Thai Chicken Pizza	2315

Determine the distribution of orders by hour of the day

```
select hour(order_time) as Hour, count(order_id) as orders from pizzahut.orders
group by hour(order_time);
```

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

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

```
select order_date, count(order_id) from pizzahut.orders
group by order_date;
```

	name	Total_quantity
▶	The Classic Deluxe Pizza	2453
	The Barbecue Chicken Pizza	2432
	The Hawaiian Pizza	2422
	The Pepperoni Pizza	2418
	The Thai Chicken Pizza	2371
	The California Chicken Pizza	2370
	The Spicy Italian Pizza	1924
	The Sicilian Pizza	1938
	The Southwest Chicken Pizza	1917
	The Four Cheese Pizza	1902
	The Italian Supreme Pizza	1884
	The Big Meat Pizza	1914
	The Vegetables + Vegetable...	1526
	The Mexicana Pizza	1484
	The Napolitana Pizza	1464
	The Spinach and Feta Pizza	1446
	The Prosciutto and Arugula ...	1457
	The Pepper Salami Pizza	1446
	The Italian Capocollo Pizza	1438
	The Greek Pizza	1420

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

```sql
select pizza_types.name, 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_types.name order by  count(order_details.quantity) desc ;
```

name	Total_quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371
The California Chicken Pizza	2370
The Spicy Italian Pizza	1924
The Sicilian Pizza	1938
The Southwest Chicken Pizza	1917
The Four Cheese Pizza	1902
The Italian Supreme Pizza	1884
The Big Meat Pizza	1914
The Vegetables + Vegetable...	1526
The Mexicana Pizza	1484
The Napolitana Pizza	1464
The Spinach and Feta Pizza	1446
The Prosciutto and Arugula ...	1457
The Pepper Salami Pizza	1446
The Italian Capocollo Pizza	1438
The Greek Pizza	1420

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 pizzas.pizza_id = order_details.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```

	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 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 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;
```

	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

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;
```

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
2015-01-07	16560.7
2015-01-08	19399.05
2015-01-09	21526.4
2015-01-10	23990.350000000002
2015-01-11	25862.65
2015-01-12	27781.7
2015-01-13	29831.300000000003
2015-01-14	32358.700000000004
2015-01-15	34343.50000000001
2015-01-16	36937.65000000001
2015-01-17	39001.75000000001
2015-01-18	40978.600000000006
2015-01-19	43365.75000000001
2015-01-20	45763.65000000001
2015-01-21	47804.20000000001

Thank You

In this project, I successfully completed numerous tasks using SQL queries. These queries were essential for retrieving, updating, and analyzing data, addressing various business requirements. The optimized SQL solutions enhanced data processing efficiency and significantly improved our database performance, leading to better data management and insightful reporting.