

---

---

# PIZZA SALES ANALYSIS

## SQL PROJECT

---

BY : RISHIKA BANSAL

---

# Introduction

Welcome to my SQL Project Presentation!

In this project, I embarked a journey from **BASIC to ADVANCED** SQL queries, exploring the power and versatility of Structured Query Language. From simple **SELECT** statements to complex **JOINS** and **SUBQUERIES**, I delve into a real world scenarios to demonstrate the practical application of SQL in **DATA MANIPULATION** and **ANALYSIS**. Join me as I unlock the potential of SQL to extract the insights and drive decision-making.

Let's dive in!

### **Basic level:-**

1. Retrieve total number of orders placed.
2. Calculate the total revenue generated from Pizzahut.
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.

### **Intermediate level:-**



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.

### **Advanced level:-**

11. Calculate the percentage contribution of each pizza type to total revenue.
12. Analyse the cumulative revenue generated over time.
13. Determine the top 3 most ordered pizza types based on revenue for each pizza category.


# Retrieve total number of orders placed

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

Result Grid  	
	total_orders
▶	21350

# Calculate the total revenue generated from pizzahut

```
select  
round(sum(order_details.quantity * pizzas.price), 2) as total_sales  
from order_details join pizzas  
on order_details.pizza_id = pizzas.pizza_id;
```

Result Grid 	
	total_sales
▶	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;
```

	name	price	
▶	the brie carre pizza	23.65	

# Identify the most common pizza size ordered

```
select pizzas.size, count(order_details.order_details_id) as pizza_count
from pizzas join order_details
on pizzas.pizza_id = order_details.pizza_id
group by pizzas.size
order by pizza_count desc;
```

	size	pizza_count	
▶	L	18526	
	M	15385	
	S	14137	
	XL	544	
	XXL	28	

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

```
select pizza_types.name, sum(order_details.quantity) as pizza_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 pizza_quantity desc limit 5;
```

	name	pizza_quantity	
►	the classic deluxe pizza	2453	
	the barbecue chicken pizza	2432	
	the big meat mizza	1914	
	the pepper salami pizza	1446	
	the five cheese pizza	1409	



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

```
select pizza_types.category, sum(order_details.quantity) as pizza_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 pizza_quantity asc;
```

	category	pizza_quantity	
▶	supreme	1936	
	veggie	2406	
	chicken	3419	
	classic	4367	

# Determine the distribution of orders by hour of the day

```
select hour(time) as hours, count(order_id) as count
from orders
group by hours;
```

	hours	count
▶	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



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

```
select category, count(name)
from pizza_types
group by category;
```

	category	count(name)	
▶	chicken	2	
	classic	2	
	supreme	2	
	veggie	2	

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

```
select round(avg(pizza_ordered), 0)
from
(select orders.date, count(order_details.order_details_id) as pizza_ordered
from orders join order_details
on orders.order_id = order_details.order_id
group by orders.date) as data;
```

Result Grid  	
	round(avg(pizza_ord
▶	136

# Determine the top 3 most ordered pizza types based on revenue

```
select pizza_types.name,  
       round(sum(order_details.quantity * pizzas.price), 0) 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;
```

	name	revenue
▶	the barbecue chicken pizza	42768
	the classic deluxe pizza	38180
	the five cheese pizza	26066

# Calculate the precentage 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 order_details.pizza_id = pizzas.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	7.48	
	chicken	7.3	
	veggie	4.89	
	supreme	4.54	

# Analyze the cumulative revenue generated over time

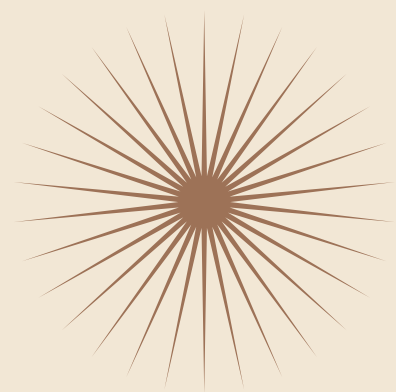
```
select date, sum(revenue)
over
(order by date) as cum_rev
from
(select orders.date, round(sum(order_details.quantity * pizzas.price), 2) as revenue
from order_details join pizzas
on order_details.pizza_id = pizzas.pizza_id
join orders
on orders.order_id = order_details.order_id
group by orders.date)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



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

```
select category, name, revenue
from
  (select category, name, revenue,
    rank() over(partition by category
      order by revenue desc) as ranks
  )
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 ranks<=3;
```





THANKS

