

# SQL PROJECT ON PIZZA SALES

01 ,Nov,2024

**Hellow**

**My name is Md Umar. In this project, I have utilized SQL queries to solve a problem related to pizza sales.**



# INTRODUCTION

Hello, I'm MD Umar. In this project, I leveraged SQL to analyze pizza sales data, addressing specific challenges to uncover customer behavior and sales trends. I utilized key SQL concepts, including JOINS, subqueries, and window functions, as well as foundational commands like ORDER BY, GROUP BY, and more, to derive insights that can help businesses make data-driven decisions. This project showcases my skills in SQL and data analysis, presenting valuable findings for the food industry.

# PROBLEMS

01 Retrieve the total number of orders placed.

SQL query

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

OUTPUT

Result Grid	
	Total_orders
▶	21350



# PROBLEMS

**02** Calculate the total revenue generated from pizza sales.

## SQL query

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

## OUTPUT

Result Grid	
	Total_Sales
▶	817860.05

# PROBLEMS

03 Identify the highest\_priced pizza.

## SQL query

```
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 3;
```

## OUTPUT

Result Grid			Filter Rows:
	name	price	
▶	The Greek Pizza	35.95	
	The Greek Pizza	25.5	
	The Brie Carre Pizza	23.65	

# PROBLEMS

04 Identify the most common pizza size ordered.

## SQL query

```
SELECT quantity, COUNT(order_details_id)
FROM orders_details
GROUP BY quantity;
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;
```

## OUTPUT

Result Grid			Filter Rows:
	quantity	COUNT(order_details_id)	
▶	1	47693	
	2	903	
	3	21	
	4	3	

# PROBLEMS



05

List the Top 5 most ordered pizza types along with their Quantities.

## SQL query

```
SELECT
    pizza_types.name, SUM(orders_details.quantity) AS Quantity
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 pizza_types.name
ORDER BY Quantity DESC
LIMIT 5;
```

## OUTPUT

Result Grid     Filter Rows: <input type="text"/>		
	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

# PROBLEMS

06

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

## SQL query

```
SELECT
    pizza_types.category,
    SUM(orders_details.quantity) AS quantity
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 pizza_types.category
ORDER BY quantity DESC;
```

## OUTPUT

Result Grid			Filter R
	category	quantity	
▶	Classic	14888	
	Supreme	11987	
	Veggie	11649	
	Chicken	11050	



# PROBLEMS

07

Determine the distribution of orders by hours of the day

SQL query

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

OUTPUT

Result Grid			Filter Rows
	hour	order_count	
▶	11	1231	
	12	2520	
	13	2455	
	14	1472	
	15	1468	
	16	1920	
	17	2336	
	18	2399	
	19	2009	

# PROBLEMS



08

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

SQL query

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

OUTPUT

Result Grid     Filter Rows:		
	category	COUNT(name)
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

# PROBLEMS

09

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

SQL query

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

OUTPUT

Result Grid		Filter Rows:
	avg_pizza_ordered_per_day	
▶	138	

# PROBLEMS



10

Determine the Top 3 most ordered pizza types based on revenue.

## SQL query

```
SELECT
    pizza_types.name,
    SUM(orders_details.quantity * pizzas.price) AS revenue
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 revenue DESC
LIMIT 3;
```

## OUTPUT

Result Grid     Filter Rows:		
	name	revenue
▶	The Thai Chicken Pizza	43434.25
	The Barbecue Chicken Pizza	42768
	The California Chicken Pizza	41409.5

# PROBLEMS



11

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

## SQL query

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

## OUTPUT

Result Grid     Filter		
	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

# PROBLEMS

12

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

## SQL query

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

## OUTPUT

	order_date	cum_revenue
▶	2015-01-01	2713.85000000000004
	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



# PROBLEMS



13

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

## SQL query

```
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(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 pizza_types.category, pizza_types.name) as a) as b
where rn <= 3;
```

## OUTPUT

Result Grid     Filter Rows: <input type="text"/>		
	name	revenue
▶	The Thai Chicken Pizza	43434.25
	The Barbecue Chicken Pizza	42768
	The California Chicken Pizza	41409.5
	The Classic Deluxe Pizza	38180.5
	The Hawaiian Pizza	32273.25
	The Pepperoni Pizza	30161.75
	The Spicy Italian Pizza	34831.25

# EXECUTIVE SUMMARY

Certainly, here's a polished executive summary for your SQL project:

Executive Summary:

In this project, I leveraged SQL queries to address key challenges related to pizza sales, focusing on analyzing customer behavior and sales trends. By implementing advanced SQL concepts such as JOINS, subqueries, and window functions, along with foundational commands like ORDER BY, GROUP BY, and DESC, I transformed raw data into actionable insights. This analysis provided a clearer understanding of purchasing patterns, popular menu items, and peak sales times, enhancing strategic decision-making capabilities. This project showcases my ability to apply SQL for real-world business insights, emphasizing my skills in data analysis, problem-solving, and critical thinking essential for industry roles.

# THANK YOU

01 Nov, 2024