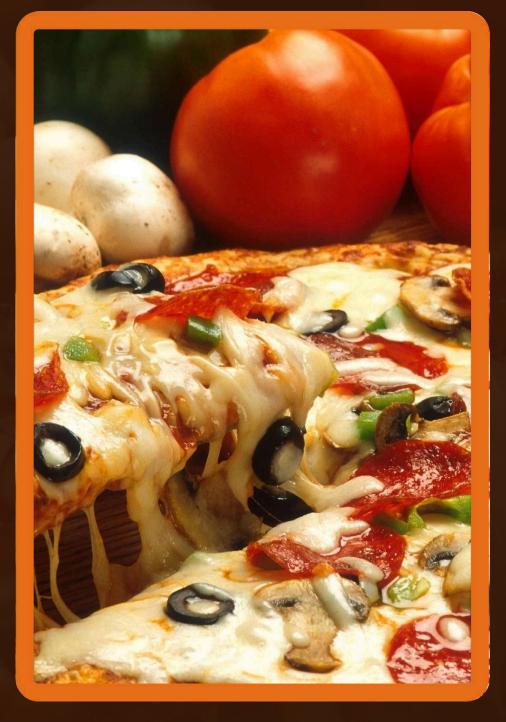
Where Every Slice is a Taste of Perfection



PROJECT











PIZZA SALES PROJECT

I'm Chandan Patel, and this is my Pizza Sales Analysis project using SQL. The database includes four maintables: orders, order_details, pizzas, and pizza_types. I used SQL to analyze sales data and answerreal business questions like:

- Which pizzas sell the most?
- What are the peak order times?
- Which sizes and categories perform best?

This project helped me strengthen my skills in data analysis, joins, aggregations, and reporting using SQL for real-world use cases.





SELECT count(Order_id) as total_orders from orders;





CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES

```
SELECT

ROUND(SUM(order_details.quantity * pizzas.price),

1) AS total_revenue

FROM

order_details

JOIN

pizzas ON pizzas.pizza_id = order_details.pizza_id;
```





IDENTIFY THE HIGHEST-PRICED PIZZA

```
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;
```



Re	esult Grid	Filter Row
	NAME	price
•	The Greek Pizza	35.95

IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED

```
pizzas.size,

COUNT(order_details.order_details_id) AS order_count

FROM

order_details

JOIN

pizzas ON pizzas.pizza_id = order_details.pizza_id

GROUP BY pizzas.size

ORDER BY order_count DESC

LIMIT 1;
```



Re	esult Grid		47	Filter
	size	order	coun	t
Þ	L	569		

LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES

```
SELECT
    pizza_types.name, SUM(order_details.quantity) AS 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 quantity DESC
LIMIT 5;
```



PC	esult Grid 📗 🙌 Filter F	(OWS:
	name	quantity
١	The Pepperoni Pizza	85
	The Classic Deluxe Pizza	73
	The Thai Chicken Pizza	70
	The Hawaiian Pizza	66
	The Italian Supreme Pizza	65

JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED

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



Result Grid		
	category	quantity
Þ	Classic	434
	Supreme	359
	Veggie	355
	Chicken	307

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY

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

Result



K	esult Gri	0 1111 1	•
	hour	count	
>	11	1231	
	12	2520	
	13	2455	
	14	1472	
	15	1468	
	16	1920	
	17	2336	
	18	2399	
	19	2009	

Pocult Guid

20	1642
21	1198
22	663
23	28
10	8
9	1

JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS

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

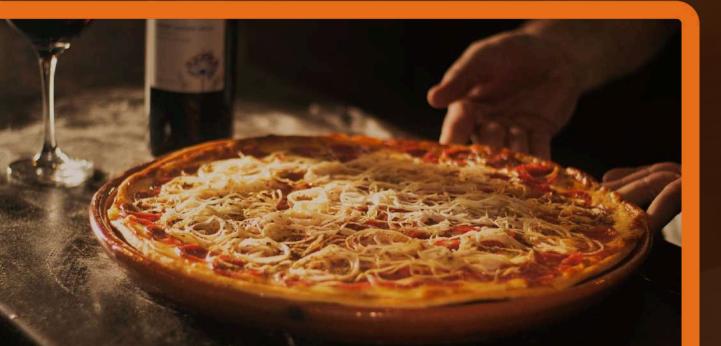


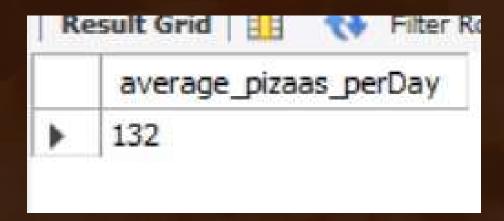
Re	esult Grid	Filter Ro
	category	count(name)
>	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY

```
SELECT
    ROUND(AVG(quantity), 0) as average_pizaas_perDay
FROM

(SELECT
    orders.order_date, SUM(order_details.quantity) AS quantity
FROM
    orders
JOIN order_details ON orders.order_Id = order_details.order_Id
GROUP BY orders.order_date) AS orders_quantity;
```







DETERMINE THE TOP 3 MOST PIZZA TYPES ORDERED BASED ON THE

REVENUE

```
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 Revenue DESC
LIMIT 3;
```





Kesuit Grid H		
	name	Revenue
>	The Thai Chicken Pizza	1296.5
	The Barbecue Chicken Pizza	1160.75
	The Italian Supreme Pizza	1158.75

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),
                       1) AS total_revenue
        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 Revenue DESC;
```



Result Grid		
	category	Revenue
>	Classic	26.44
	Supreme	25.99
	Veggie	24.83
	Chicken	22.74

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 pizzas.pizza_id = order_details.pizza_id

join orders

on orders.order_Id = order_details.order_Id group by orders.order_date) as sales;
```



Re	esult Grid	Filter Rows:
	order_date	Cum_Revenue
Þ	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

DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON THE 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 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, pizza_types.name
) as a) as b where rn<=3;</pre>
```





THE GLECK FIZZE	317.3
The Italian Supreme Pizza	1158.75
The Spicy Italian Pizza	1127.75
The Sicilian Pizza	954.75
The Five Cheese Pizza	980.5
The Four Cheese Pizza	947.60
The Mexicana Pizza	806.75

THANK YOU :: FOR ATTENTION

SQL PROJECT