

Where Every Slice is a Taste of Perfection

WELCOME TO SQL PIZZA SALES PROJECT

Start Your Slide

ORDER
NOW





OVERVIEW

I am Nikhil Jain, and I have developed this Pizza Sales project using SQL. The project involves analyzing sales data through structured SQL queries to derive key business insights such as total revenue, sales trends, top-selling products, and customer preferences. It showcases the practical application of SQL in real-world data analysis and decision-making.

QUESTIONS :


- 1- Retrieve 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.



1- Retrieve the total number of orders placed.



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

Result Grid		
	total_orders	
▶	21350	



2-Calculate the total revenue generated from pizza sales.



```
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;
```

Result Grid	
	total_sales
▶	817860.05



3- 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	

4- Identify the most common pizza size ordered.

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

Result Grid			Filter Rows:
	size	order_count	
▶	L	18526	

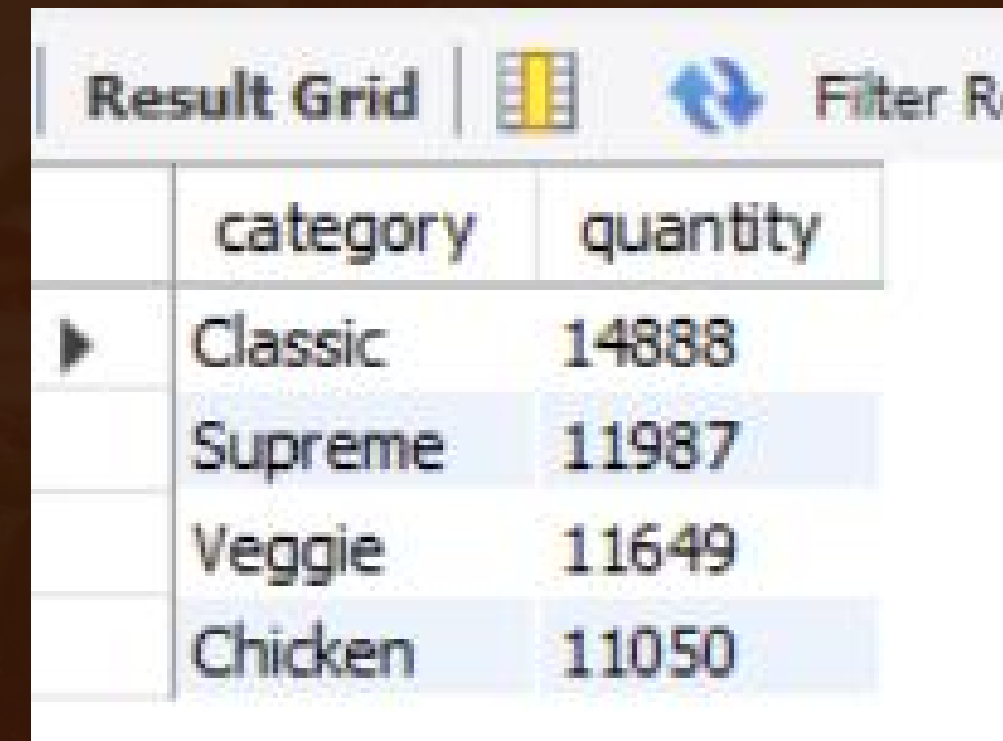
5-List the top 5 most ordered pizza types along with their quantities.

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

Result Grid			Filter Rows:
	name	total_pizz	
▶	The Classic Deluxe Pizza	2453	
	The Barbecue Chicken Pizza	2432	
	The Hawaiian Pizza	2422	
	The Pepperoni Pizza	2418	
	The Thai Chicken Pizza	2371	

6- 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 order_details.pizza_ID = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity DESC;
```



The screenshot shows a database interface with a 'Result Grid' tab. It displays the results of the SQL query, showing four rows of data. The first row is 'Classic' with a quantity of 14888. The second row is 'Supreme' with a quantity of 11987. The third row is 'Veggie' with a quantity of 11649. The fourth row is 'Chicken' with a quantity of 11050. The columns are labeled 'category' and 'quantity'.

	category	quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

7- Determine the distribution of orders by hour of the day.



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

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



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

```
SELECT
    category, COUNT(pizza_type_id) AS quantity
FROM
    pizza_types
GROUP BY category;
```

Result Grid			Filter
	category	quantity	
▶	Chicken	6	
	Classic	8	
	Supreme	9	
	Veggie	9	

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

```
SELECT
    ROUND(AVG(quantity), 0) AS avg_pizza_ordered_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 order_quantity;
```

Result Grid		Filter Rows:
	avg_pizza_ordered_perday	
▶	138	

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

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

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

11- 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),
            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;
```

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

12- 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 orders  
join order_details on orders.order_ID = order_details.order_ID  
join pizzas on pizzas.pizza_id = order_details.pizza_ID  
group by orders.order_date) as sales;
```

Result Grid				 Filter Rows:	
	order_date	cum_revenue			
	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	23990.35000000000002			



13- 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 rn
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 rn<=3;
```

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



CONCLUSION

THIS PROJECT PRESENTS AN IN-DEPTH ANALYTICAL EXPLORATION OF PIZZA SALES DATA, WHEREIN I METICULOUSLY ADDRESSED A SERIES OF SQL QUERIES THROUGH THE ADEPT UTILIZATION OF ADVANCED SQL FUNCTIONS. EMPLOYING SOPHISTICATED TECHNIQUES SUCH AS DATA AGGREGATION, CONDITIONAL FILTERING, RELATIONAL JOINS, AND HIERARCHICAL SORTING, THE ANALYSIS EXTRACTS ACTIONABLE BUSINESS INTELLIGENCE—ENCOMPASSING REVENUE TRAJECTORIES, HIGH-PERFORMING PRODUCT CATEGORIES, AND CONSUMER BEHAVIOR PATTERNS. THIS WORK UNDERSCORES THE PRAGMATIC APPLICATION OF SQL IN STRATEGIC DATA INTERPRETATION AND PERFORMANCE OPTIMIZATION.

