



SQL PROJECT ON PIZZA SALES DATASET



INTRODUCTION

In this project, I have worked on writing complex queries with subqueries and leveraging window functions for advanced analytics such as ranking, and also have applied various aggregate functions (SUM, AVG, COUNT) to generate critical business insights and improve decision-making. In a recent project, I analyzed a comprehensive pizza sales dataset to identify the most commonly ordered pizza types, revenue trends, and other key performance metrics. Additionally, I've optimized query performance, efficiently handled large datasets, and generated detailed reports to drive strategic business intelligence and support data-driven decision-making.

```
1      -- Retrieve the total number of orders placed.  
2  ●    SELECT  
3          COUNT(order_id)  
4  FROM  
5      orders;
```

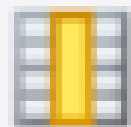
Result Grid



	COUNT(order_id)
▶	21350

```
1      -- Calculate the total revenue generated from pizza sales.
2  ●  SELECT
3      SUM(details.quantity * pizzas.price) AS revenue
4  FROM
5      details
6      JOIN
7      pizzas ON details.pizza_id = pizzas.pizza_id;
```

Result Grid

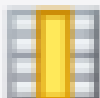



Filter

	revenue
▶	817860.0499999993

```
1  -- Identify the highest-priced pizza.
2  ● SELECT
3      pizza_types.name, pizzas.price
4  FROM
5      pizza_types
6      JOIN
7      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
8  GROUP BY name
9  ORDER BY price DESC
10 LIMIT 1;
11
```

Result Grid

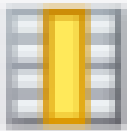






Filter Rows:

	name	price
▶	The Brie Carre Pizza	23.65



```
1      -- Identify the most common pizza size ordered.
2  ●    select size from
3  ⊖    (SELECT
4        size, sum(details.quantity)
5      FROM
6        details
7        JOIN
8        pizzas ON details.pizza_id = pizzas.pizza_id
9        group by size
10       order by quantity desc limit 1) as a;
```

Result Grid		
	size	
▶	M	



```
1      -- List the top 5 most ordered pizza types along with their quantities.
2  ●    SELECT
3          name, SUM(quantity) AS quantities
4  FROM
5      pizza_types
6      JOIN
7      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
8      JOIN
9      details ON details.pizza_id = pizzas.pizza_id
10 GROUP BY name
11 ORDER BY quantities DESC
12 LIMIT 5;
```

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

```
1  -- Join the necessary tables to find the total quantity of each pizza category ordered.
2  • SELECT
3      category, SUM(quantity) AS quantities
4  FROM
5      pizza_types
6      JOIN
7      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
8      JOIN
9      details ON details.pizza_id = pizzas.pizza_id
10 GROUP BY category
11 ORDER BY quantities;
```

Result Grid				 Filter Rows
	category	quantities		
▶	Chicken	11050		
	Veggie	11649		
	Supreme	11987		
	Classic	14888		


```
1  -- Determine the distribution of orders by hour of the day.
2  ●  SELECT
3      HOUR(orders.ordertime) AS hours,
4      COUNT(order_id) AS total_orders
5  FROM
6      orders
7  GROUP BY hours
8  ORDER BY hours ASC;
```

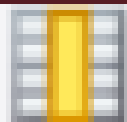
Result Grid				 Filter Rows
	hours	total_orders		
▶	9	1		
	10	8		
	11	1231		
	12	2520		
	13	2455		
	14	1472		
	15	1468		
	16	1920		

```
1  -- Join relevant tables to find the category-wise distribution of pizzas.
2  •  SELECT
3      category, COUNT(name) AS distribution
4  FROM
5      pizza_types
6  GROUP BY category
7  ORDER BY category;
8  |
```

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



```
1  -- Group the orders by date and calculate the average number of pizzas ordered per day.
2  ●  SELECT
3      AVG(quantity) AS average
4  FROM
5  ⊖  (SELECT
6      orders.date, SUM(details.quantity) AS quantity
7      FROM
8      details
9      JOIN orders ON details.order_id = orders.order_id
10     GROUP BY date) AS a;
```

Result Grid



	average
▶	138.4749

```
1  -- Determine the top 3 most ordered pizza types based on revenue.
2  ●  SELECT
3      name, revenue
4  FROM
5      (SELECT
6          pizza_types.name,
7          SUM(details.quantity * pizzas.price) AS revenue
8      FROM
9          pizza_types
10     JOIN pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
11     JOIN details ON details.pizza_id = pizzas.pizza_id
12     GROUP BY name
13     ORDER BY revenue DESC
14     LIMIT 3) AS a
```

Result Grid   Filter Rows:

	name	revenue
▶	The Thai Chicken Pizza	43434.25
	The Barbecue Chicken Pizza	42768
	The California Chicken Pizza	41409.5

```

1  -- Calculate the percentage contribution of each pizza type to total revenue.
2  ●  SELECT
3      pizza_types.category,
4      ROUND(SUM(details.quantity * pizzas.price) / (SELECT
5          SUM(details.quantity * pizzas.price) AS revenue
6      FROM
7          details
8          JOIN
9          pizzas ON details.pizza_id = pizzas.pizza_id) * 100,
10      2) AS percentage_contribution
11  FROM
12      pizza_types
13      JOIN
14      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
15      JOIN
16      details ON details.pizza_id = pizzas.pizza_id
17  GROUP BY category;

```

Result Grid			Filter Rows:
	category	percentage_contribution	
▶	Classic	26.91	
	Veggie	23.68	
	Supreme	25.46	
	Chicken	23.96	

```
1  -- Analyze the cumulative revenue generated over time.
2  • select sum(revenue)
3      over(partition by ordertime order by ordertime) as cumulative_revenue from
4  (select orders.ordertime, sum(details.quantity*pizzas.price) as revenue
5   from orders join details
6   on orders.order_id=details.order_id
7   join pizzas
8   on details.pizza_id=pizzas.pizza_id) as a;
9
```

Result Grid |   Filter

	cumulative_revenue
▶	817860.0499999993

```

1  -- Determine the top 3 most ordered pizza types based on revenue for each pizza category.
2  • select category,name,revenue,details from
3  (select category,name, revenue,
4  rank() over(partition by category order by revenue desc ) as details from
5  (SELECT
6      category,pizza_types.name,
7      SUM(details.quantity * pizzas.price) AS revenue
8  FROM
9      pizza_types
10     JOIN pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
11     JOIN details ON details.pizza_id = pizzas.pizza_id
12     group by category,name
13     order by revenue desc)as a) as b
14     where details<=3;
15

```

Result Grid					Filter Rows:	Export:	Wrap Cell C
	category	name	revenue	details			
▶	Chicken	The Thai Chicken Pizza	43434.25	1			
	Chicken	The Barbecue Chicken Pizza	42764.2768	2			
	Chicken	The California Chicken Pizza	41409.5	3			
	Classic	The Classic Deluxe Pizza	38180.5	1			
	Classic	The Hawaiian Pizza	32273.25	2			
	Classic	The Pepperoni Pizza	30161.75	3			
	Supreme	The Spicy Italian Pizza	34831.25	1			
	Supreme	The Italian Supreme Pizza	33476.75	2			