BLUE CRUST | SALES INSIGHTS REPORT

Data-Driven Insights, One Slice at a Time

Project name	SQL Data Analysis for Pizza sales
Tools	MYSQL, Power BI, Canva
Presented by	Kartik Zingade | Data analyst, Internship Studio

PROJECT DESCRIPTION

Blue Crust Pizzeria, a popular chain known for its gourmet pizzas and diverse menu, serves hundreds of customers daily. In the F&B industry, understanding sales velocity, order trends, and revenue contribution is key to optimizing business strategy. This project dives into Blue Crust's transactional data to uncover insights that drive profitability and efficiency.

What's on the Menu? (A.K.A. Our SQL Investigation)

We start with fundamentals—total orders, revenue, and best-selling pizzas—then explore demand segmentation by size and category. Moving deeper, we analyze peak order hours, category-wise sales, and temporal trends.

Our insights fall into four key categories:

›Sales Performance – Tracking revenue, order volumes, and top-selling items.
›Time-Based Trends – Analyzing peak ordering hours and seasonal patterns.
›Category-Wise Analysis – Understanding demand across different pizza types and sizes.
›Revenue Contribution – Identifying high-value pizzas and overall sales distribution.

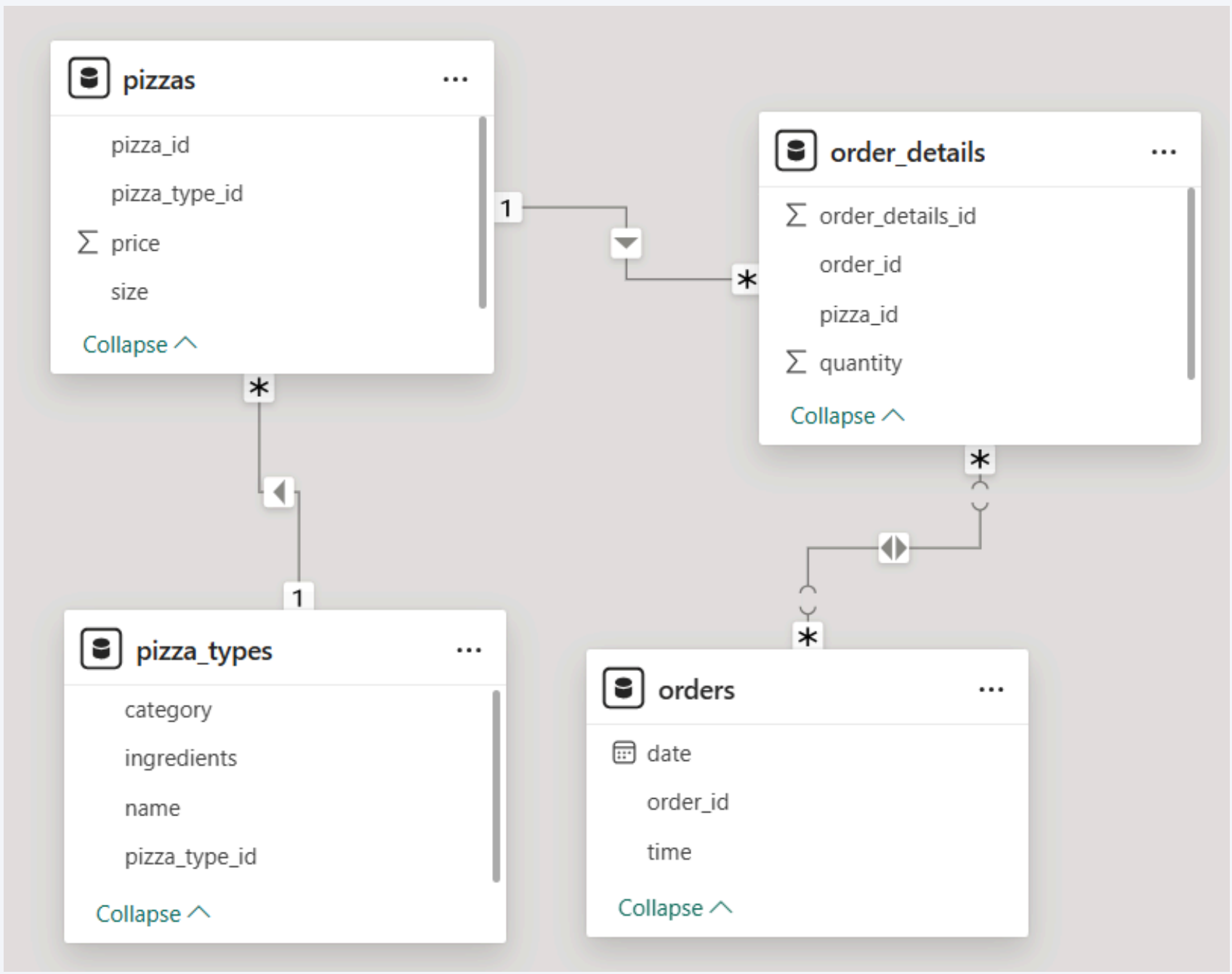
Finally, our advanced analytics focus on revenue attribution, cumulative sales forecasting, and product mix optimization—critical for any data-driven restaurant.

With SQL as our secret ingredient, let's slice through the data and uncover what truly drives pizza sales!

BLUE CRUST DATASET: WHAT WE'RE ANALYZING

Table Name	Key Columns
Orders	`order_id`, `date`, `time`
Order Details	`order_details_id`, `order_id`, `pizza_id`, `quantity`
Pizzas	`pizza_id`, `pizza_type_id`, `price`, `size`
Pizza Types	`pizza_type_id`, `name`, `category`, `ingredients`

SCHEMA



AD HOC BUSINESS INSIGHTS

--The total number of orders placed?

```
Select
count(distinct order_id)
as 'Total Orders'
from orders;
```

Results | Messages

	Total Orders
1	21338

--The total revenue generated from pizza sales.

```
Select cast(sum(order_details.quantity *
pizzas.price) as decimal(10,2))
as 'Total Revenue'
from order_details join pizzas
on
pizzas.pizza_id = order_details.pizza_id
```

Results | Messages

	Total Revenue
1	817860.05

--Identify the highest-priced pizza.

```
Select top 1 pizza_types.name
as 'Pizza Name',
cast(pizzas.price as decimal(10,2)) as
'Price'
from pizzas join pizza_types
on pizza_types.pizza_type_id =
pizzas.pizza_type_id
order by price desc
```

Results | Messages

	Pizza Name	Price
1	The Greek Pizza	35.95

-- Identify the most common pizza size ordered.

```
Select pizzas.size, count(distinct order_id)
as 'No of Orders', sum(quantity) as 'Total
Quantity Ordered'
from order_details join pizzas
on pizzas.pizza_id = order_details.pizza_id
-- join pizza_types on
pizza_types.pizza_type_id =
pizzas.pizza_type_id
group by pizzas.size
order by count(distinct order_id) desc
```

Results | Messages

	size	No of Orders	Total Quantity Ordered
1	L	12736	18956

--Calculate the average order value

```
Select
Sum(order_details.quantity*pizzas.price) /
count(distinct order_details.order_id)
as 'Average_order_value'
from order_details join pizzas on
order_details.pizza_id=pizzas.pizza_id
```

Results | Messages

	Average_order_value
1	38.3072623343546

--Identify the least-ordered pizza type.

```
Select top 1 pizzas.pizza_type_id,
count(distinct order_details.order_id)
as 'No of orders'
from pizzas join order_details on
pizzas.pizza_id=order_details.pizza_id
group by pizzas.pizza_type_id
order by
count (distinct order_details.order_id) asc
```

Results | Messages

	pizza_type_id	No of orders
1	brie_carre	480

KEY METRICS AND HIGHLIGHTS

TOTAL NUMBER OF ORDERS	TOTAL REVENUE FROM SALES	AVERAGE ORDER VALUE (AOV)
21338	817860.05	38.31

MOST COMMON PIZZA SIZE ORDERED	HIGHEST PRICED PIZZA	LEAST ORDERED PIZZA
Large (L)	The Greek Pizza	Brie Carre

SALES PERFORMANCE & REVENUE INSIGHTS

--Total quantity of pizzas sold by category

```
Select pizza_types.category,
sum(quantity)
as 'Total Quantity Ordered'
from order_details
join pizzas on
pizzas.pizza_id =
order_details.pizza_id
join pizza_types on
pizza_types.pizza_type_id =
pizzas.pizza_type_id
group by pizza_types.category
order by sum(quantity)  desc
```

Results | Messages

	category	Total Quantity Ordered
1	Classic	14888
2	Supreme	11987
3	Veggie	11649
4	Chicken	11050

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

```
Select top 5 pizza_types.name as
'Pizza', sum(quantity) as 'Total
Ordered'
from order_details
join pizzas on
pizzas.pizza_id =
order_details.pizza_id
join pizza_types on
pizza_types.pizza_type_id =
pizzas.pizza_type_id
group by pizza_types.name
order by sum(quantity) desc
```

Results | Messages

	Pizza	Total Ordered
1	The Classic Deluxe Pizza	2453
2	The Barbecue Chicken Pizza	2432
3	The Hawaiian Pizza	2422
4	The Pepperoni Pizza	2418
5	The Thai Chicken Pizza	2371

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

```
Select datepart(hour, time)
as 'Hour of the day',
count(distinct order_id)
as 'No of Orders'
from orders
group by datepart(hour, time)
order by [No of Orders] desc
```

Results | Messages

	Hour of the day	No of Orders
1	12	2520
2	13	2455
3	18	2399
4	17	2336
5	19	2009
6	16	1920
7	20	1642
8	14	1472
9	15	1468
10	11	1231
11	21	1198
12	22	663
13	23	28
14	10	8
15	9	1

-- Find the category-wise distribution of pizzas

```
Select category,
count(distinct pizza_type_id)
as [No of pizzas]
from pizza_types
group by category
order by [No of pizzas]
```

Results | Messages

	category	No of pizzas
1	Chicken	6
2	Classic	8
3	Supreme	9
4	Veggie	9

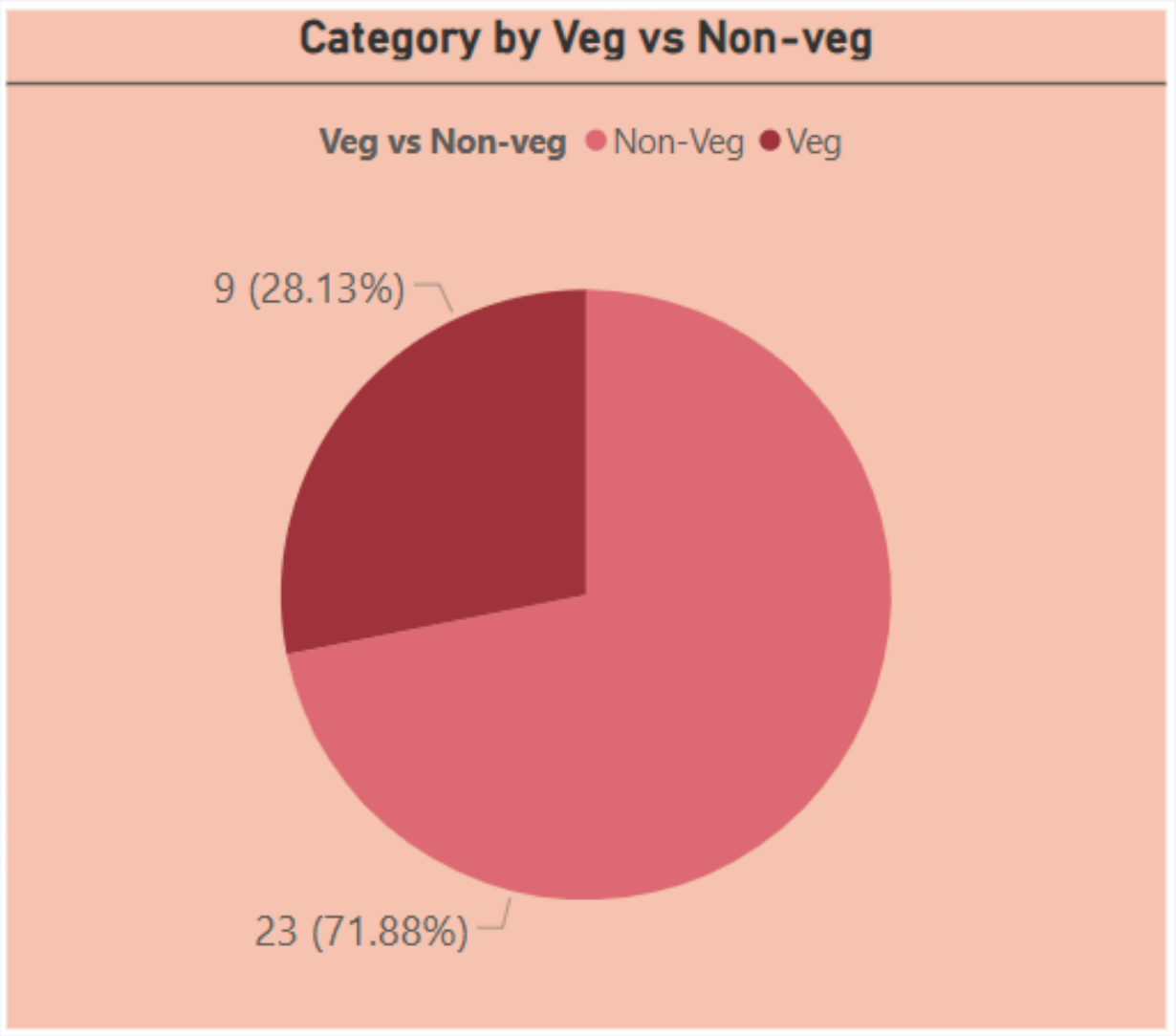
-- Determine the top 3 most ordered pizza types based on revenue.

```
Select top 3 pizza_types.name,
sum(order_details.quantity*pizzas.price) as
'Revenue from pizza'
from order_details
join pizzas on pizzas.pizza_id =
order_details.pizza_id
join pizza_types on pizza_types.pizza_type_id
= pizzas.pizza_type_id
group by pizza_types.name
order by [Revenue from pizza] desc
```

Results | Messages

	name	Revenue from pizza
1	The Thai Chicken Pizza	43434.25
2	The Barbecue Chicken Pizza	42768
3	The California Chicken Pizza	41409.5

MENU DISTRIBUTION OVER VEG BY NON-VEG



TOTAL ORDERS AND QUANTITY SOLD BY PIZZA SIZE

OPERATIONAL & ORDER TRENDS

-- Calculate the average number of pizzas ordered per day.

```
Select
avg([Total Pizza Ordered that day]) as
[Avg Number of pizzas ordered per
day]
from
--starting subquery
(Select orders.date as 'Date',
sum(order_details.quantity)
as 'Total Pizza Ordered that day'
from order_details
join orders on
order_details.order_id =
orders.order_id
group by orders.date
)
as pizzas_ordered
```

Results | Messages

	Avg Number of pizzas ordered per day
1	138

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

```
with cte as (
Select category, name,
cast(sum(quantity*price) as decimal(10,2)) as
Revenue
from order_details
join pizzas on
pizzas.pizza_id = order_details.pizza_id
join pizza_types on pizza_types.pizza_type_id
= pizzas.pizza_type_id
group by category, name
-- order by category, name, Revenue desc
)

, cte1 as (
Select category, name, Revenue,
rank() over (partition by category order by
Revenue desc) as rnk
from cte
)

Select category, name, Revenue
from cte1
where rnk in (1,2,3)
order by category, name, Revenue
```

Results | Messages

	category	name	Revenue
1	Chicken	The Barbecue Chicken Pizza	42768.00
2	Chicken	The California Chicken Pizza	41409.50
3	Chicken	The Thai Chicken Pizza	43434.25
4	Classic	The Classic Deluxe Pizza	38180.50
5	Classic	The Hawaiian Pizza	32273.25
6	Classic	The Pepperoni Pizza	30161.75
7	Supreme	The Italian Supreme Pizza	33476.75
8	Supreme	The Sicilian Pizza	30940.50
9	Supreme	The Spicy Italian Pizza	34831.25
10	Veggie	The Five Cheese Pizza	26066.50
11	Veggie	The Four Cheese Pizza	32265.70
12	Veggie	The Mexicana Pizza	26780.75

-- Calculate the percentage contribution of each pizza type to total revenues

```
Select pizza_types.name,
concat(cast((sum
(order_details.quantity*pizzas.price) /
(select
sum(order_details.quantity*pizzas.price)
from order_details
join pizzas on
pizzas.pizza_id = order_details.pizza_id
))*100
as decimal(10,2)), '%')
as 'Revenue contribution from pizza'
from order_details
join pizzas on
pizzas.pizza_id = order_details.pizza_id
join pizza_types on
pizza_types.pizza_type_id =
pizzas.pizza_type_id
group by pizza_types.name
order by [Revenue contribution from
pizza] desc
```

Results | Messages

	name	Revenue contribution from pizza
1	The Thai Chicken Pizza	5.31%
2	The Barbecue Chicken Pizza	5.23%
3	The California Chicken Pizza	5.06%
4	The Classic Deluxe Pizza	4.67%
5	The Spicy Italian Pizza	4.26%
6	The Southwest Chicken Pizza	4.24%
7	The Italian Supreme Pizza	4.09%
8	The Four Cheese Pizza	3.95%
9	The Hawaiian Pizza	3.95%
10	The Sicilian Pizza	3.78%
11	The Pepperoni Pizza	3.69%
12	The Greek Pizza	3.48%
13	The Mexicana Pizza	3.27%
14	The Five Cheese Pizza	3.19%
15	The Pepper Salami Pizza	3.12%
16	The Italian Capocollo Pizza	3.07%
17	The Vegetables + Vegetables Pizza	2.98%
18	The Prosciutto and Arugula Pizza	2.96%
19	The Napolitana Pizza	2.95%
20	The Spinach and Feta Pizza	2.85%
21	The Big Meat Pizza	2.81%
22	The Pepperoni, Mushroom, and P...	2.30%
23	The Chicken Alfredo Pizza	2.07%
24	The Chicken Pesto Pizza	2.04%
25	The Soppressata Pizza	2.01%
26	The Italian Vegetables Pizza	1.96%
27	The Calabrese Pizza	1.95%
28	The Spinach Pesto Pizza	1.91%
29	The Mediterranean Pizza	1.88%
30	The Spinach Supreme Pizza	1.87%
31	The Green Garden Pizza	1.71%
32	The Brie Carre Pizza	1.42%

-- Analyze the cumulative revenue generated over time.

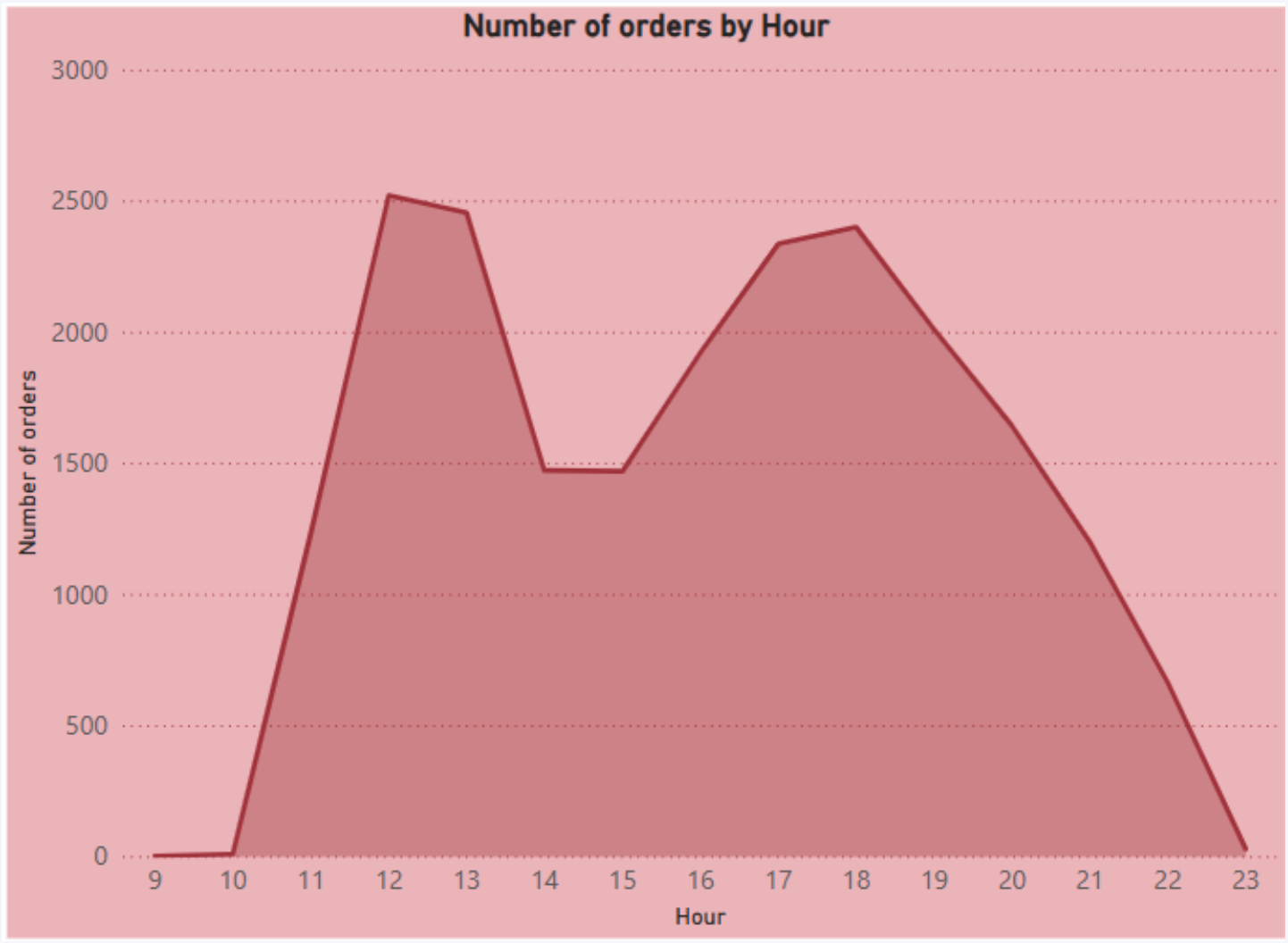
```
with cte as (
Select date as 'Date',
cast(sum(quantity*price) as
decimal(10,2))
as Revenue
from order_details
join orders on
order_details.order_id = orders.order_id
join pizzas on
pizzas.pizza_id = order_details.pizza_id
group by date
-- order by [Revenue] desc
)

Select Date, Revenue, sum(Revenue)
over (order by date)
as 'Cumulative Sum'
from cte
group by date, Revenue
```

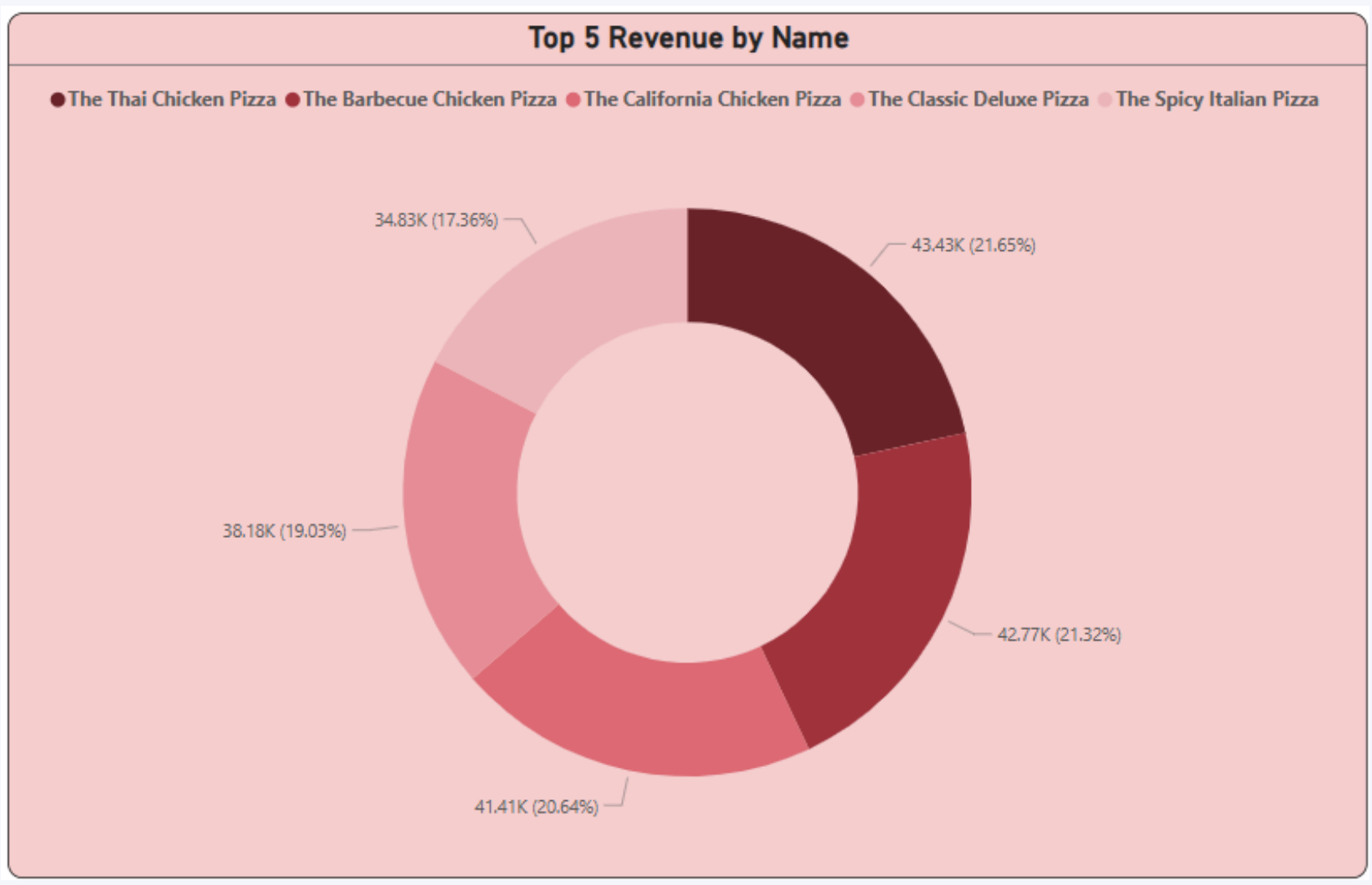
Results | Messages

	Date	Revenue	Cumulative Sum
1	2023-01-01	2713.85	2713.85
2	2023-01-02	2731.90	5445.75
3	2023-01-03	2662.40	8108.15
4	2023-01-04	1755.45	9863.60
5	2023-01-05	2065.95	11929.55
6	2023-01-06	2428.95	14358.50
7	2023-01-07	2202.20	16560.70
8	2023-01-08	2838.35	19399.05
9	2023-01-09	2127.35	21526.40
10	2023-01-10	2463.95	23990.35
11	2023-01-11	1872.30	25862.65
12	2023-01-12	1919.05	27781.70
13	2023-01-13	2049.60	29831.30
14	2023-01-14	2527.40	32358.70
15	2023-01-15	1984.80	34343.50
16	2023-01-16	2594.15	36937.65
17	2023-01-17	2064.10	39001.75
18	2023-01-18	1976.85	40978.60
19	2023-01-19	2387.15	43365.75
20	2023-01-20	2397.90	45763.65
21	2023-01-21	2040.55	47804.20
22	2023-01-22	2496.70	50300.90
23	2023-01-23	2423.70	52724.60
24	2023-01-24	2289.25	55013.85
25	2023-01-25	1617.55	56631.40
26	2023-01-26	1884.40	58515.80
27	2023-01-27	2528.05	61043.85
28	2023-01-28	2016.00	63059.85
29	2023-01-29	2045.30	65105.15
30	2023-01-30	2270.30	67375.45

HOURLY TREND OF THE ORDERS



REVENUE DISTRIBUTION BY TOP 5 PIZZAS SOLD



BLUE CRUST | SALES INSIGHTS REPORT

Key Insights and Findings

- The Average Order Value (**AOV**) across all sales stands at **$38.31**, reflecting the typical spend per transaction.
- The most commonly ordered pizza size is **Large (L)**, followed by Medium (M) and Small (S), indicating customer preference trends.
- All of the top five most-ordered pizzas are non-vegetarian, with **'The Classic Deluxe Pizza'** emerging as the best-selling menu item.
- Order volume peaks during lunchtime **(12:00 - 15:00)**, followed by a secondary spike during dinner hours **(18:00 - 20:00)**, highlighting key service periods.
- The average daily order volume is approximately **138**, providing insights into baseline demand patterns.

Recommendations for The Blue Crust

- Given that XL and XXL sizes have the lowest order volume, inventory optimization strategies should be implemented by maintaining a lean stock of these bases to minimize holding costs.
- As The Brie Carre ranks as the least ordered pizza, we can enhance its visibility through strategic promotions, such as bundling it in value meal combos or applying targeted discounting to drive sales.
- With peak service hours identified, we can improve operational efficiency by allocating non-peak hours for back-of-house tasks like inventory reconciliation, data entry, and preemptive cleaning to streamline workflow.

KARTIK ZINGADE
DATA ANALYST
zingadekartik@gmail.com github.com/kartikkwearsmask