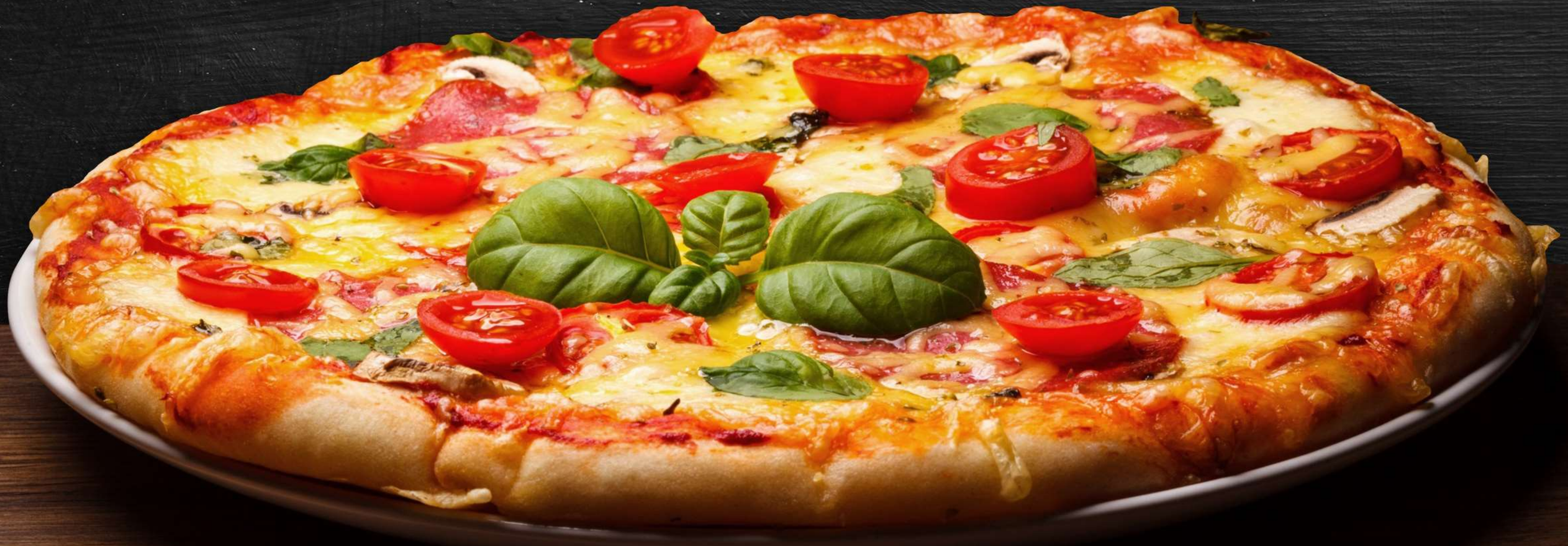
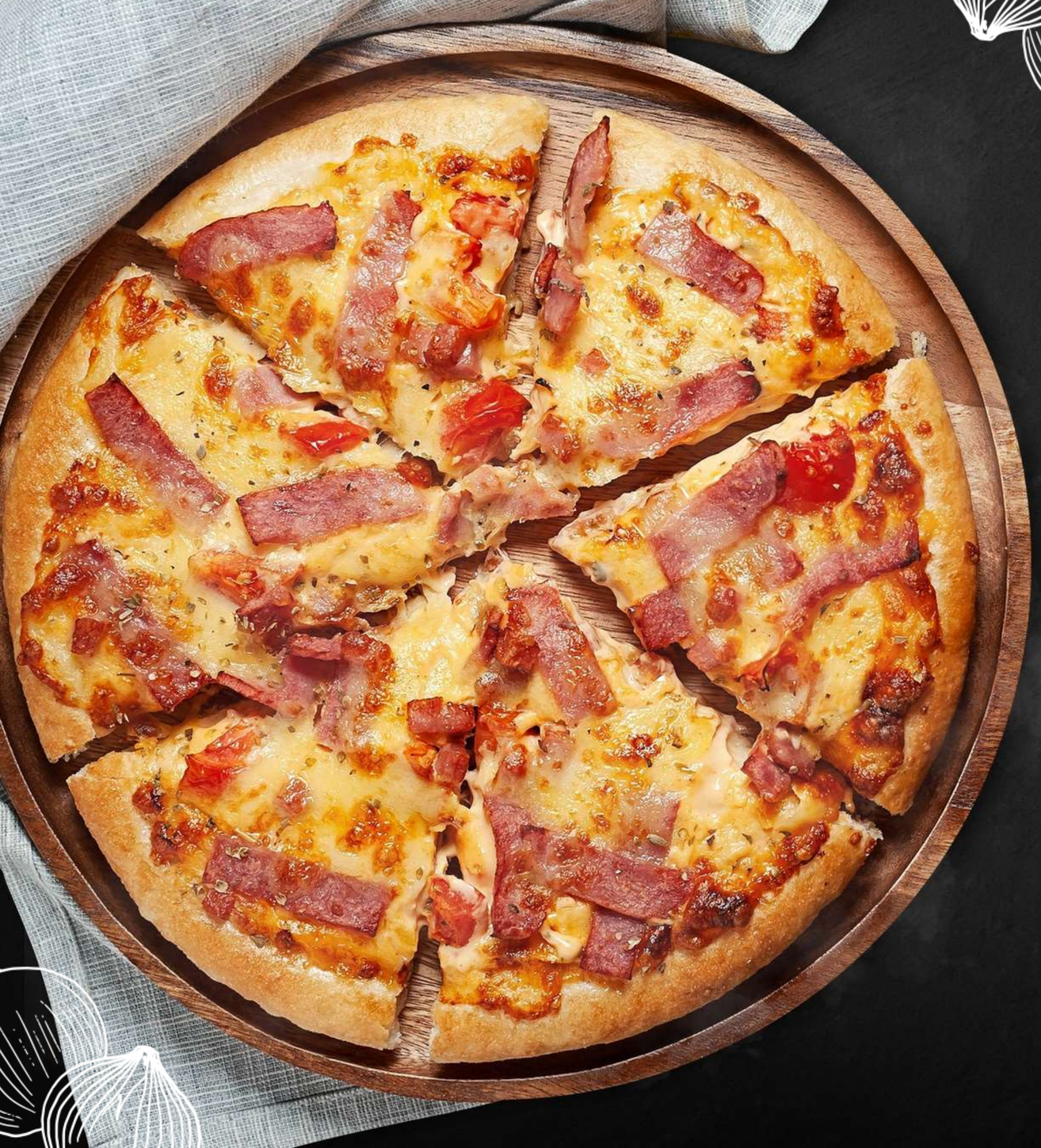
DATACRUST PIZZA

MYSQL-DRIVEN PIZZA SALES ANALYTICS: A DATA ANALYSIS PROJECT



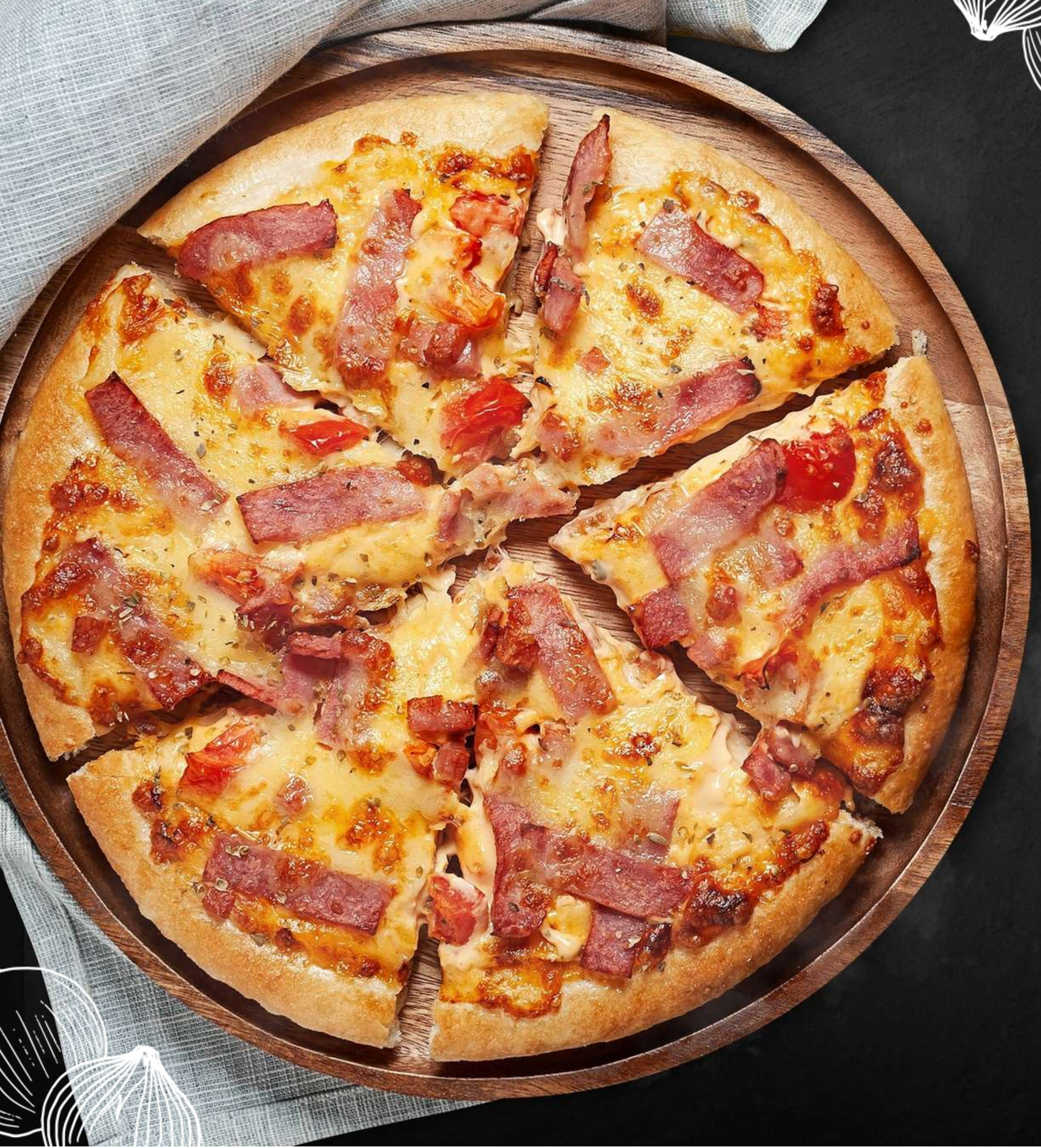


PROBLEM STATEMENT

In the competitive food industry, businesses face challenges in understanding customer preferences, optimizing product offerings, and maximizing revenue. Pizza sales, being a popular segment, generate vast amounts of data that remain underutilized. The lack of actionable insights from this data often leads to missed opportunities for growth, inefficient resource allocation, and suboptimal decision-making.

PROJECT DESCRIPTION

This project is a comprehensive data analysis system built using MySQL, designed to derive valuable insights from pizza sales data. The system analyzes various aspects of sales, including total orders, revenue generation, product popularity, customer preferences, and order distribution across time. It addresses essential business questions, such as identifying the most popular pizza types, determining peak sales hours, and calculating the revenue contribution of individual pizzas. The project leverages SQL queries, joins, and advanced analytical techniques to provide actionable insights for better decision-making.



VISSION , MISSION & PURPOSE

VISSION

To empower businesses in the food industry with data-driven insights that unlock customer preferences, optimize product offerings, and drive sustainable growth.

MISSION

To provide a user-friendly and insightful data analysis platform for pizza sales, enabling businesses to make informed decisions by leveraging advanced SQL techniques to uncover trends, maximize revenue, and enhance customer satisfaction.

PURPOSE

I developed this project to showcase my data analysis and SQL skills while addressing real-world business challenges. It demonstrates my ability to handle large datasets, extract meaningful insights, and provide solutions that drive business growth. This project reflects my analytical mindset and my capacity to contribute to data-driven decision-making in any organization. It also serves as a testament to my readiness for roles requiring strong SQL and data interpretation skills.

DATACRUST PIZZA

Set 1: "Revenue and Growth-Centric Insights“

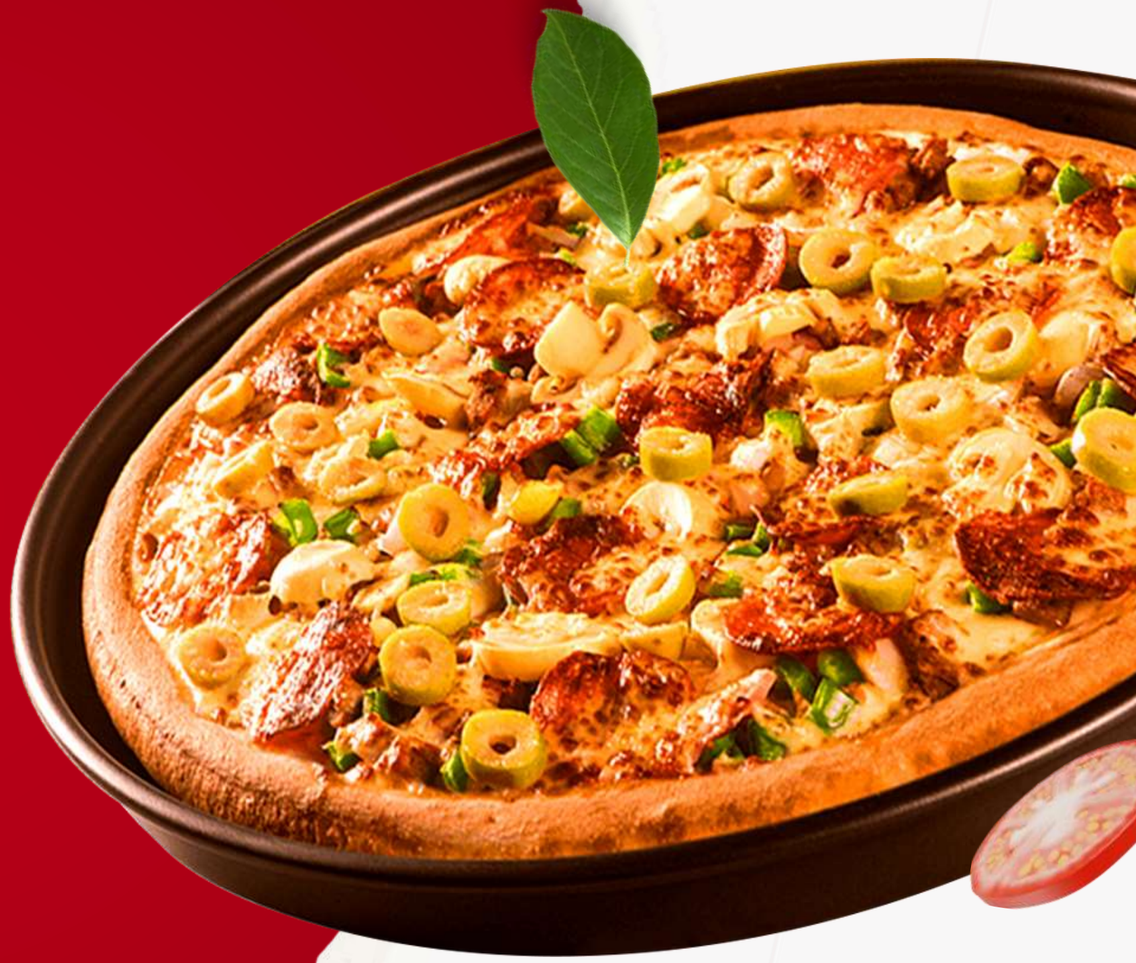
includes questions directly tied to revenue and business growth, critical for data-driven decision-making.

Set 2: "Foundational Analysis of Customer Behavior and Operations“

focuses on foundational analysis and insights related to customer behavior and operations.

Set 3: "Advanced and Granular Analytical Skills“

demonstrates advanced or granular skills, valuable but less impactful in immediate business terms.



DATACRUST PIZZA

SET 1 QUESTIONS

1.Calculate the total revenue generated from pizza sales.

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

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

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

5. Analyze the cumulative revenue generated over time.

SET 2 QUESTIONS

1.Retrieve the total number of orders placed..

2. Identify the highest-priced pizza.

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

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

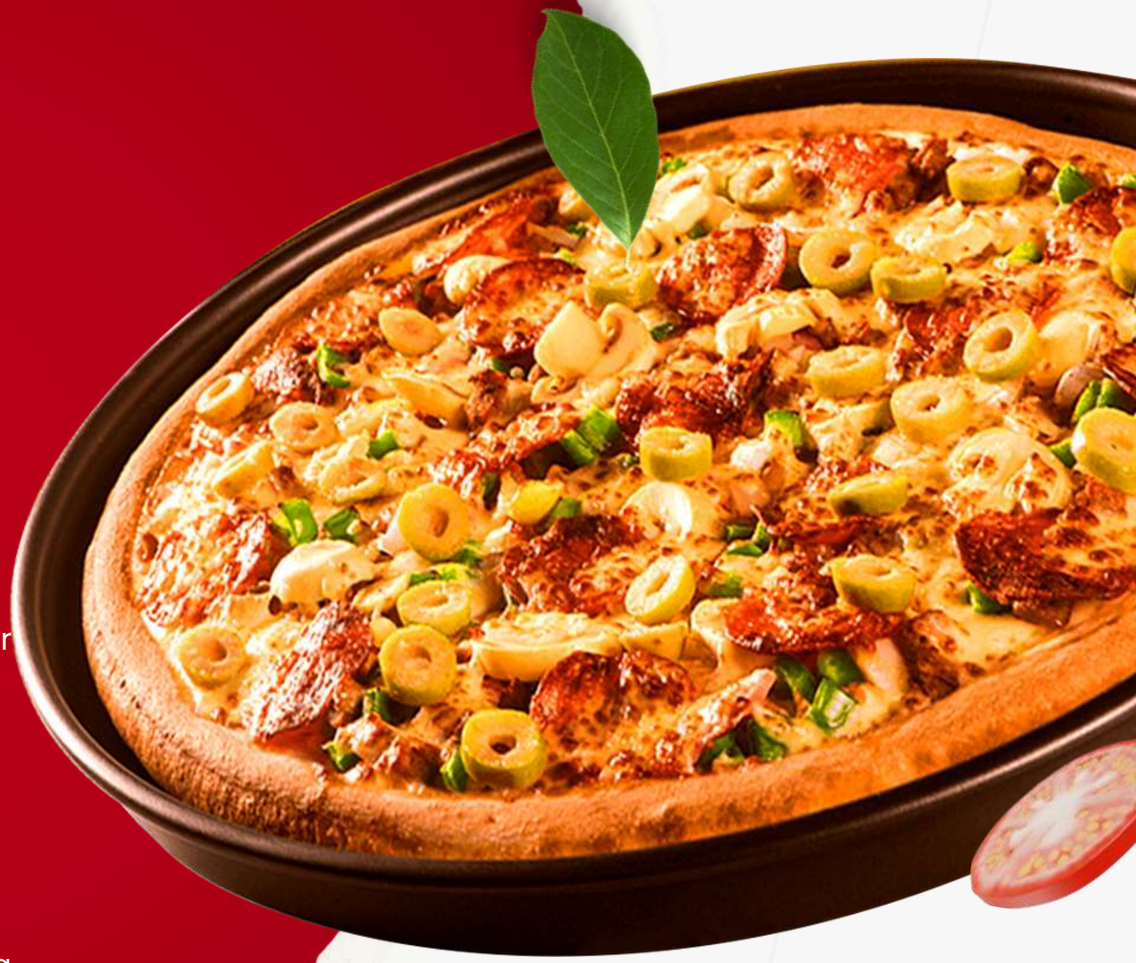
5. Group the orders by date and calculate the average number of pizzas ordered per day.

SET 3 QUESTIONS

1.Identify the most common pizza size ordered.

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

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



SET 1

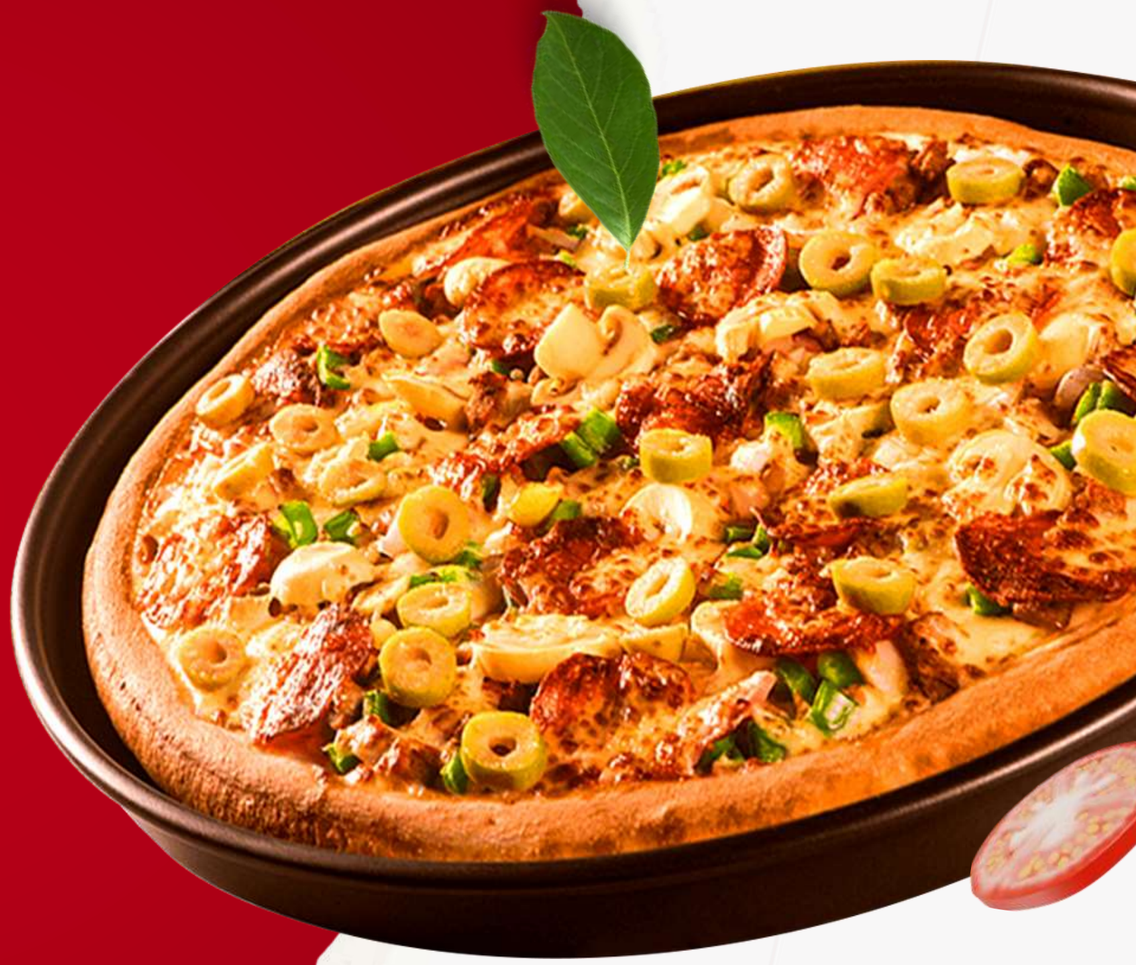
(includes questions directly tied to revenue and business growth, critical for data-driven decision-making)

1:Calculate the total revenue generated from pizza sales

```sql
-- First query to calculate total sales
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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_sales
817860.05



SET 1

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

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

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

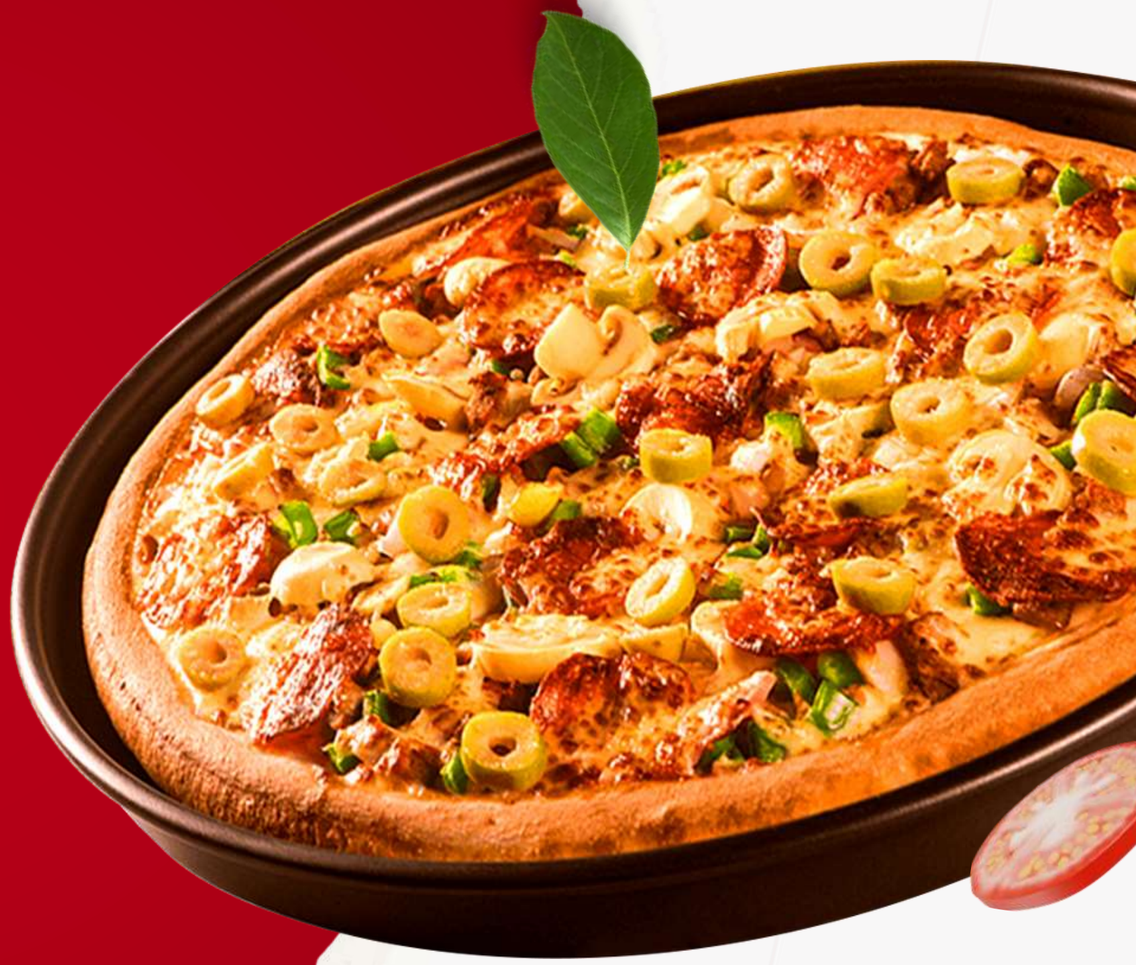
name	quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

Result 4

Output

Action Output

#	Time	Action	Message
1	14:20:46	SELECT pizza_types.name, SUM(orders_details.quantity) as quantity FROM pizza_types JOIN pizzas ON...	5 row(s) returned



SET 1

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

```sql
-- Determine the top 3 most ordered pizza types based on 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(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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

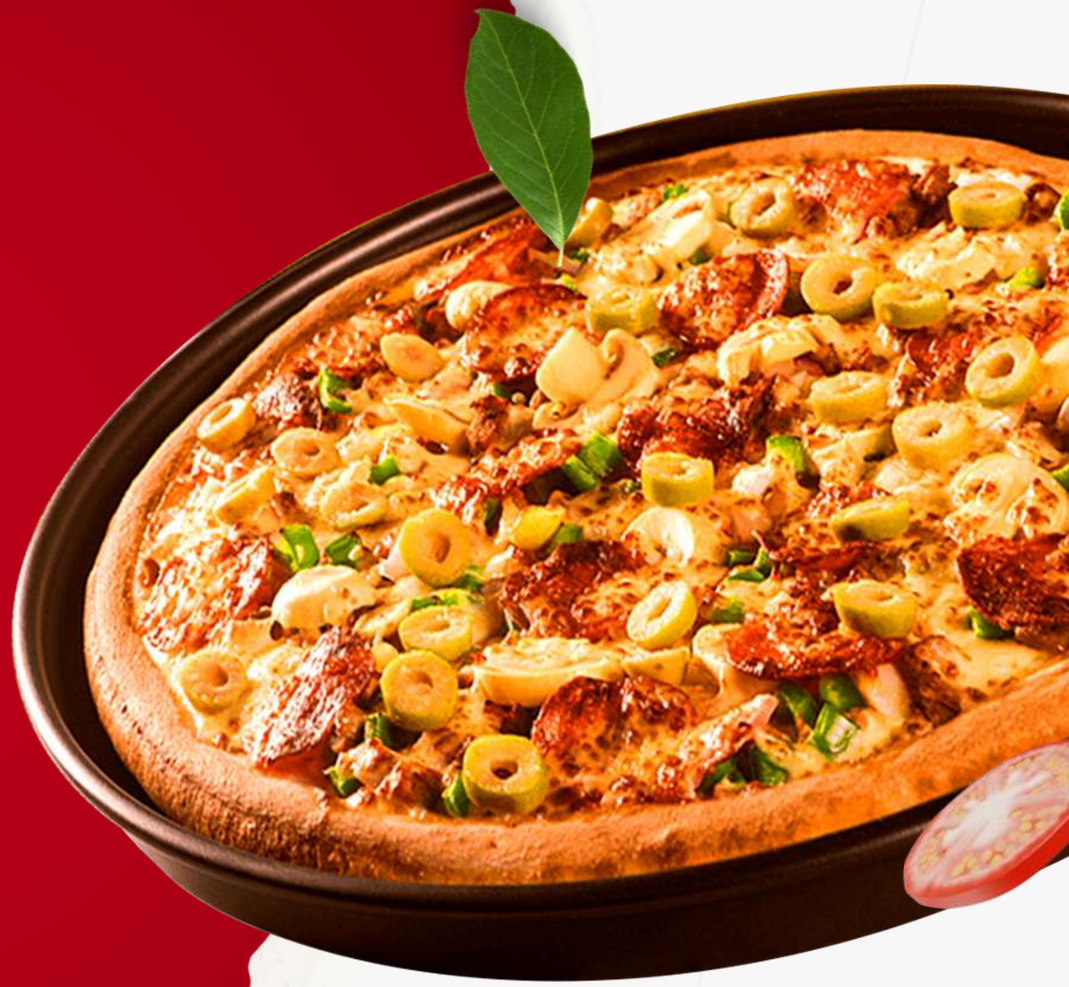
name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
The Classic Deluxe Pizza	38180.5
The Hawaiian Pizza	32273.25
The Pepperoni Pizza	30161.75
The Spicy Italian Pizza	34831.25
The Italian Supreme Pizza	33476.75
The Sicilian Pizza	30940.5
The Four Cheese Pizza	32265.70000000065
The Mexicana Pizza	26780.75
The Five Cheese Pizza	26066.5

Result 6 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	15:14:32	select name,revenue from (select category,name,revenue, rank() over(partition by category order by revenue d...	12 row(s) returned	0.203 sec / 0.000 sec



SET 1

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

```sql
-- (SET1)Calculate the percentage contribution of each pizza type to total revenue.
SELECT
    pizza_types.category,
    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 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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

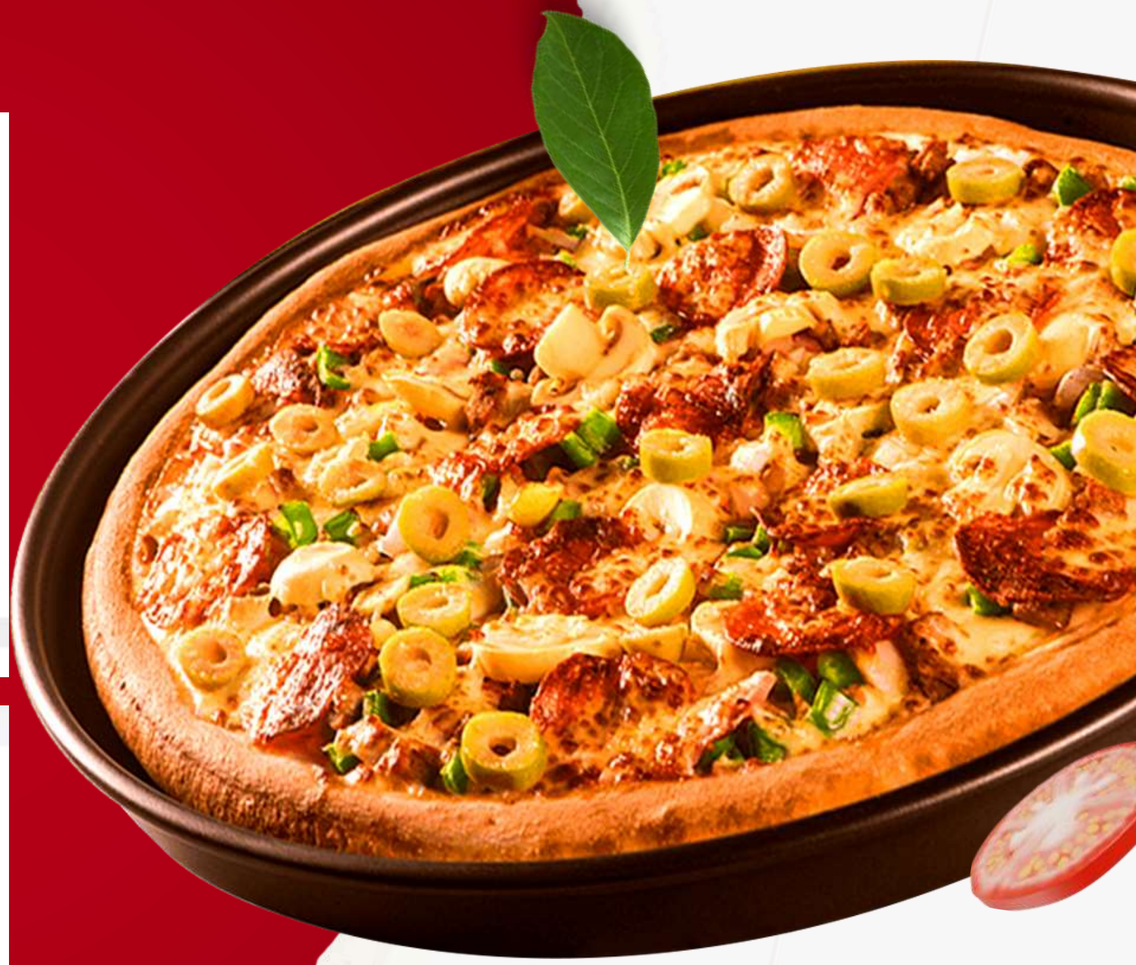
category	revenue
Classic	26.90596025566967
Supreme	25.45631126009862
Chicken	23.955137556847287
Veggie	23.682590927384577

Result 8

Output

Action Output

#	Time	Action		Message
1	15:20:20	SELECT	pizza_types.category, SUM(orders_details.QUANTITY * pizzas.price) / (SELECT ROU...	4 row(s) returned



SET 1

5:Analyze the cumulative revenue generated over time.

```sql
-- Analyze the cumulative revenue generated over time.
SELECT
    ORDER_DATE,
    SUM(revenue) OVER (ORDER BY ORDER_DATE) AS cum_rev
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 daily_revenue;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

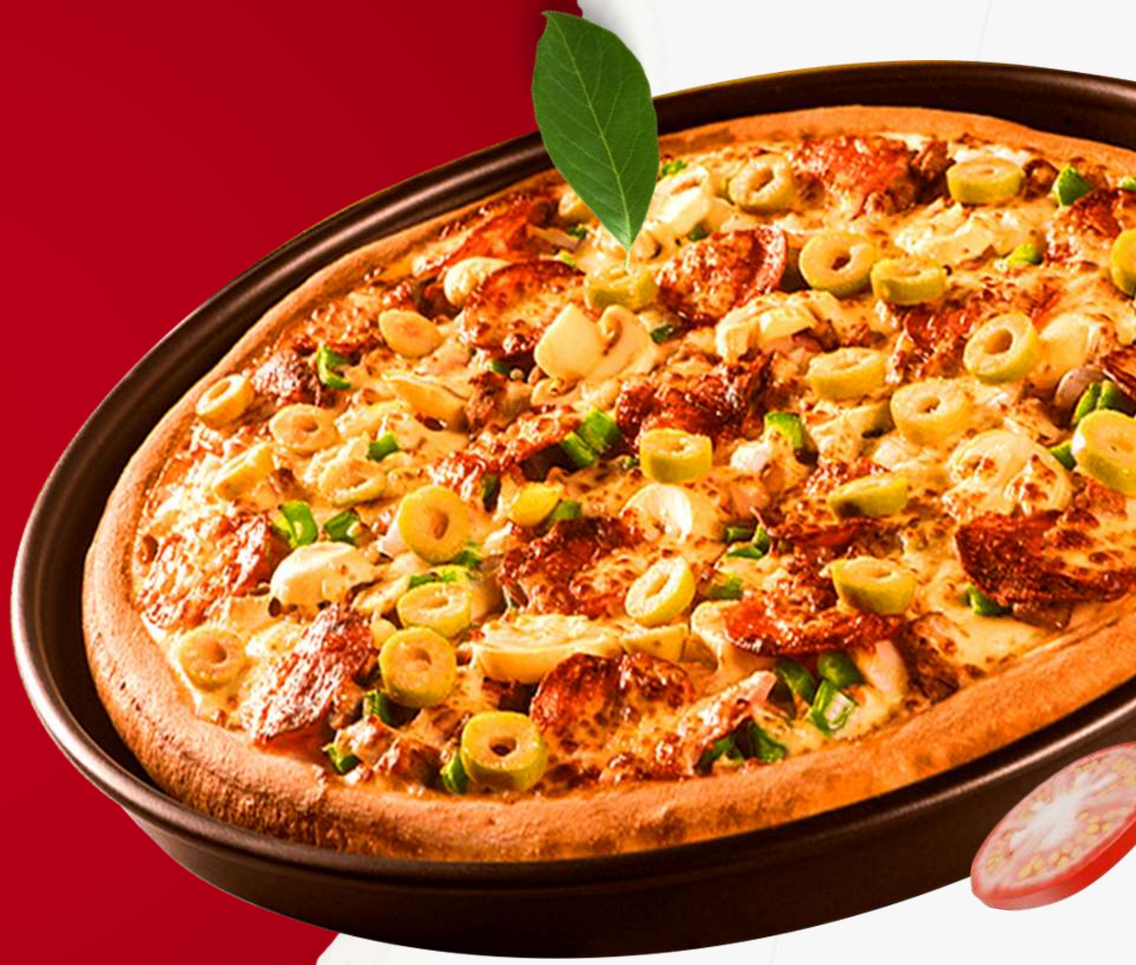
ORDER_DATE	cumulative_revenue
2015-01-01	2713.8500000000004
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55
2015-01-06	14358.5
2015-01-07	16560.7
2015-01-08	19399.05
2015-01-09	21526.4
2015-01-10	23990.350000000002
2015-01-11	25862.65
2015-01-12	27781.7
2015-01-13	29831.300000000003
2015-01-14	32358.700000000004

Result 15 x Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	15:25:32	SELECT ORDER_DATE, SUM(revenue) OVER (ORDER BY ORDER_DATE) AS cum_rev FROM (S...	358 row(s) returned	0.171 sec / 0.000 sec
2	15:26:15	SELECT ORDER_DATE, SUM(revenue) OVER (ORDER BY ORDER_DATE) AS cumulative_revenue F...	358 row(s) returned	0.171 sec / 0.000 sec



SET 2

(focuses on foundational analysis and insights related to customer behavior and operations)

1:Retrieve the total number of orders placed.

```
-- (SET 2)Retrieve the total number of orders placed.

SELECT count(ORDER_ID) AS TOTAL_ORDER FROM ORDERS;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

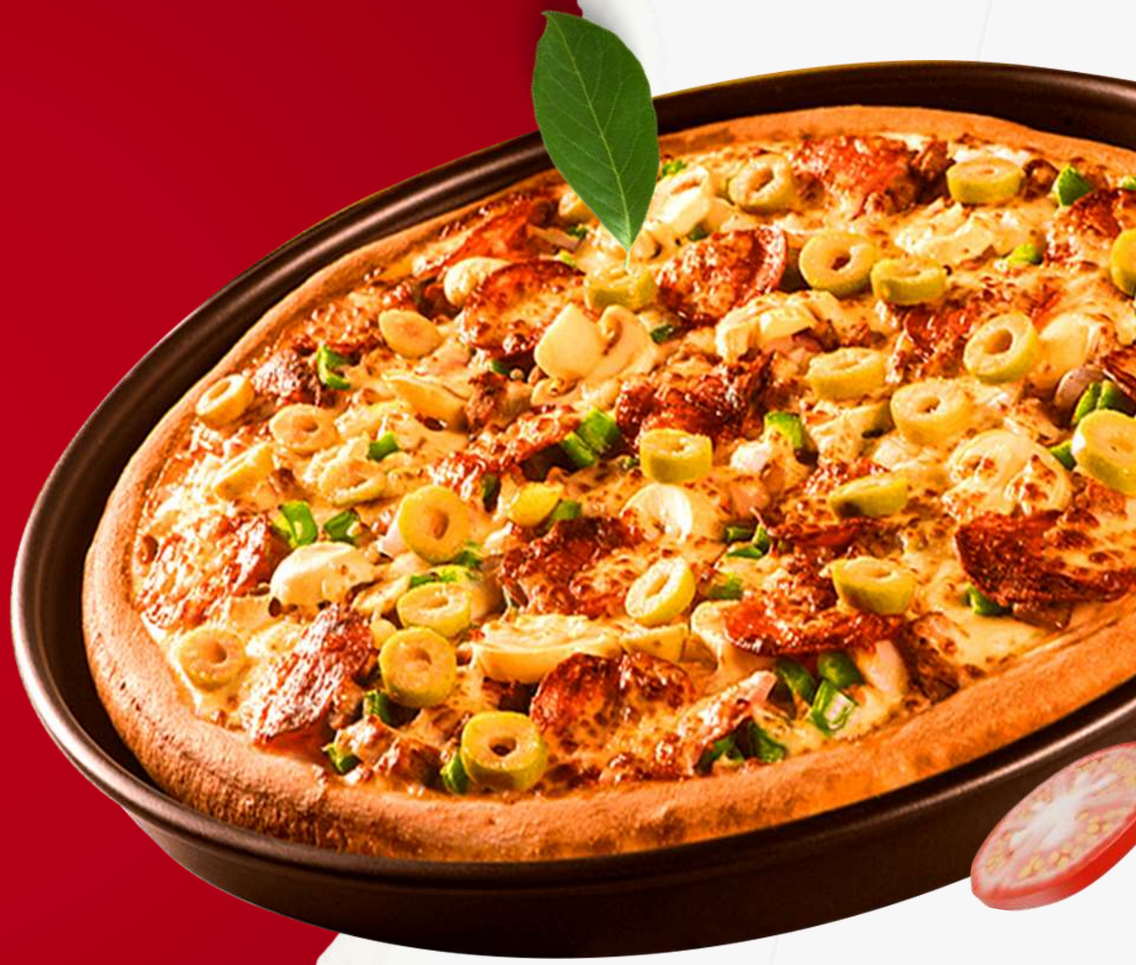
TOTAL_ORDER
21350

Result 8

Output

Action Output

#	Time	Action	Message
1	15:40:14	SELECT count(ORDER_ID) AS TOTAL_ORDER FROM ORDERS LIMIT 0, 200	1 row(s) returned



SET 2

2:Identify the highest-priced pizza.

```
-- query to find the highest-priced pizza
SELECT MAX(pizzas.price) AS highest_price
FROM pizzas;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

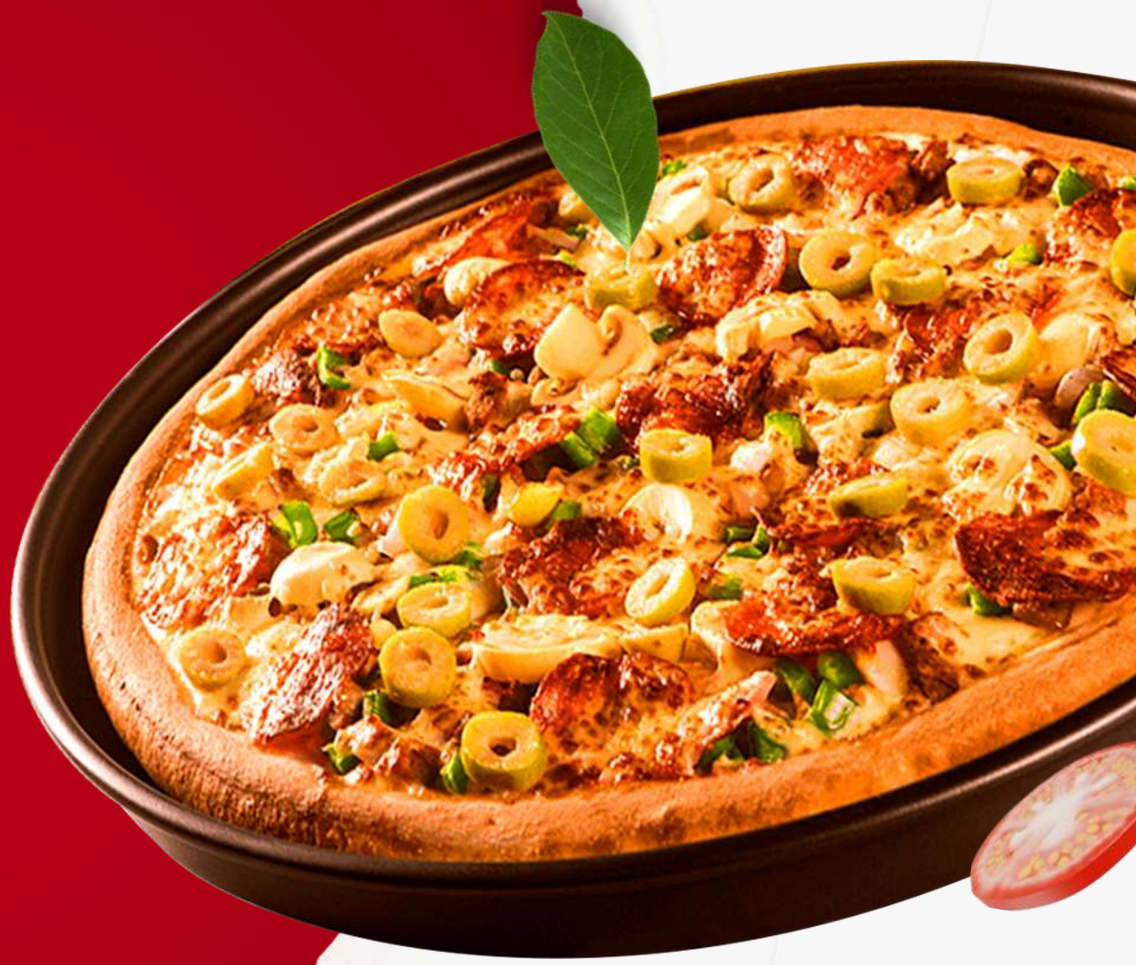
highest_price
35.95

Result 9

Output

Action Output

#	Time	Action	Message
1	15:43:31	SELECT MAX(pizzas.price) AS highest_price FROM pizzas LIMIT 0, 200	1 row(s) returned



SET 2

3:Determine the distribution of orders by hour of the day.

```sql
-- Determine the distribution of orders by hour of the day.
SELECT
    HOUR(order_time) AS hour, COUNT(Order_id) AS id
FROM
    orders
GROUP BY HOUR(order_time);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

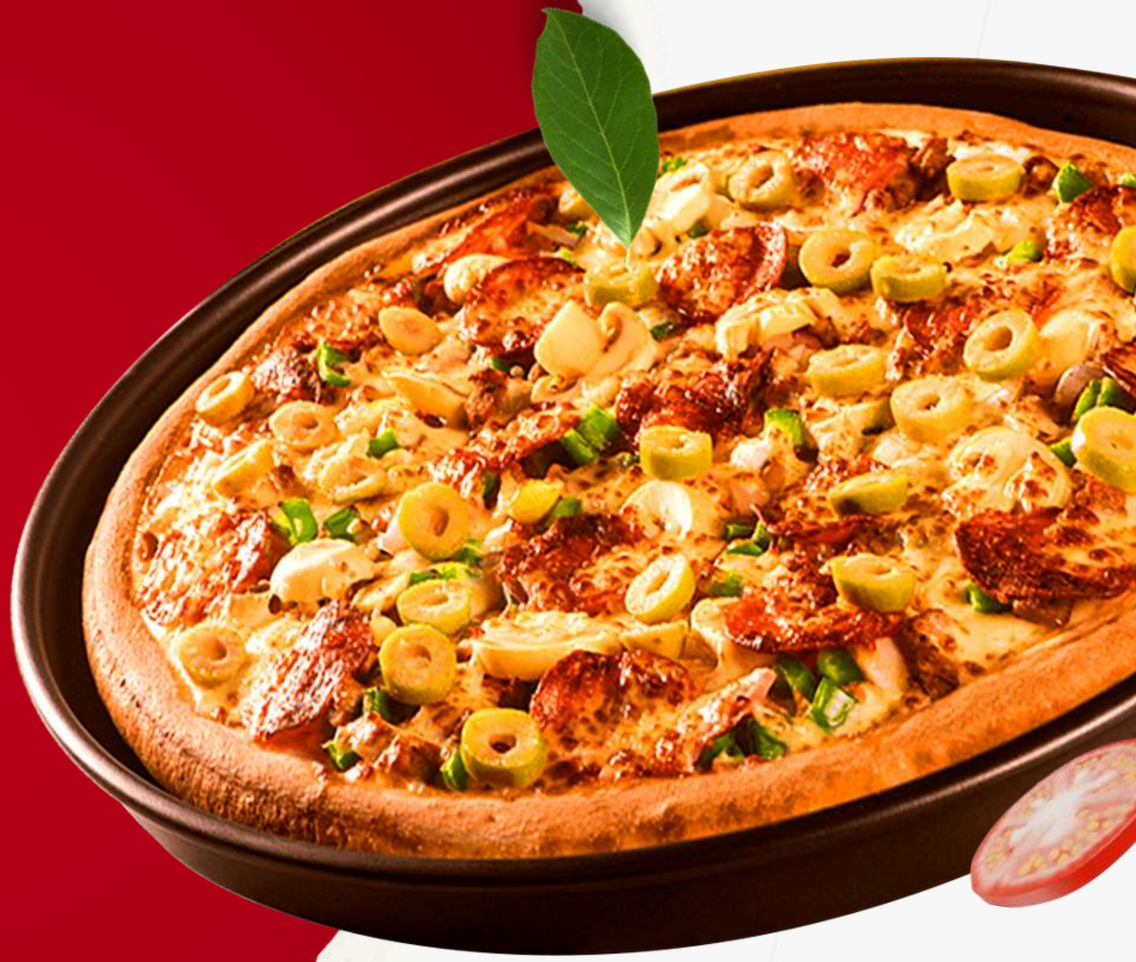
hour	id
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009
20	1642
21	1198
22	663
23	28
10	8

Result 1 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	15:43:31	SELECT MAX(pizzas.price) AS highest_price FROM pizzas LIMIT 0, 200	1 row(s) returned	0.016 sec / 0.000 sec
2	15:45:34	SELECT HOUR(order_time) AS hour, COUNT(Order_id) AS id FROM orders GROUP BY HOUR(order_ti...	15 row(s) returned	0.016 sec / 0.000 sec



SET 2

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

```sql
-- (INTERMIDIATE)Join the necessary tables to find the total quantity of each pizza category ordered
SELECT
    pizza_types.CATEGORY,
    SUM(orders_details.QUANTITY) AS quantity
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.CATEGORY
ORDER BY quantity DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

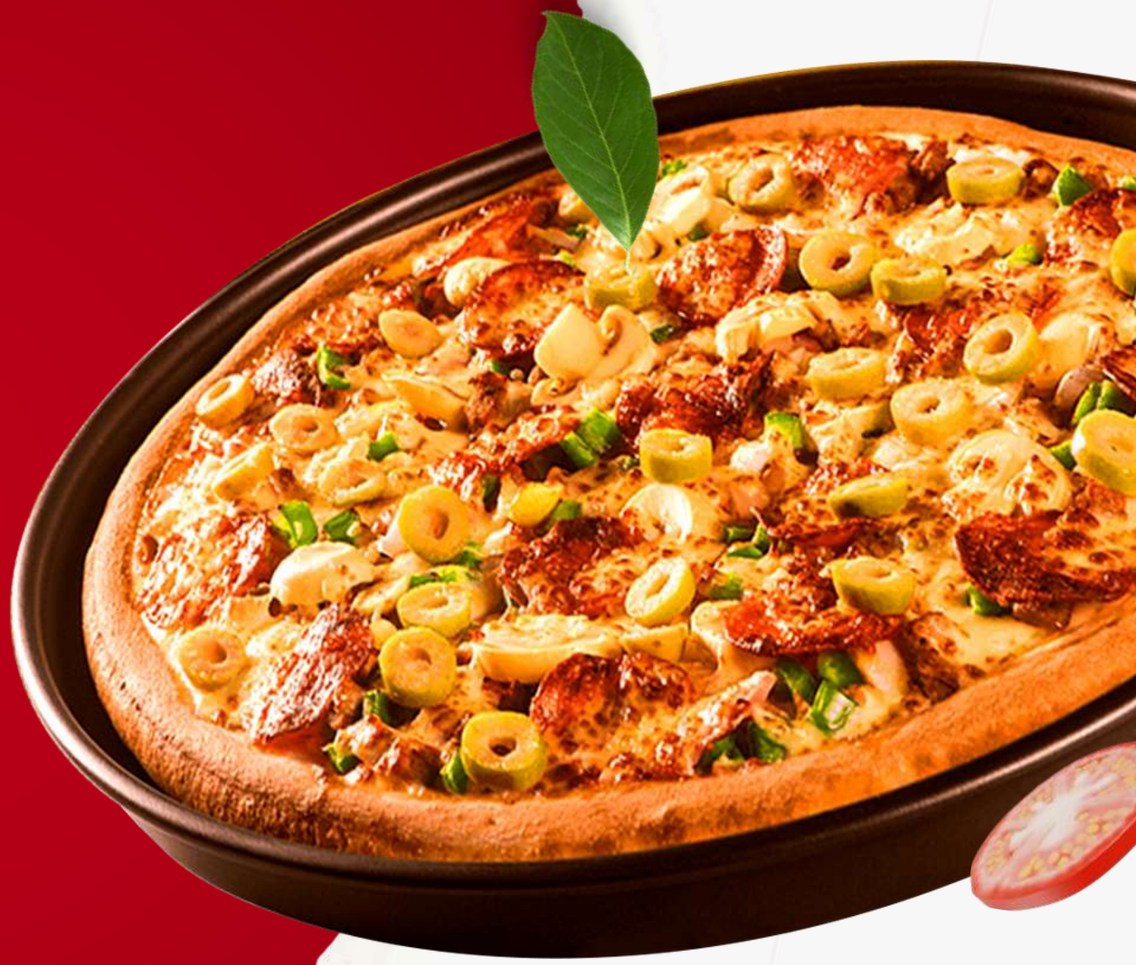
CATEGORY	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Result 2

Output

Action Output

#	Time	Action	Message
1	15:43:31	SELECT MAX(pizzas.price) AS highest_price FROM pizzas LIMIT 0, 200	1 row(s) returned
2	15:45:34	SELECT HOUR(order_time) AS hour, COUNT(Order_id) AS id FROM orders GROUP BY HOUR(order_ti...	15 row(s) returned
3	15:50:13	SELECT pizza_types.CATEGORY, SUM(orders_details.QUANTITY) AS quantity FROM pizza_types ...	4 row(s) returned



SET 2

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

```sql
-- Group the orders by date and calculate the average number of pizzas ordered per day.
SELECT
    o.ORDER_DATE, ROUND(AVG(od.QUANTITY)) AS AvgPizzasPerDay
FROM
    orders o
        JOIN
    orders_details od ON o.ORDER_ID = od.ORDER_ID
GROUP BY o.ORDER_DATE;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

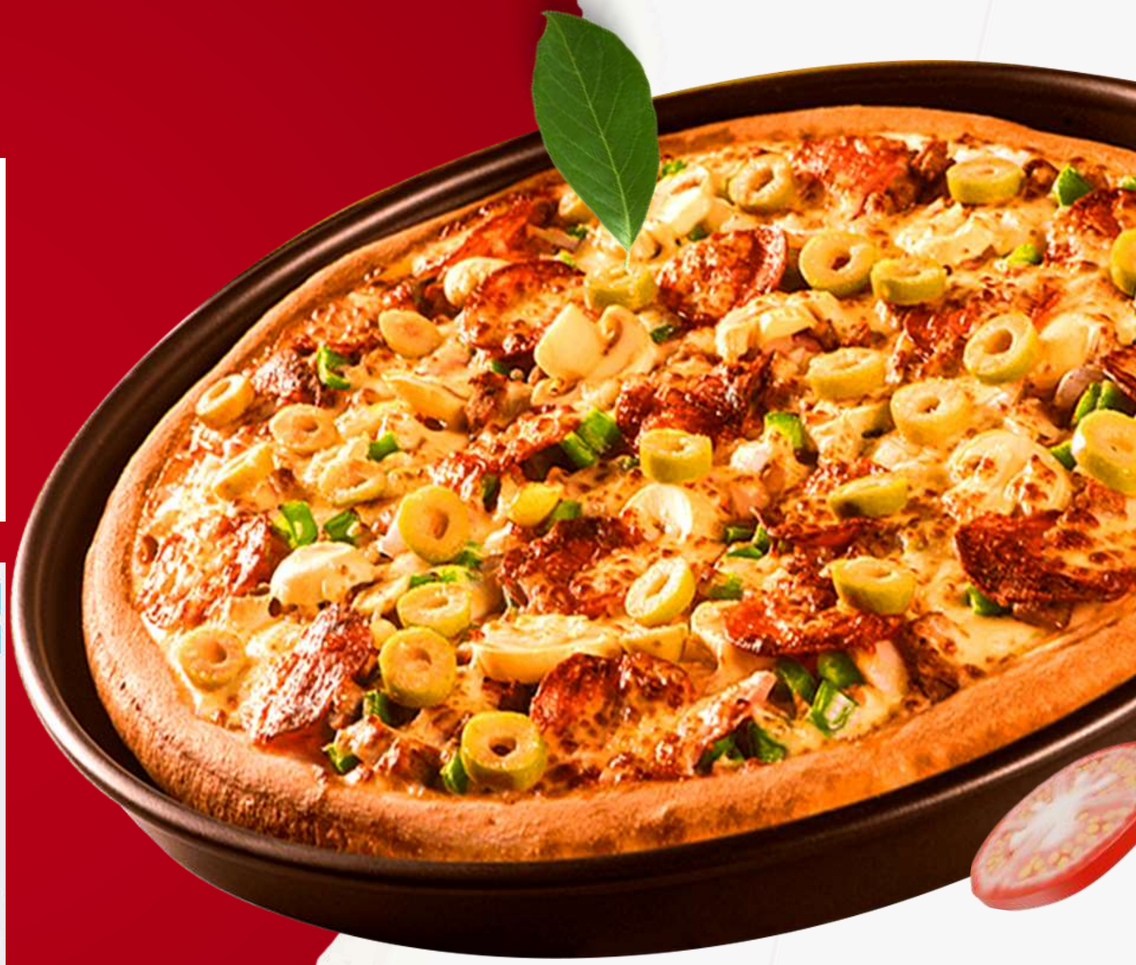
ORDER_DATE	AvgPizzasPerDay
2015-01-25	1
2015-01-26	1
2015-01-27	1
2015-01-28	1
2015-01-29	1
2015-01-30	1
2015-01-31	1
2015-02-01	1
2015-02-02	1
2015-02-03	1
2015-02-04	1
2015-02-05	1
2015-02-06	1
2015-02-07	1

Result 3 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	15:43:31	SELECT MAX(pizzas.price) AS highest_price FROM pizzas LIMIT 0, 200	1 row(s) returned	0.016 sec / 0.000 sec
2	15:45:34	SELECT HOUR(order_time) AS hour, COUNT(Order_id) AS id FROM orders GROUP BY HOUR(order_ti...	15 row(s) returned	0.016 sec / 0.000 sec
3	15:50:13	SELECT pizza_types.CATEGORY, SUM(orders_details.QUANTITY) AS quantity FROM pizza_types ...	4 row(s) returned	0.172 sec / 0.000 sec
4	15:52:45	SELECT o.ORDER_DATE, ROUND(AVG(od.QUANTITY)) AS AvgPizzasPerDay FROM orders o J...	200 row(s) returned	0.093 sec / 0.000 sec



SET 3

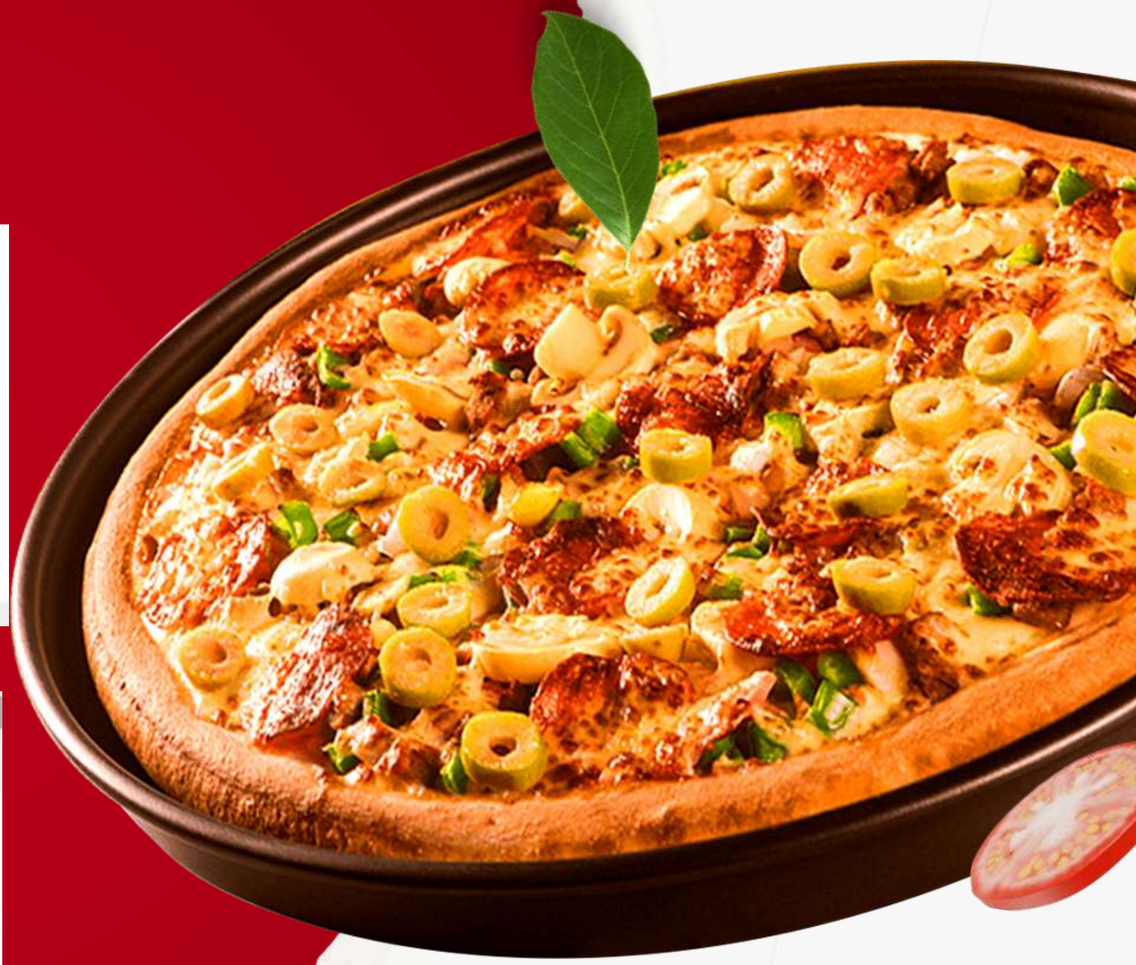
These are niche or supplementary questions that demonstrate advanced technical skills but may not always align directly with key business outcomes

1:Identify the most common pizza size ordered.

```sql
-- Identify the most common pizza size ordered
SELECT
    pizzas.size, COUNT(orders_details.order_details_id)
FROM
    pizzas
        JOIN
    orders_details ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY pizzas.size;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

size	COUNT(orders_details.order_details_id)
M	15385
L	18526
S	14137
XL	544
XXL	28



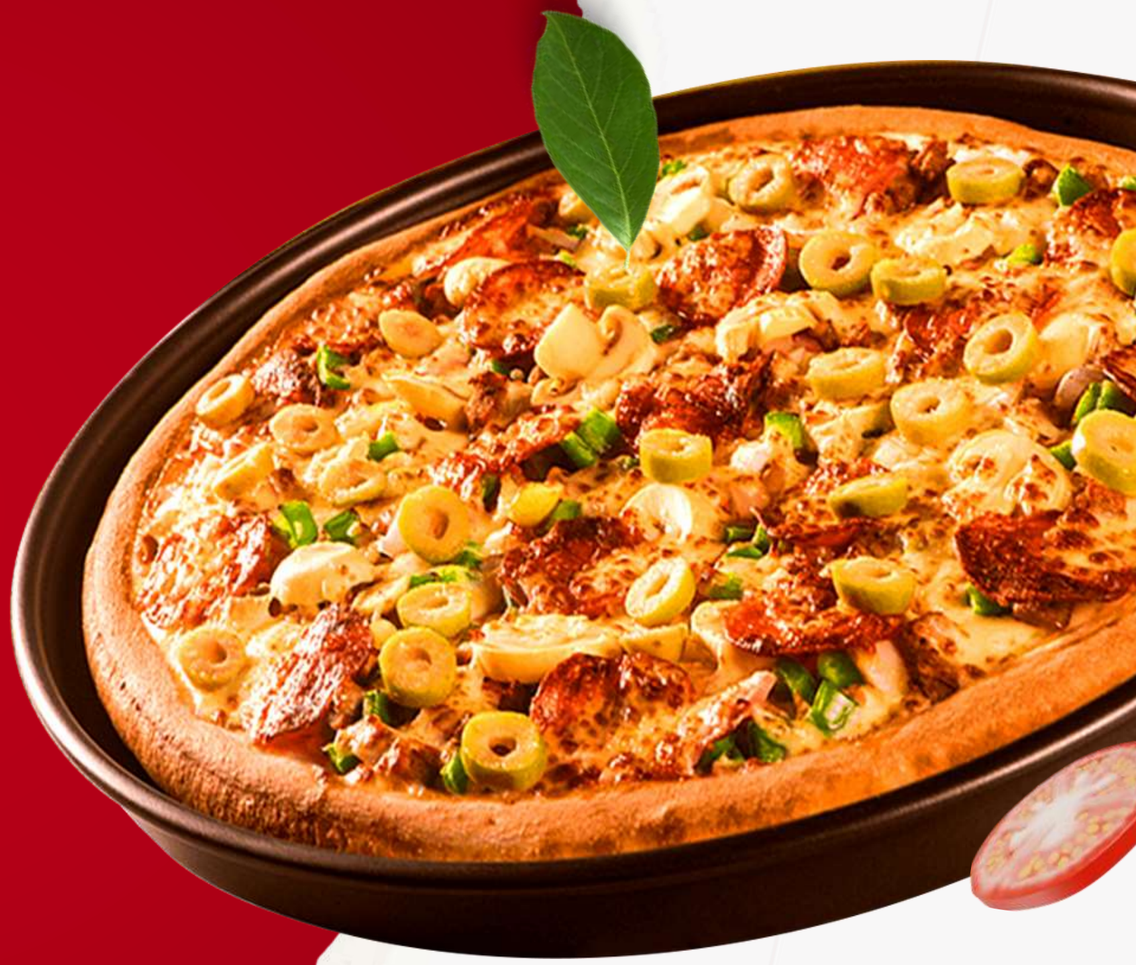
SET 3

2:Find the total number of pizzas ordered in each category.

```sql
-- Find the total number of pizzas ordered in each category
SELECT
    pizza_types.category, COUNT(pizzas.pizza_id) AS total_pizzas
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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Con

category	total_pizzas
Classic	14579
Veggie	11449
Supreme	11777
Chicken	10815

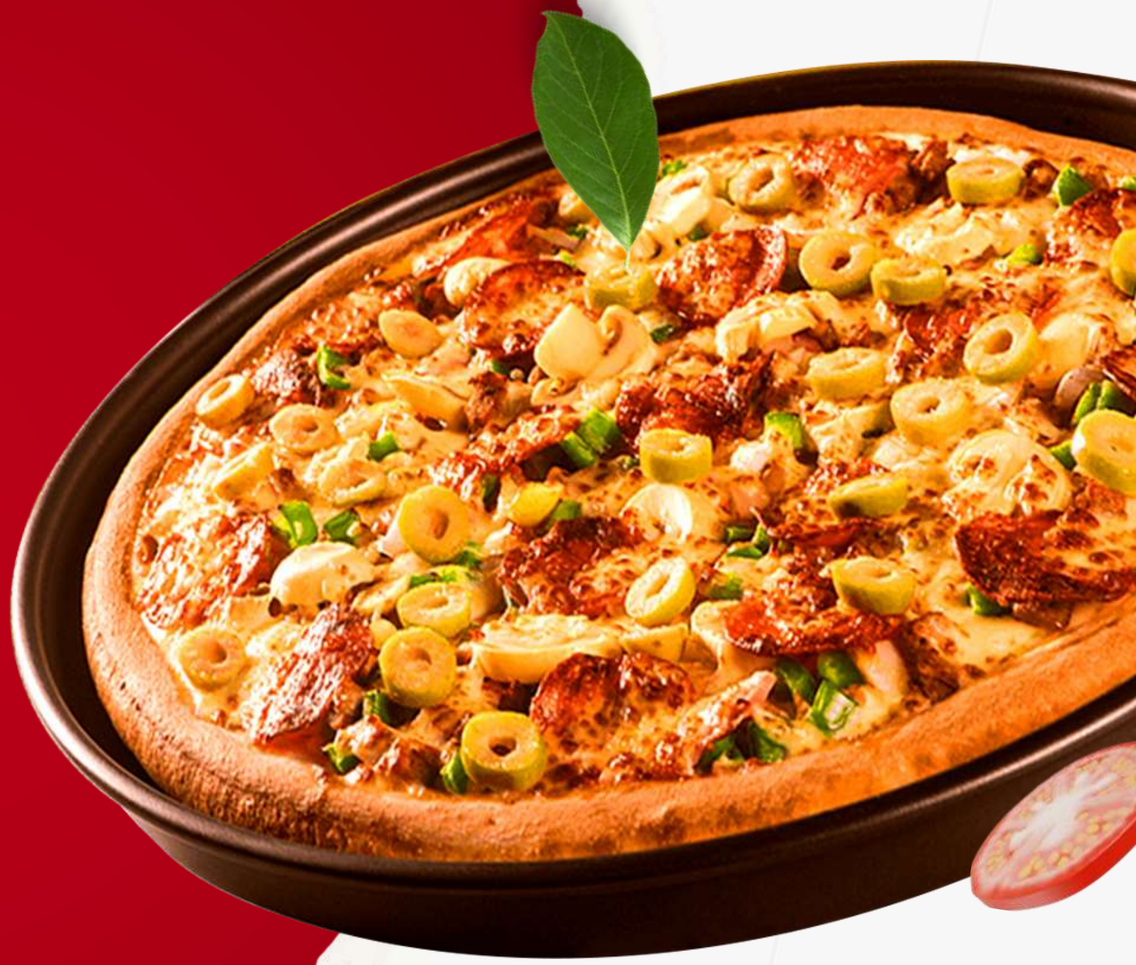


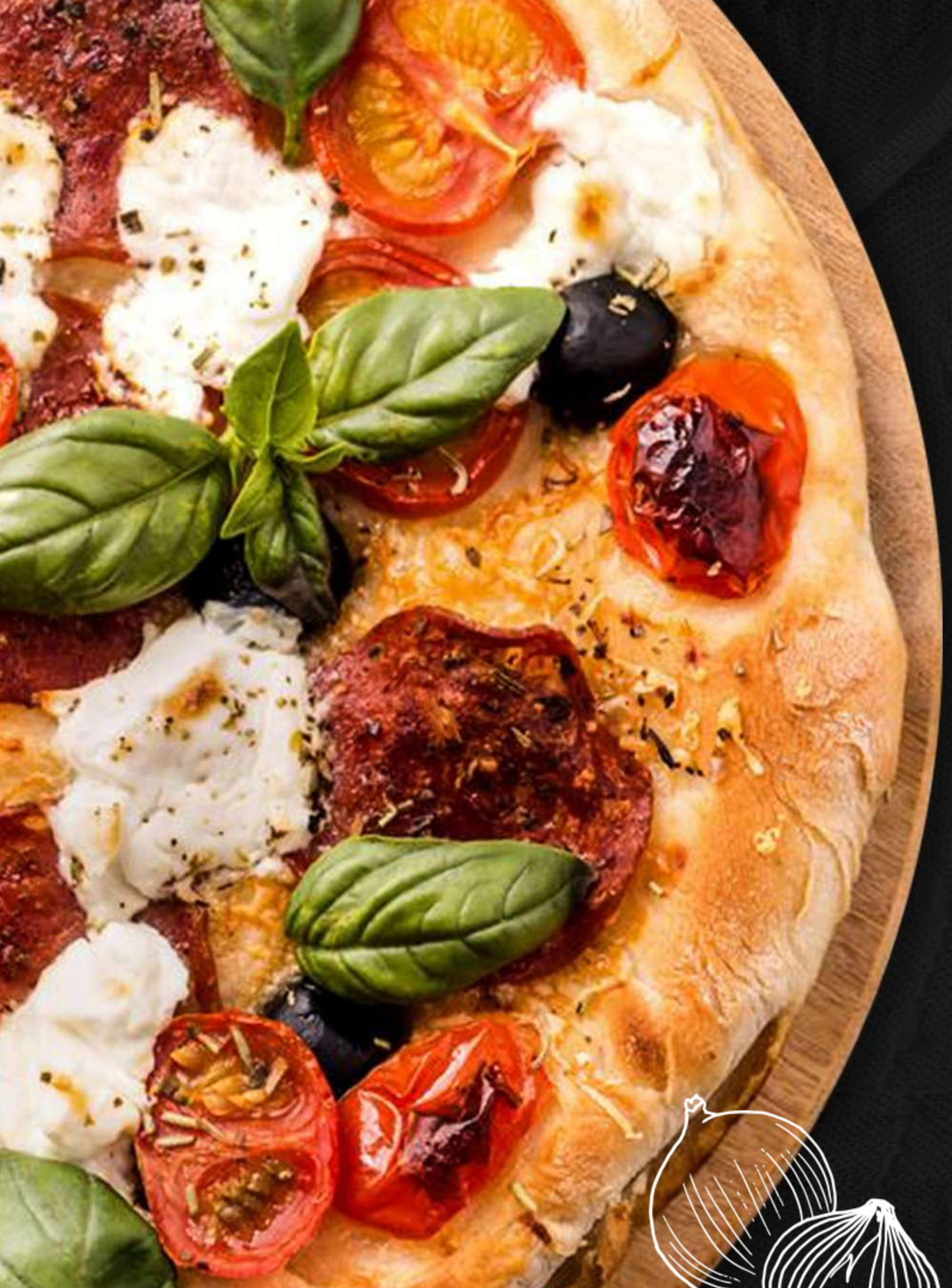
SET 3

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

```sql
-- Determine the top 3 most ordered pizza types based on revenue.
SELECT
    P.PIZZA_ID, SUM(O.QUANTITY * P.PRICE) AS Revenue
FROM
    PIZZAS AS P
        JOIN
    ORDERS_DETAILS AS O ON P.PIZZA_ID = O.PIZZA_ID
GROUP BY P.PIZZA_ID
ORDER BY Revenue DESC
LIMIT 3;
```

PIZZA_ID	Revenue
thai_ckn_l	29257.5
five_cheese_l	26066.5
four_cheese_l	23622.200000000554





DATACRUST PIZZA

SUMMARY OF FINDINGS AND CONCLUSION

1. Summary of Findings:

•**Revenue Insights:** Identified top-selling pizza types and revenue-generating pizzas.

•**Customer Behavior:** Analyzed the distribution of orders by time of day, most common pizza sizes, and categories.

•**Key Trends:** Highlighted peak hours and the most frequently ordered pizzas in each category.

2. Conclusion:

This project demonstrates the ability to derive meaningful insights from sales data, enabling businesses to make data-driven decisions that enhance operational efficiency, optimize pricing strategies, and improve customer satisfaction.

The findings provide actionable recommendations for business growth, highlighting the importance of understanding customer preferences and sales trends

DATACRUST PIZZA

THANK YOU!