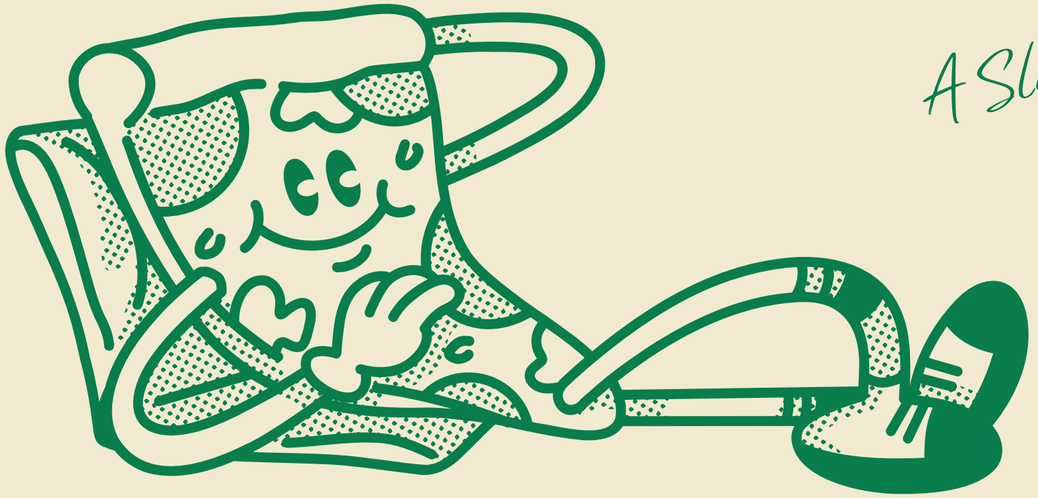
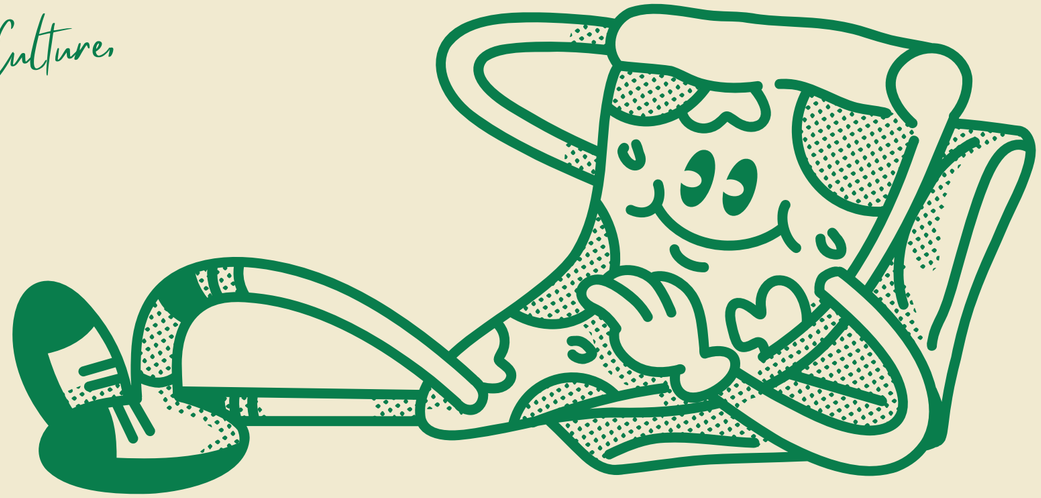
PIZZA SALES ANALYSIS



A Slice of Joy, Culture, and Flavor



HELLO

WELCOME TO MY SQL PROJECT ON PIZZA SALES ANALYSIS.

IN THIS PROJECT, I EXPLORED HOW SQL CAN ANSWER REAL BUSINESS QUESTIONS USING SALES DATA.

INTRODUCTION

THIS PROJECT USES SQL QUERIES TO ANALYZE PIZZA SALES BY ANSWERING QUESTIONS ON ORDERS, REVENUE, POPULAR PIZZAS, CATEGORY DISTRIBUTION, AND REVENUE TRENDS.

Basic:

Retrieve the total number of orders placed.

Calculate the total revenue generated from pizza sales.

Identify the highest-priced pizza.

Identify the most common pizza size ordered.

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

Intermediate:

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

Determine the distribution of orders by hour of the day.

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

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

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

Advanced:

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

Analyze the cumulative revenue generated over time.

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

DATA MODEL

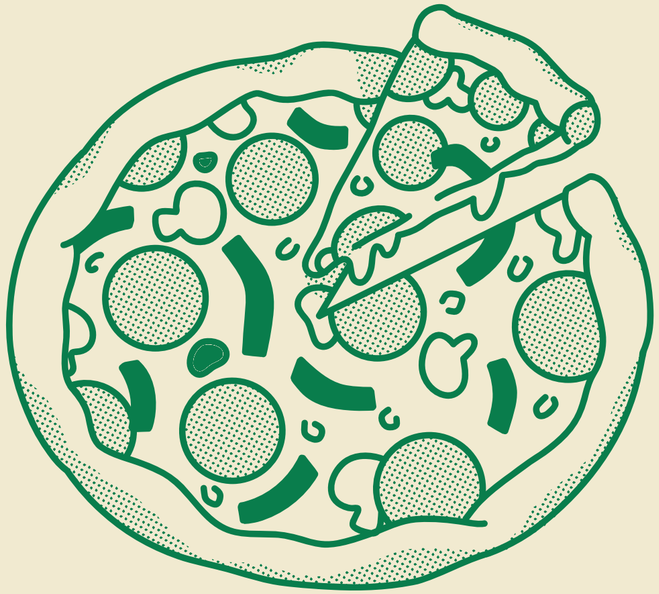

order_details
order_details_id
order_id
pizza_id
quantity
Collapse
pizzas
pizza_id
pizza_type_id
price
size
Collapse
pizza_types
category
ingredients
name
pizza_type_id
Collapse
orders
date
order_id
time
Collapse

RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

```sql
SELECT
    COUNT(order_id) AS total_orders
FROM
    orders;
```

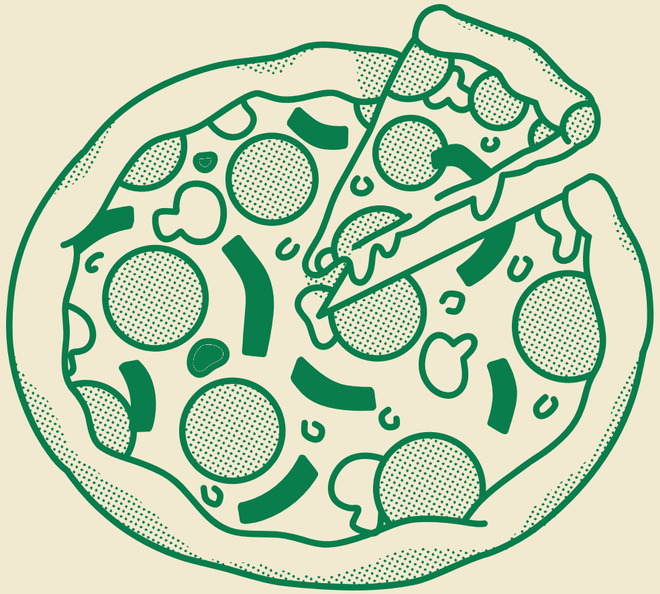
Result Grid

total_orders
21350



CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

```sql
SELECT
    ROUND(SUM(o.quantity * p.price), 2) AS total_Sales
FROM
    pizzas p
        JOIN
    order_detail o ON p.pizza_id = o.pizza_id;
```

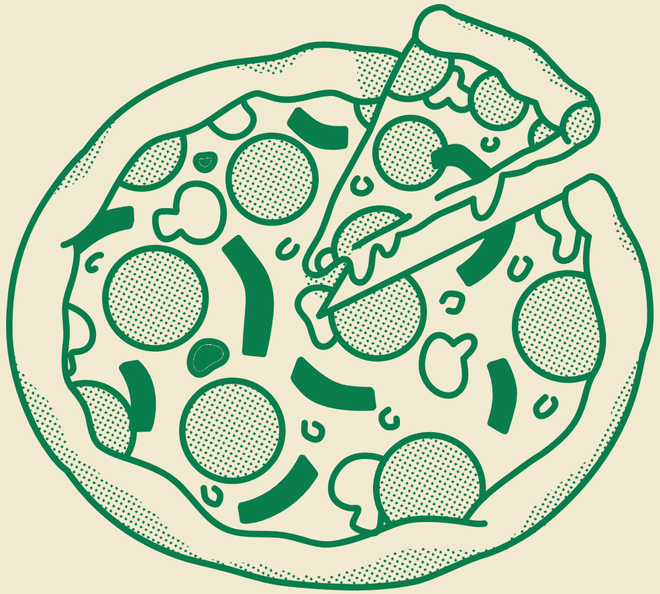


Result Grid

total_Sales
827450

IDENTIFY THE HIGHEST-PRICED PIZZA.

```sql
SELECT
    name, MAX(Price) AS max_price
FROM
    pizzas p
        JOIN
    pizza_types t ON p.pizza_type_id = t.pizza_type_id
GROUP BY name
ORDER BY max_price DESC
LIMIT 1;
```

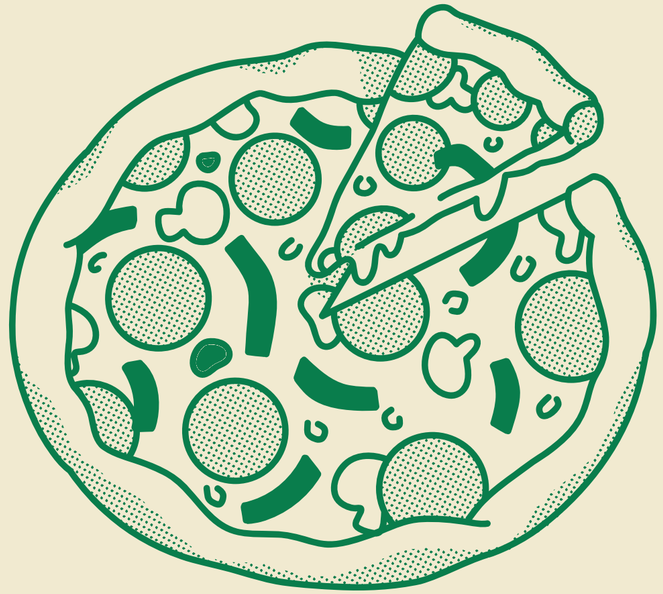


Result Grid | Filter Rows:

name	max_price
The Greek Pizza	36

IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

```
SELECT
    size, COUNT(order_details_id) AS quantity_ordered
FROM
    pizzas p
        JOIN
    order_detail o ON p.pizza_id = o.pizza_id
GROUP BY size
ORDER BY quantity_ordered DESC;
```

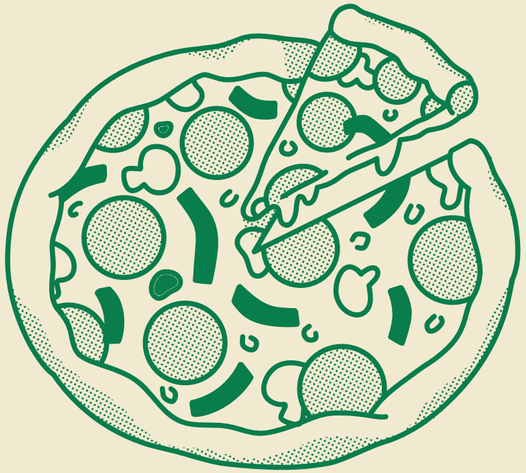


Result Grid | Filter Rows:

size	quantity_ordered
L	18526
M	15385
S	14137
XL	544
XXL	28

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

```sql
SELECT
    name, SUM(quantity) AS sum_quantity
FROM
    pizzas p
        JOIN
    order_detail o ON p.pizza_id = o.pizza_id
        JOIN
    pizza_types t ON t.pizza_type_id = p.pizza_type_id
GROUP BY name
ORDER BY sum_quantity DESC
LIMIT 5;
```

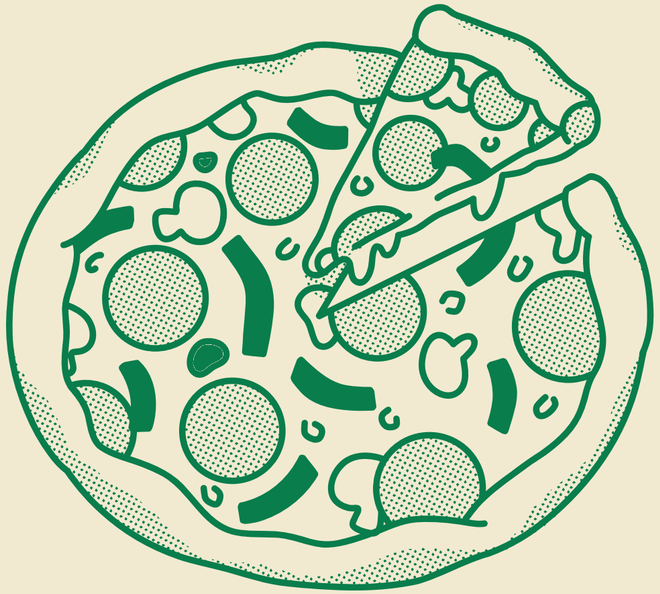


Result Grid | Filter Rows: | Export:

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

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

```
SELECT
    category, SUM(quantity) AS total_quantity
FROM
    pizza_types t
        JOIN
    pizzas p ON p.pizza_type_id = t.pizza_type_id
        JOIN
    order_detail o ON o.pizza_id = p.pizza_id
GROUP BY category
ORDER BY total_quantity DESC;
```



Result Grid | Filter Rows:

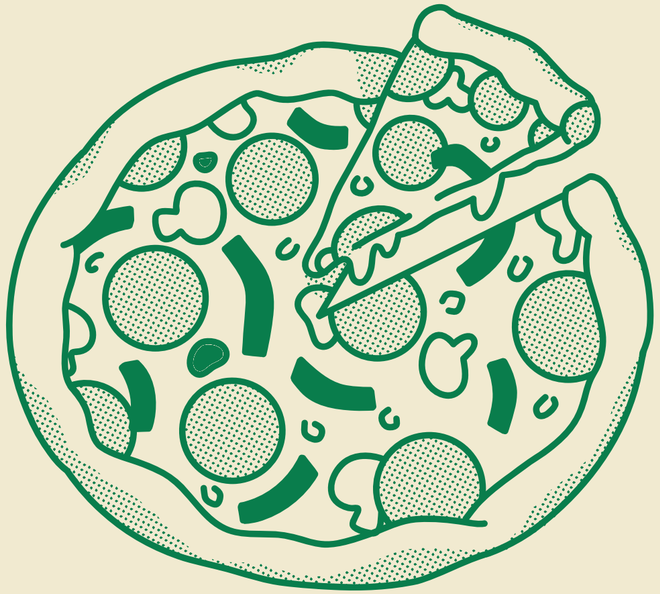
category	total_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

```sql
SELECT
    HOUR(order_time) AS Hour, COUNT(order_id) AS orders_count
FROM
    orders
GROUP BY HOUR(order_time);
```

Result Grid | Filter Rows:

Hour	orders_count
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

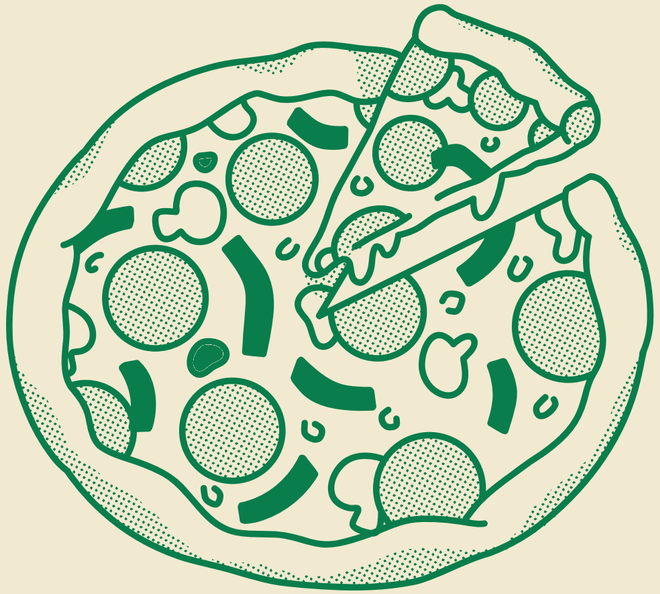


JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

```
SELECT
    category, COUNT(*)
FROM
    pizza_types
GROUP BY category;
```

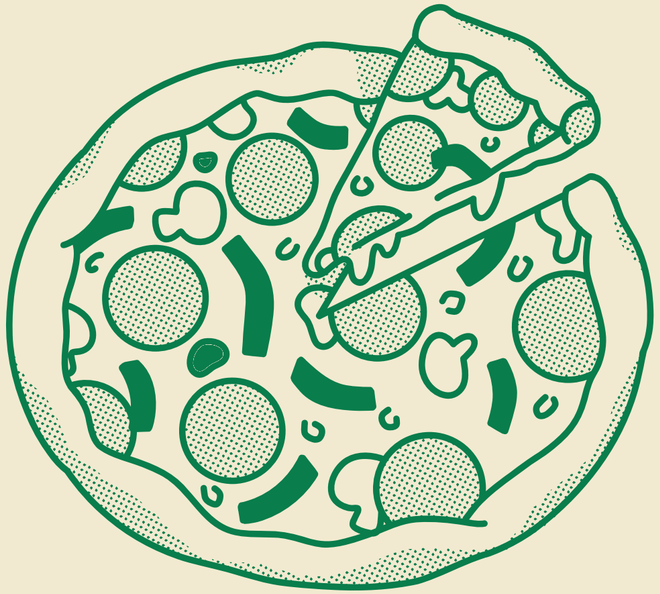
Result Grid | Filter Rows:

category	COUNT(
Chicken	6
Classic	8
Supreme	9
Veggie	9



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

```sql
SELECT
    ROUND(AVG(quantity), 0) AS avg_pizza_ordered_per_Day
FROM
    (SELECT
        o.order_date, SUM(d.quantity) AS quantity
    FROM
        orders o
    JOIN order_detail d ON o.order_id = d.order_id
    GROUP BY o.order_date) AS order_quantity;
```

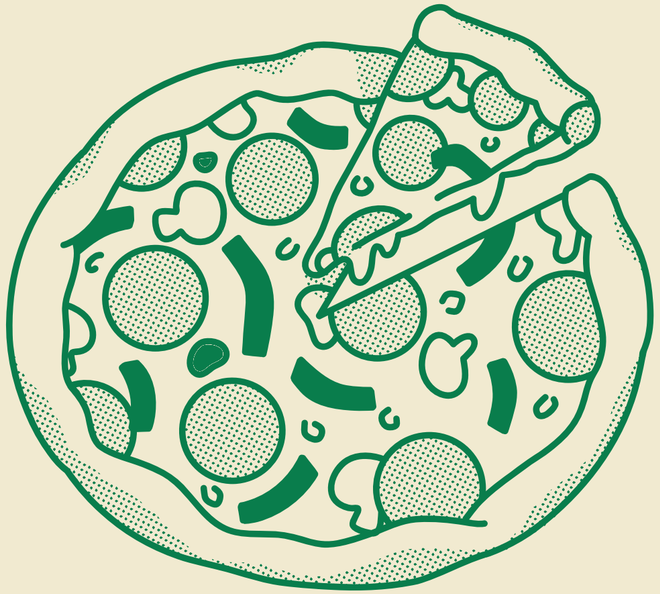


Result Grid | Filter Rows:

avg_pizza_ordered_per_Day
138

DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```sql
SELECT
    name, SUM(price * quantity) AS revenue
FROM
    pizzas p
        JOIN
    order_detail o ON p.pizza_id = o.pizza_id
        JOIN
    pizza_types t ON p.pizza_type_id = t.pizza_type_id
GROUP BY name
ORDER BY revenue DESC
LIMIT 3;
```

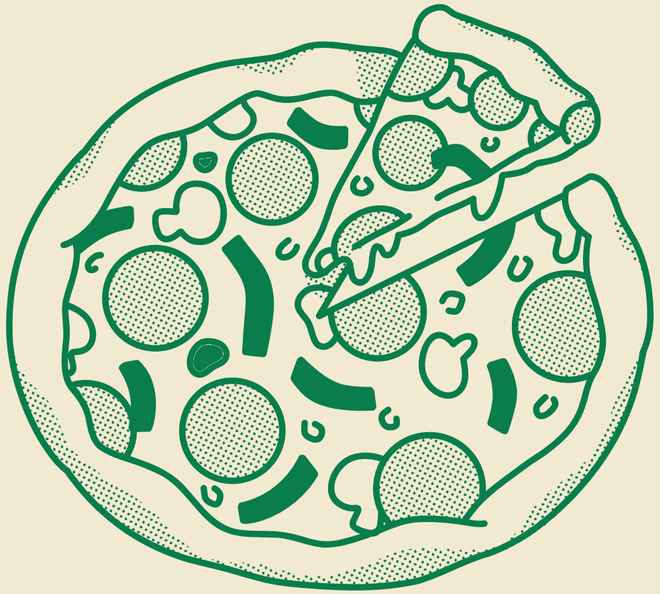


Result Grid | Filter Rows: | Expo

name	revenue
The Thai Chicken Pizza	44027
The Barbecue Chicken Pizza	43376
The California Chicken Pizza	42002

CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

```sql
SELECT
    category,
    ROUND((SUM(price * quantity) / (SELECT
                    ROUND(SUM(o.quantity * p.price), 2) AS total_Sales
                FROM
                    pizzas p
                        JOIN
                    order_detail o ON p.pizza_id = o.pizza_id)) * 100,
            2) AS revenue
FROM
    pizzas p
        JOIN
    order_detail o ON p.pizza_id = o.pizza_id
        JOIN
    pizza_types t ON t.pizza_type_id = p.pizza_type_id
GROUP BY category;
```

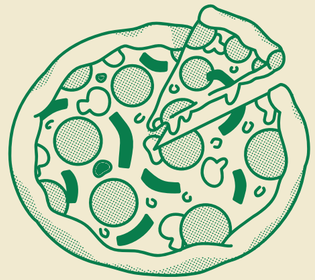


Result Grid | Filter Row

category	revenue
Classic	26.96
Veggie	23.53
Supreme	25.51
Chicken	24.01

ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME

```sql
SELECT order_date,
    SUM(revenue) OVER(order by order_date) AS cumulative_revenue
FROM
(SELECT order_date,
    SUM(quantity * price) AS revenue
FROM orders o
JOIN order_detail d
ON o.order_id = d.order_id
JOIN pizzas p
ON p.pizza_id = d.pizza_id
GROUP BY order_date) AS sales
```



Result Grid | Filter Rows:

order_date	cumulative_revenue
2015-01-01	2746
2015-01-02	5512
2015-01-03	8203
2015-01-04	9983
2015-01-05	12075
2015-01-06	14532
2015-01-07	16761
2015-01-08	19628
2015-01-09	21777
2015-01-10	24270
2015-01-11	26161
2015-01-12	28105
2015-01-13	30182
2015-01-14	32742

Result 10

DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.

```sql
SELECT name, revenue
FROM
(SELECT category, name, revenue,
RANK() OVER(partition by category ORDER by revenue DESC) AS rn
from
(SELECT category, name,
    SUM(price * quantity) AS revenue
FROM order_detail d
JOIN pizzas p
ON p.pizza_id = d.pizza_id
JOIN pizza_types t
ON t.pizza_type_id = p.pizza_type_id
GROUP BY name, category
ORDER BY revenue DESC) AS a) AS b
WHERE rn <= 3;
```

Result Grid | Filter Rows: | Expo

name	revenue
The Thai Chicken Pizza	44027
The Barbecue Chicken Pizza	43376
The California Chicken Pizza	42002
The Classic Deluxe Pizza	38417
The Hawaiian Pizza	33122
The Pepperoni Pizza	30637
The Spicy Italian Pizza	35516
The Italian Supreme Pizza	34232
The Sicilian Pizza	30456
The Four Cheese Pizza	32478
The Five Cheese Pizza	26771
The Mexicana Pizza	26564

KEY LEARNING

STEP 01

LEARNED HOW TO APPLY SQL QUERIES (BASIC TO ADVANCED) TO ANALYZE REAL-WORLD SALES DATA.

STEP 02

UNDERSTOOD THE USE OF JOINS, AGGREGATIONS, AND GROUP FUNCTIONS TO GENERATE MEANINGFUL INSIGHTS.

STEP 03

IDENTIFIED SALES TRENDS, POPULAR PIZZAS, AND REVENUE PATTERNS THROUGH STRUCTURED ANALYSIS.

STEP 04

IMPROVED PROBLEM-SOLVING AND ANALYTICAL THINKING BY TRANSLATING BUSINESS QUESTIONS INTO SQL QUERIES.

THANK YOU FOR TAKING THE TIME TO GO THROUGH MY PROJECT. THIS ANALYSIS GAVE ME THE OPPORTUNITY TO APPLY SQL CONCEPTS IN A PRACTICAL WAY, FROM BASIC QUERIES TO ADVANCED ANALYSIS, WHILE EXPLORING INSIGHTS FROM PIZZA SALES DATA.

THANK YOU