# Pizza Sales Report

Vedant Thorat

## ABSTRACT

This project focuses on analyzing pizza sales data using a relational database management system (RDBMS). The objective is to gain insights into various aspects of pizza sales, including order patterns, revenue generation, and popular pizza types. By leveraging SQL queries and database management techniques, the project aims to provide valuable insights for pizza businesses to optimize their operations and enhance customer satisfaction.

The database consists of four interconnected tables: Pizzas, Pizza_types, Orders, and order_details. Each table captures different dimensions of the pizza sales process, such as pizza types, order details, and customer information. By querying these tables, the project addresses a range of questions, including total orders placed, revenue generated, highest-priced pizza, and common pizza sizes ordered.

Through the analysis, the project highlights trends in pizza sales, identifies top-performing pizza types, and provides recommendations for pricing strategies and inventory management. By utilizing the power of RDBMS and SQL, this project demonstrates the importance of data-driven decision-making in the food industry and showcases the potential for leveraging relational databases to optimize business processes and drive profitability.

## Introduction

In the bustling world of food service, understanding customer preferences, optimizing inventory management, and maximizing revenue are essential for success. Among the myriad culinary delights enjoyed worldwide, pizza stands out as a beloved favorite, offering a versatile canvas for flavor exploration and customization. In this context, the management and analysis of pizza sales data hold immense significance for pizza businesses seeking to thrive in a competitive market.

This project delves into the realm of pizza sales analysis, leveraging the capabilities of a Relational Database Management System (RDBMS) to extract actionable insights from a comprehensive dataset. Through the lens of SQL queries and database management techniques, the project endeavors to unravel the intricacies of pizza sales dynamics, uncovering patterns, trends, and opportunities that can inform strategic decision-making.

At its core, the project aims to address key questions pertinent to pizza sales operations, ranging from fundamental inquiries such as total orders placed and revenue generated, to more nuanced investigations into popular pizza types, pricing strategies, and customer preferences. By harnessing the power of relational databases, the project endeavors to empower pizza businesses with the knowledge and tools necessary to optimize their operations, enhance customer satisfaction, and drive profitability.

Through meticulous data analysis and interpretation, this project seeks not only to illuminate the current landscape of pizza sales but also to provide actionable insights and recommendations for future growth and success. By embracing a data-driven approach, pizza businesses can navigate the complexities of the market with confidence, ensuring that every slice served is not just delicious, but also strategically positioned for success.

## Database Schema

The database for the pizza sales project consists of four interconnected tables designed to capture various aspects of the pizza sales process. Each table serves a specific purpose and contributes to the comprehensive analysis of pizza sales data. Below is the schema detailing the structure of each table:

1. Pizzas Table

| Column Name | Data Type | Description |
|---|---|---|
| Pizza_ID | VARCHAR (20) | Pizzas ID (Primary Key) |
| Pizza_Type_ID | VARCHAR (20) | Types of pizza ID |
| Size | VARCHAR (5) | Size of pizzas |
| Price | INT | Price of pizzas |

2. Pizza_Types

| Column Name | Data Type | Description |
|---|---|---|
| Pizza_Type_ID | VARCHAR (50) | Types of pizza ID (Primary key) |
| Name | TEXT | Names of pizza |
| Category | TEXT | Category of pizza |
| Ingredient | TEXT | Ingredient in pizzas |

3. Orders

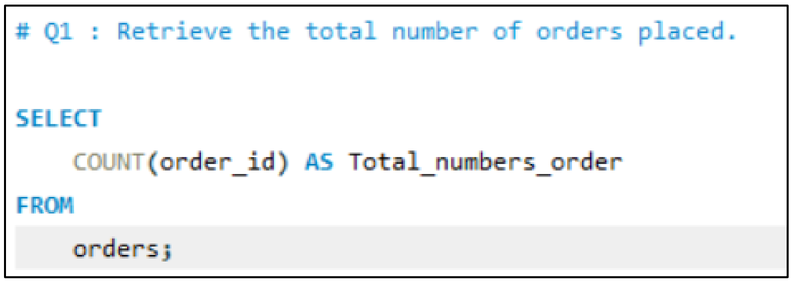
| Column Name | Data Type | Description |
|---|---|---|
| Order_ID | INT | Order unique ID (Primary Key) |
| Date | DATE | Date of order |
| Time | TIME | Time of order |

4. Order_Details

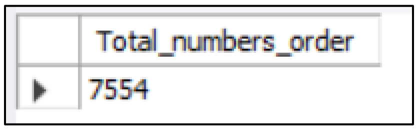
| Column Name | Data Type | Description |
|---|---|---|
| Order_Details_ID | INT | Order Details ID (Primary Key) |
| Order_ID | INT | Order ID (Foreign Key) |
| Pizza_ID | VARCHAR (20) | Pizzas ID (Foreign Key) |
| Quantity | INT | Quantity |

## Question Solved

1. Retrieve the total number of orders placed.



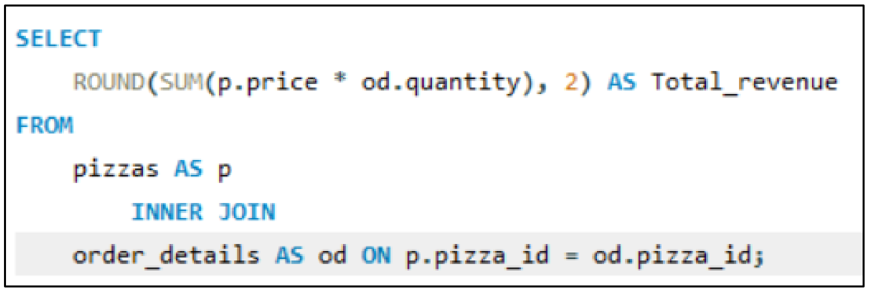
```
# Q1 : Retrieve the total number of orders placed.

SELECT
    COUNT(order_id) AS Total_numbers_order
FROM
    orders;
```



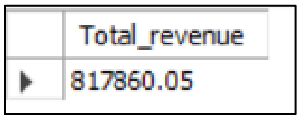
```
Total_numbers_order
7554
```

This query provides a fundamental metric indicating the volume of orders processed within a given time frame. A higher number of orders may indicate increased demand, while a lower number could signify potential areas for improvement in marketing or customer engagement strategies.

2. Calculate the total revenue generated from pizza sales:



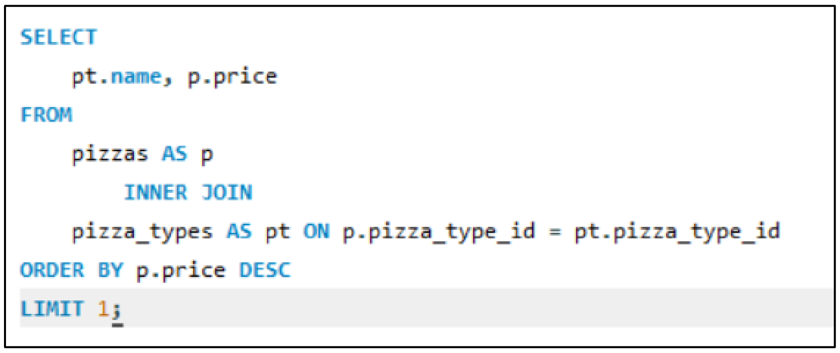
```
SELECT
    ROUND(SUM(p.price * od.quantity), 2) AS Total_revenue
FROM
    pizzas AS p
        INNER JOIN
    order_details AS od ON p.pizza_id = od.pizza_id;
```



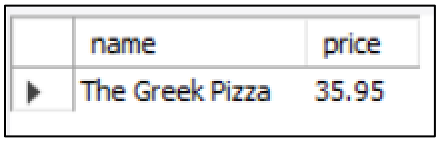
| Total_revenue |
|---|
| 817860.05 |

Total revenue is a key performance indicator reflecting the financial success of the business. By summing up the prices of all pizzas sold, this query quantifies the overall financial impact of pizza sales, guiding decisions related to pricing strategies, profit margins, and revenue forecasting.

## 3. Identify the highest-priced pizza:



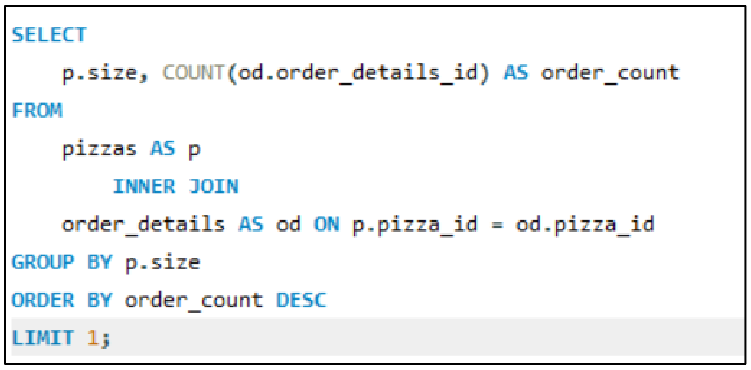
```
SELECT
    pt.name, p.price
FROM
    pizzas AS p
        INNER JOIN
    pizza_types AS pt ON p.pizza_type_id = pt.pizza_type_id
ORDER BY p.price DESC
LIMIT 1;
```



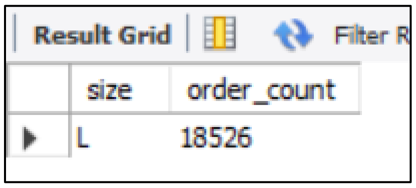
| name | price |
|---|---|
| The Greek Pizza | 35.95 |

Identify the highest-priced pizza offers insights into customer preferences for premium or specialty pizzas. Understanding which pizzas command the highest prices enables businesses to capitalize on market demand for upscale offerings and tailor marketing strategies accordingly.

## 4. Identify the most common pizza size ordered:



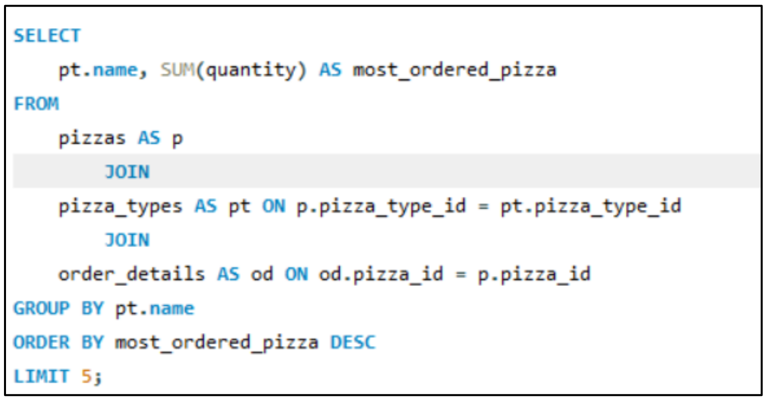
```
SELECT
    p.size, COUNT(od.order_details_id) AS order_count
FROM
    pizzas AS p
        INNER JOIN
    order_details AS od ON p.pizza_id = od.pizza_id
GROUP BY p.size
ORDER BY order_count DESC
LIMIT 1;
```



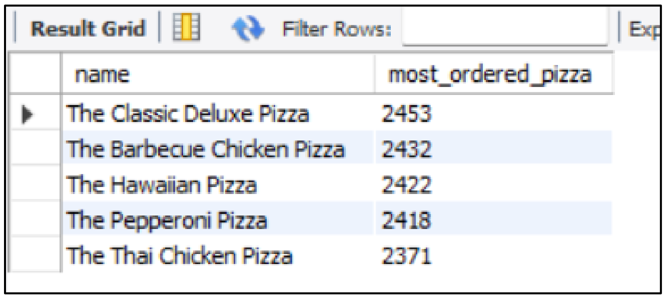
| size | order_count |
|---|---|
| L | 18526 |

The popularity of different pizza sizes can inform inventory management, production planning, and pricing strategies. By identifying the most commonly ordered size, businesses can optimize their supply chain, streamline operations, and ensure customer satisfaction by maintaining adequate stock levels of preferred sizes.

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



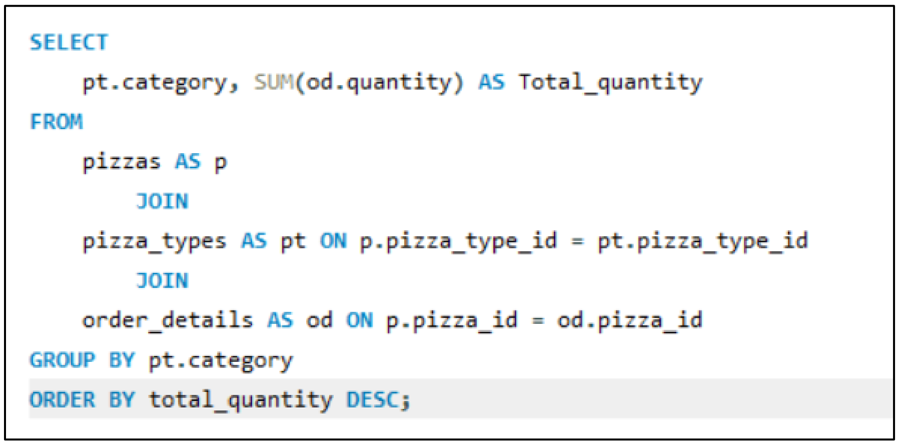
```sql
SELECT
    pt.name, SUM(quantity) AS most_ordered_pizza
FROM
    pizzas AS p
        JOIN
    pizza_types AS pt ON p.pizza_type_id = pt.pizza_type_id
        JOIN
    order_details AS od ON od.pizza_id = p.pizza_id
GROUP BY pt.name
ORDER BY most_ordered_pizza DESC
LIMIT 5;
```



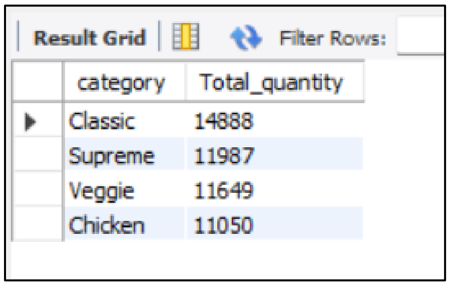
| name | most_ordered_pizza |
|---|---|
| The Classic Deluxe Pizza | 2453 |
| The Barbecue Chicken Pizza | 2432 |
| The Hawaiian Pizza | 2422 |
| The Pepperoni Pizza | 2418 |
| The Thai Chicken Pizza | 2371 |

This query highlights the best-selling pizza types, shedding light on customer preferences and consumption patterns. By identifying the top-performing pizza types, businesses can focus on promoting popular offerings, optimizing menu offerings, and leveraging cross-selling opportunities to maximize sales and revenue.

6. Find the total quantity of each pizza category ordered:



```
SELECT
    pt.category, SUM(od.quantity) AS Total_quantity
FROM
    pizzas AS p
        JOIN
    pizza_types AS pt ON p.pizza_type_id = pt.pizza_type_id
        JOIN
    order_details AS od ON p.pizza_id = od.pizza_id
GROUP BY pt.category
ORDER BY total_quantity DESC;
```

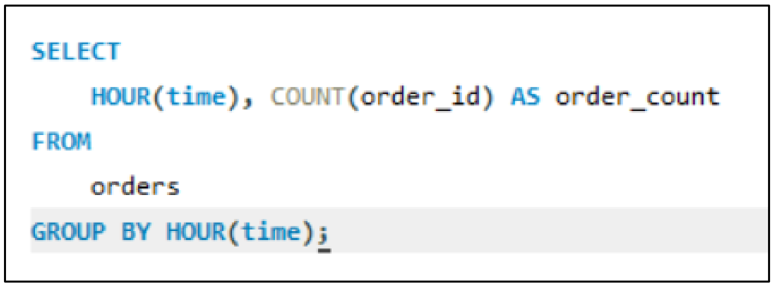


Result Grid
Filter Rows:

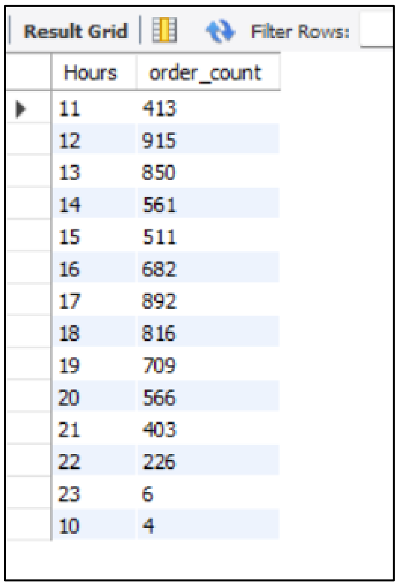
| category | Total_quantity |
|---|---|
| Classic | 14888 |
| Supreme | 11987 |
| Veggie | 11649 |
| Chicken | 11050 |

Analyzing the total quantity of each pizza category ordered provides valuable insights into the popularity of different pizza categories. By grouping pizzas by category and summing their quantities, businesses can identify trends, adjust inventory levels, and tailor marketing efforts to capitalize on consumer preferences for specific pizza categories.

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



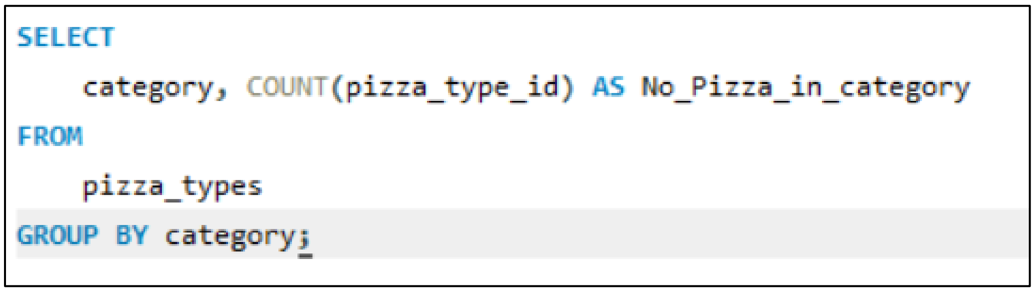
```
SELECT
    HOUR(time), COUNT(order_id) AS order_count
FROM
    orders
GROUP BY HOUR(time);
```



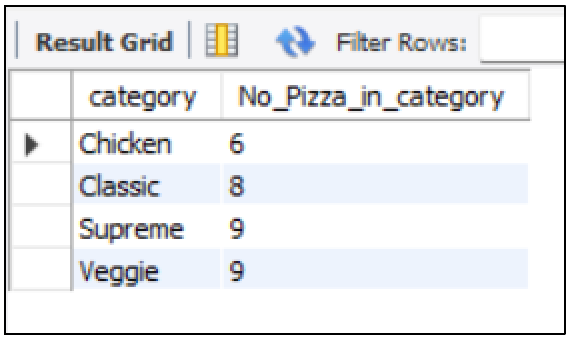
| Hours | order_count |
|---|---|
| 11 | 413 |
| 12 | 915 |
| 13 | 850 |
| 14 | 561 |
| 15 | 511 |
| 16 | 682 |
| 17 | 892 |
| 18 | 816 |
| 19 | 709 |
| 20 | 566 |
| 21 | 403 |
| 22 | 226 |
| 23 | 6 |
| 10 | 4 |

Understanding the distribution of orders by hour helps businesses optimize staffing, production, and delivery schedules to meet fluctuating demand throughout the day. By analyzing order patterns, businesses can allocate resources more efficiently, reduce wait times, and enhance overall customer experience.

## 8. Find the category-wise distribution of pizzas:



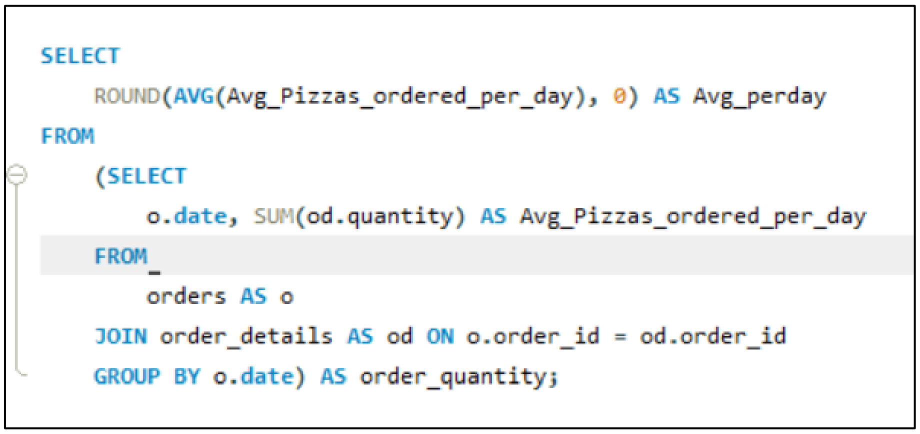
```
SELECT
    category, COUNT(pizza_type_id) AS No_Pizza_in_category
FROM
    pizza_types
GROUP BY category;
```



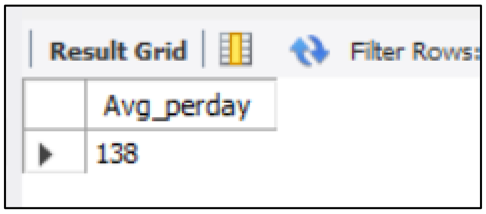
| category | No_Pizza_in_category |
|---|---|
| Chicken | 6 |
| Classic | 8 |
| Supreme | 9 |
| Veggie | 9 |

Analyzing the category-wise distribution of pizzas offers insights into the popularity of different pizza categories among customers. By aggregating pizza quantities by category, businesses can identify trends, adjust inventory levels, and tailor marketing strategies to promote popular categories and optimize menu offerings.

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



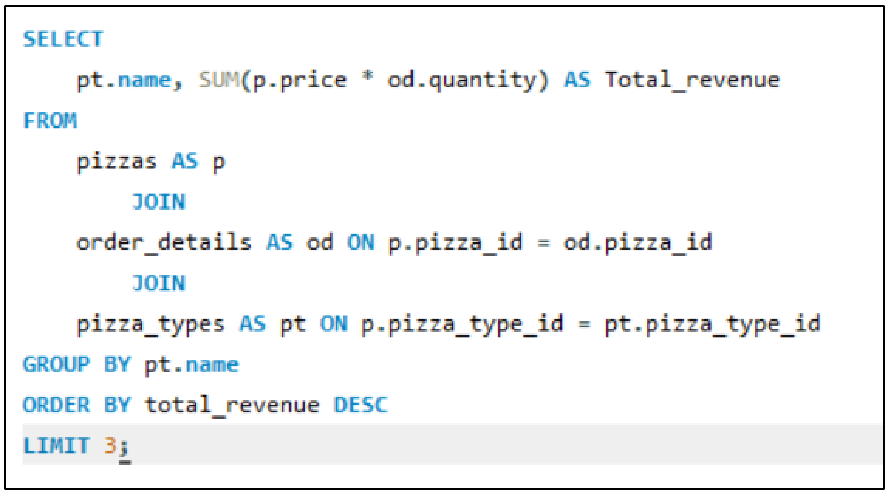
```
SELECT
    ROUND(AVG(Avg_Pizzas_ordered_per_day), 0) AS Avg_perday
FROM
    (SELECT
        o.date, SUM(od.quantity) AS Avg_Pizzas_ordered_per_day
    FROM
        orders AS o
    JOIN order_details AS od ON o.order_id = od.order_id
    GROUP BY o.date) AS order_quantity;
```



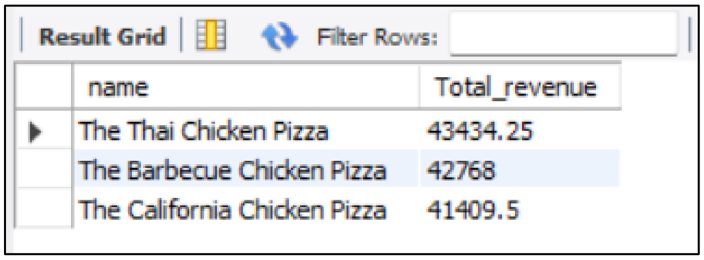
```
Avg_perday
138
```

Calculation the average number of pizzas ordered per day helps businesses understand demand trends over time and plan production and staffing accordingly. By analyzing average daily order volumes, businesses can identify peak periods, forecast future demand, and optimize resource allocation to meet customer needs efficiently.

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



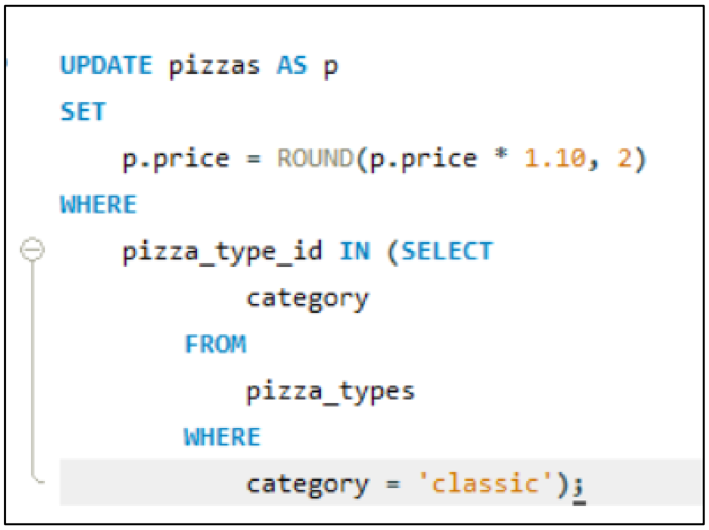
```
SELECT
    pt.name, SUM(p.price * od.quantity) AS Total_revenue
FROM
    pizzas AS p
        JOIN
    order_details AS od ON p.pizza_id = od.pizza_id
        JOIN
    pizza_types AS pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.name
ORDER BY total_revenue DESC
LIMIT 3;
```



| name | Total_revenue |
|---|---|
| The Thai Chicken Pizza | 43434.25 |
| The Barbecue Chicken Pizza | 42768 |
| The California Chicken Pizza | 41409.5 |

Identifying the top-selling pizza types based on revenue provides insights into the most profitable offerings. By analyzing revenue generated by each pizza type, businesses can prioritize marketing efforts, adjust pricing strategies, and optimize menu offerings to maximize profitability and customer satisfaction.

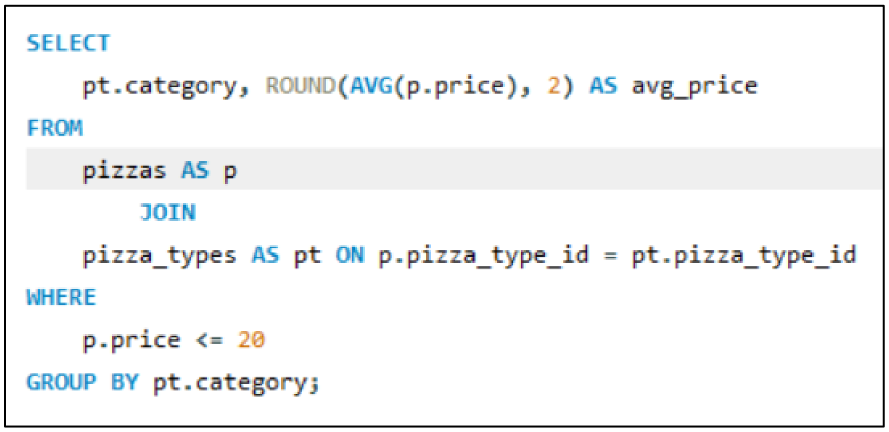
11. Update the prices of all pizzas in a specific category by increasing them by 10%, rounding to the nearest dollar:



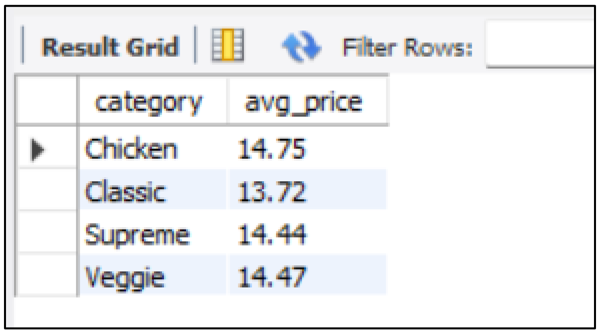
```
UPDATE pizzas AS p
SET
    p.price = ROUND(p.price * 1.10, 2)
WHERE
    pizza_type_id IN (SELECT
            category
        FROM
            pizza_types
        WHERE
            category = 'classic');
```

Query facilitates dynamic pricing adjustments based on category-specific considerations. By updating prices programmatically, businesses can respond to market trends, cost fluctuations, or promotional strategies while ensuring consistency and accuracy in pricing across categories.

12. Write a SQL query to find the average price of pizzas for each category, excluding pizzas with a price greater than $20:



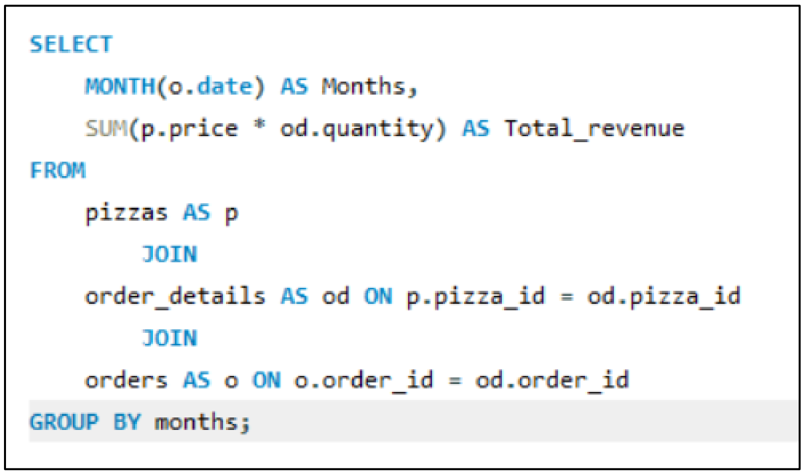
```
SELECT
    pt.category, ROUND(AVG(p.price), 2) AS avg_price
FROM
    pizzas AS p
        JOIN
    pizza_types AS pt ON p.pizza_type_id = pt.pizza_type_id
WHERE
    p.price <= 20
GROUP BY pt.category;
```

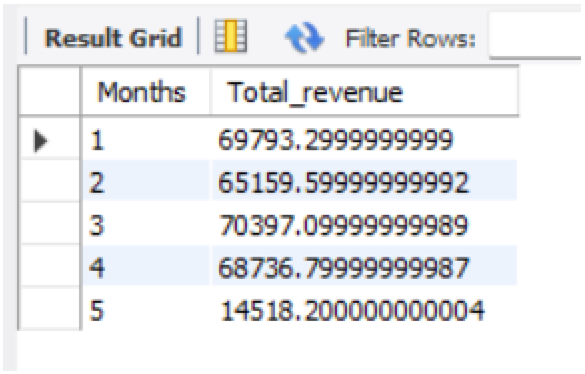

Result Grid | Filter Rows:

| category | avg_price |
|---|---|
| Chicken | 14.75 |
| Classic | 13.72 |
| Supreme | 14.44 |
| Veggie | 14.47 |

Calculating the average price of pizzas for each category allows businesses to assess pricing dynamics and competitiveness within each product segment. By excluding high-priced outliers, this query provides a more representative measure of category-specific pricing trends, enabling businesses to make informed pricing decisions and maintain competitive pricing strategies.

13. Write a SQL query to calculate the total revenue generated from pizza orders in months wise.

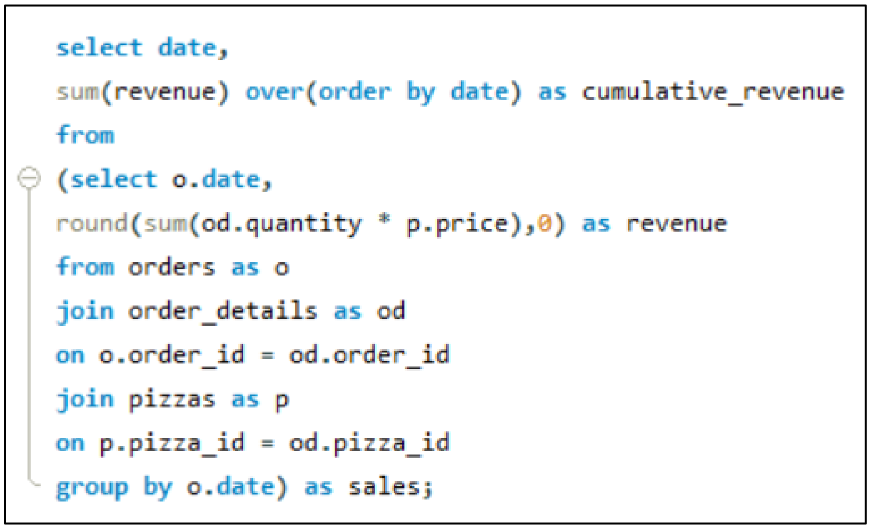

```
SELECT
    MONTH(o.date) AS Months,
    SUM(p.price * od.quantity) AS Total_revenue
FROM
    pizzas AS p
        JOIN
    order_details AS od ON p.pizza_id = od.pizza_id
        JOIN
    orders AS o ON o.order_id = od.order_id
GROUP BY months;
```


Result Grid | Filter Rows:

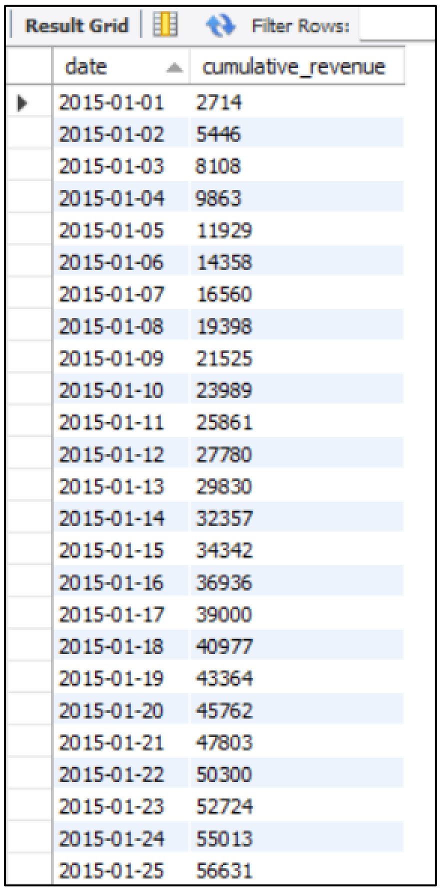
| Months | Total_revenue |
|---|---|
| 1 | 69793.2999999999 |
| 2 | 65159.59999999992 |
| 3 | 70397.09999999989 |
| 4 | 68736.79999999987 |
| 5 | 14518.20000000004 |

Analyzing revenue on a monthly basis helps businesses track performance trends over time and identify seasonal fluctuations in demand. By aggregating revenue by month, businesses can identify revenue peaks and valleys, plan marketing campaigns, and allocate resources effectively to capitalize on seasonal variations in consumer behavior.

14. Analyze the cumulative revenue generated over time.
15.



```
select date,
sum(revenue) over(order by date) as cumulative_revenue
from
(select o.date,
round(sum(od.quantity * p.price),0) as revenue
from orders as o
join order_details as od
on o.order_id = od.order_id
join pizzas as p
on p.pizza_id = od.pizza_id
group by o.date) as sales;
```



| date | cumulative_revenue |
|---|---|
| 2015-01-01 | 2714 |
| 2015-01-02 | 5446 |
| 2015-01-03 | 8108 |
| 2015-01-04 | 9863 |
| 2015-01-05 | 11929 |
| 2015-01-06 | 14358 |
| 2015-01-07 | 16560 |
| 2015-01-08 | 19398 |
| 2015-01-09 | 21525 |
| 2015-01-10 | 23989 |
| 2015-01-11 | 25861 |
| 2015-01-12 | 27780 |
| 2015-01-13 | 29830 |
| 2015-01-14 | 32357 |
| 2015-01-15 | 34342 |
| 2015-01-16 | 36936 |
| 2015-01-17 | 39000 |
| 2015-01-18 | 40977 |
| 2015-01-19 | 43364 |
| 2015-01-20 | 45762 |
| 2015-01-21 | 47803 |
| 2015-01-22 | 50300 |
| 2015-01-23 | 52724 |
| 2015-01-24 | 55013 |
| 2015-01-25 | 56631 |

Cumulative revenue analysis allows businesses to observe revenue trends over time, providing insights into the overall direction of the business's financial performance. A steady upward trend in cumulative revenue indicates healthy growth, while fluctuations or stagnation may signal areas for improvement or external factors impacting sales.

## Conclusion:

The analysis of cumulative revenue generated over time provides valuable insights into the financial performance and growth trajectory of the pizza business. By tracking revenue trends, identifying top-selling products, and assessing the impact of strategic initiatives, businesses can make informed decisions to optimize operations and drive profitability. Despite fluctuations in revenue due to seasonal variations and market dynamics, the overall trend indicates steady growth and resilience in the face of challenges.