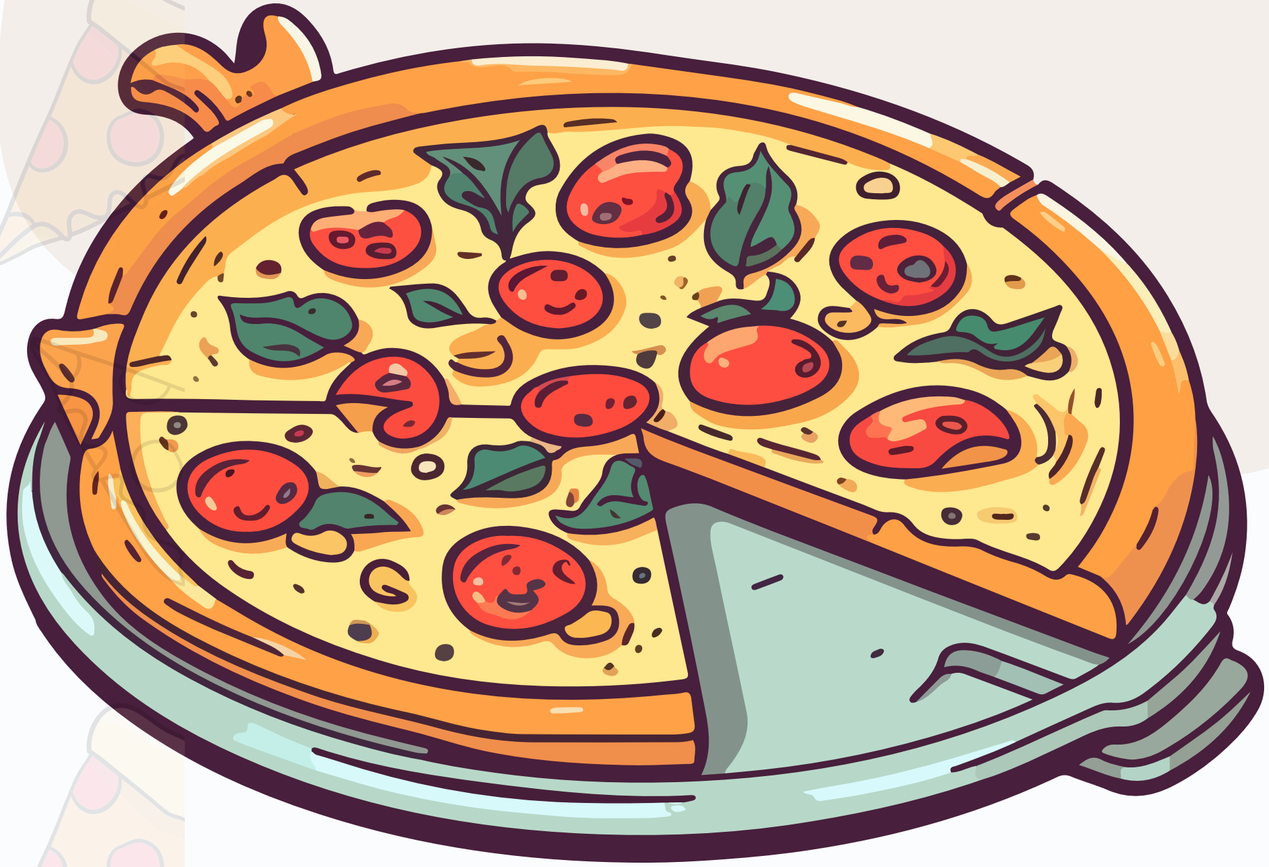
Analyzing Pizza Sales Data with Real-World Queries

A hands-on **SQL Project** exploring insights from a pizza sales dataset



Advit Tiple

Why Pizza Sales Data?

Analyzing pizza sales data offers a great opportunity to apply SQL in solving real-world business problems. Understanding sales performance, customer preferences, and revenue patterns is critical for any business aiming to improve profitability and operational efficiency.

By answering questions such as the total number of orders, the most common pizza sizes, and category-wise revenue contributions, this project mimics the type of analysis businesses need to make data-driven decisions.

The goal is to demonstrate how SQL can be used not just for retrieving data but for generating actionable insights that help businesses optimize their offerings, identify trends, and ultimately enhance decision-making.

Table of Contents

Key Queries Covered in This Project

01

Sales Performance Analysis

Identifying sales trends, evaluating team performance, and comparing historical sales data to improve forecasting.

02

Customer Behavior Insights

Analyzing customer purchase patterns, preferences, and behaviors to enhance segmentation and improve marketing strategies.

03

Revenue Trends

Monitoring revenue growth, key drivers, profit margins, and providing insights for strategic financial planning.

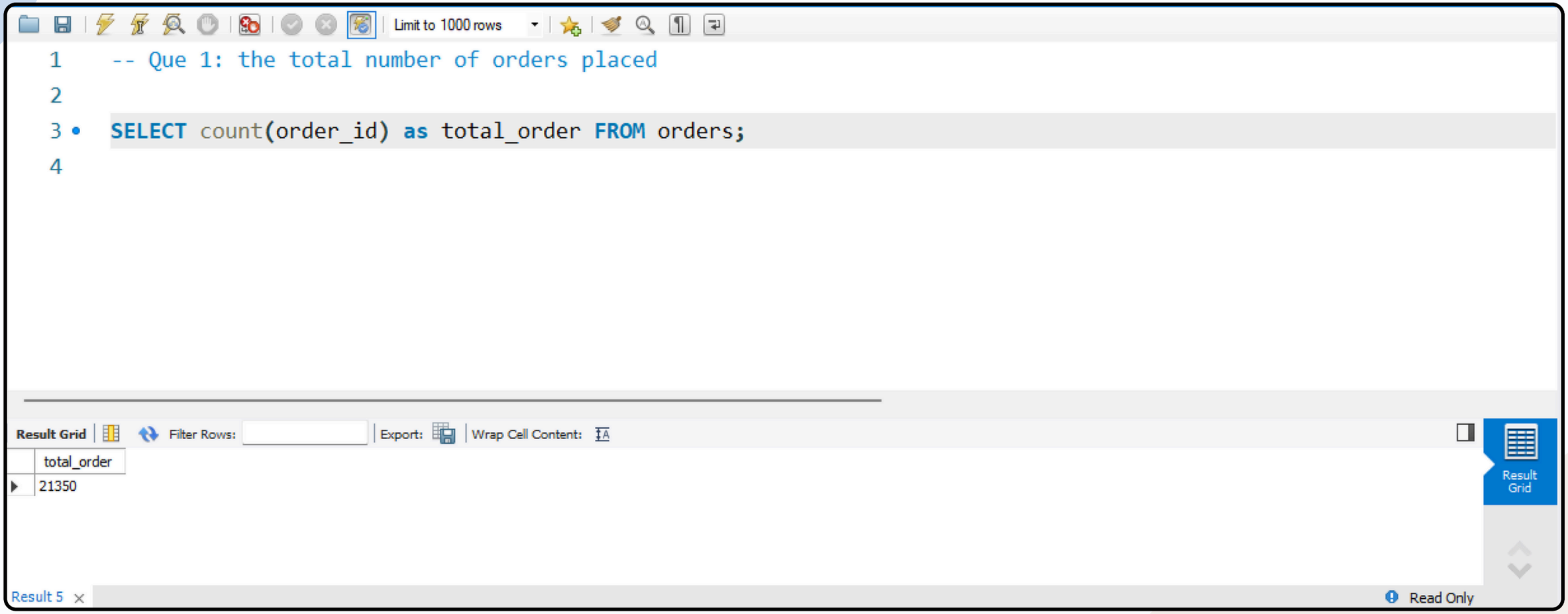
04

Operational Efficiency Metrics

Assessing process efficiency, resource utilization, and cost-effectiveness to optimize overall business performance.

01. Retrieve the total number of orders placed

This query calculates the total number of orders placed in the dataset to understand the overall sales volume.



02. Calculate the total revenue generated from pizza sales

This query sums up the total revenue by multiplying the price of each pizza by the quantity sold, providing insights into the business's total earnings.

Que 1 | Que 2

Limit to 1000 rows

```sql
-- Que 2: Calculate the total revenue generated from pizza sales.

SELECT
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS total_revenue
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id
```

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

total_revenue
817860.05

03. Identify the highest-priced pizza

This query retrieves the highest-priced pizza from the menu, helping the business identify premium offerings.

Que 1 | Que 2 | Que 3

Limit to 1000 rows

```sql
-- Que 3: Identify the highest-priced pizza.

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

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

name	price
The Greek Pizza	35.95

04. Identify the most common pizza size ordered

This query counts the number of orders by pizza size to find out which size is most frequently ordered by customers.

Que 1 | Que 2 | Que 3 | Que 4

Limit to 1000 rows

```sql
-- Que 4: Identify the most common pizza size ordered.
SELECT
    pizzas.size,
    COUNT(order_details.order_details_id) AS order_count
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC;
```

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

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

Result 2

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

This query ranks the top 5 pizzas based on the total quantity ordered, helping the business focus on its best-selling items

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

SELECT
    pizza_types.name, SUM(order_details.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_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

06. Find the total quantity of each pizza category ordered

This query joins the necessary tables to aggregate the total quantity of each pizza category, such as vegetarian or non-vegetarian pizzas

```sql
-- Que 6: Join the necessary tables to find the total quantity of each pizza category ordered.

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

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

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

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

This query analyzes the time of day when orders were placed, helping the business identify peak hours for sales

```
1   -- Que 7: Determine the distribution of orders by hour of the day.
2
3   SELECT hour(order_time) AS hour, count(order_id) AS order_count
4   FROM orders
5   GROUP BY hour;
6
```

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

hour	order_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
9	1

Result 4 | Read Only

08. Find the category-wise distribution of pizzas

This query joins relevant tables and calculates the distribution of orders for each pizza category, providing deeper insights into customer preferences.

```sql
-- Que 8: Join relevant tables to find the category-wise distribution of pizzas.

SELECT category, count(name) AS pizza_name
FROM pizza_types
GROUP BY category
ORDER BY pizza_name DESC;
```

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

category	pizza_name
Supreme	9
Veggie	9
Classic	8
Chicken	6

Result Grid

Form Editor

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

This query groups orders by date and computes the average number of pizzas ordered per day, helping the business understand daily sales trends

```sql
-- Que 9: Group the orders by date and calculate the average number of pizzas ordered per day.

SELECT
    ROUND(AVG(quantity), 0) AS avg_pizza_ordered_per_day
FROM
    (SELECT
        orders.order_date, SUM(order_details.quantity) AS quantity
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY orders.order_date) AS order_quanity;
```

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

avg_pizza_ordered_per_day
138

Result Grid

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

This query identifies the top 3 pizza types that generated the highest revenue, offering insights into the most profitable items.

```sql
-- Que 10: Determine the top 3 most ordered pizza types based on revenue.

SELECT
    pizza_types.name,
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```

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

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

Result Grid

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

This query calculates the percentage contribution of each pizza type to the overall revenue, helping the business assess the importance of each product in driving sales.

```sql
-- Que 11: Calculate the percentage contribution of each pizza type to total revenue.

SELECT
    pizza_types.category,
    ROUND((SUM(order_details.quantity * pizzas.price) / (SELECT
                    ROUND(SUM(order_details.quantity * pizzas.price),
                                2) AS total_revenue
                FROM
                    order_details
                        JOIN
                    pizzas ON pizzas.pizza_id = order_details.pizza_id)) * 100,
            2) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_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	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

Result Grid

12. Analyze the cumulative revenue generated over time

This query tracks cumulative revenue over time to show how sales have grown, providing valuable insights for trend analysis.

```sql
-- Que 12: Analyze the cumulative revenue generated over time.

SELECT
    order_date,
    sum(revenue) over (ORDER BY order_date) as cumu_revenue
FROM

(SELECT
    orders.order_date,
    SUM(order_details.quantity * pizzas.price) AS revenue
FROM
    order_details
        JOIN
    pizzas ON order_details.pizza_id = pizzas.pizza_id
        JOIN
    orders ON orders.order_id = order_details.order_id
GROUP BY orders.order_date) As sales;
```

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

order_date	cumu_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

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

This query identifies the top 3 pizza types with the highest revenue in each category, offering a more granular view of profitability across different product lines.

```sql
-- Que 13: Determine the top 3 most ordered pizza types based on revenue for each pizza category.

SELECT
    category, 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(order_details.quantity * pizzas.price) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_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:

category	name	revenue
Chicken	The Thai Chicken Pizza	43434.25
Chicken	The Barbecue Chicken Pizza	42768
Chicken	The California Chicken Pizza	41409.5
Classic	The Classic Deluxe Pizza	38180.5
Classic	The Hawaiian Pizza	32273.25
Classic	The Pepperoni Pizza	30161.75

Result Grid | Form Editor

Challenges Faced & Lessons Learned

Overcoming complex queries and unlocking key business insights through hands-on SQL practice

01

Handling Data Complexity

Joining multiple tables and managing relationships accurately was crucial for category-wise and revenue-based queries. This improved my understanding of complex joins and table structures.

02

Managing Aggregations

Writing queries with GROUP BY and HAVING clauses for meaningful insights taught me how to handle advanced aggregations effectively.

Challenges Faced & Lessons Learned

Overcoming complex queries and unlocking key business insights through hands-on SQL practice

03

Optimizing Query Performance

Ensuring that queries ran efficiently on large datasets helped me learn optimization techniques and best practices for performance tuning.

04

Date and Time Analysis

Working with time-based queries, such as analyzing order distribution by hour and cumulative revenue over time, enhanced my skills in using SQL date and time functions.

05

Deriving Actionable Insights:

Interpreting the query results and translating them into meaningful business insights, such as identifying peak sales periods and high-revenue products, strengthened my analytical thinking.

Conclusion

This project provided a hands-on experience in applying SQL to a real-world dataset, enhancing both my technical and analytical skills. By working through various levels of complexity—from basic metrics to advanced revenue analysis—I gained a deeper understanding of SQL's powerful role in data-driven decision-making. It also reinforced the importance of clean, efficient query writing and interpreting results to derive actionable insights.

Overall, this project not only improved my proficiency in SQL techniques such as joins, aggregations, and time-based analysis but also demonstrated how data analysis can help businesses optimize their operations, identify trends, and maximize profitability.

Thank you

advittiple

advittiple.connect@gmail.com

advittiple