Pizza Sales Analysis Using SQL

Unlocking Insights from Pizza Sales Data Through SQL Queries

By : Parshwa Jain

❖ Introduction

- In this project, we dive into the realm of pizza sales data to uncover valuable insights using SQL queries.
- Our dataset comprises detailed sales information, including various types of pizzas, customer preferences, and sales trends.
- By applying SQL techniques, we aim to address specific business questions and provide actionable insights that can help in strategic decision-making.
- This presentation will guide you through the process, highlighting key findings and demonstrating the power of SQL in data analysis.

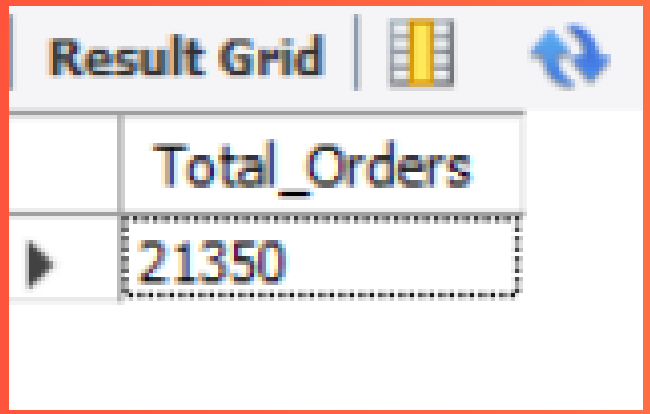
Source : GitHub Link

Problem 01 : Retrieve the total number of orders place.

- **Query :**

```
1    -- Problem 01 : Retrieve the total number of orders place.
2
3    select count(order_id) as Total_Orders from orders
```

- **Result :**



Problem 02 : Calculate the total revenue generated from pizza sales.

- **Query :**

```sql
-- Problem 02 : Calculate the total revenue generated from pizza sales.

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

- **Result :**

Result Grid

total_bill
817860.05

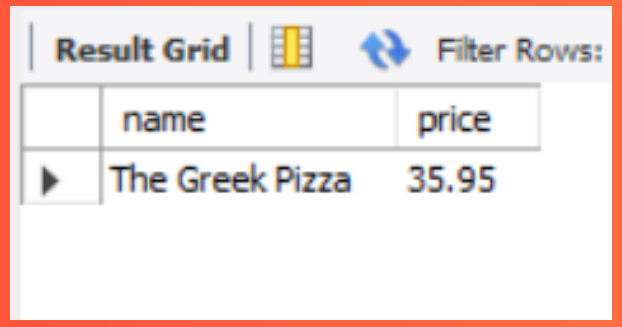
Problem 03 : Identify the highest-priced pizza.

❑ **Query :**

```sql
-- Problem 03 : 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 :**



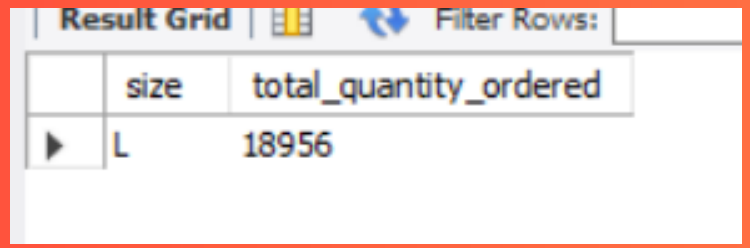
Problem 04 : Identify the most common pizza size ordered.

❑ **Query :**

```sql
-- Problem 04 : Identify the most common pizza size ordered.

SELECT
    pizzas.size,
    SUM(order_details.quantity) AS total_quantity_ordered
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY total_quantity_ordered DESC
LIMIT 1
```

➢ **Result :**



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

❑ **Query :**

```sql
-- Problem 05 : 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 pizzas.pizza_id = order_details.pizza_id
GROUP BY pizza_types.name
ORDER BY quantity DESC
LIMIT 5
```

➢ **Result :**

Result Grid | Filter 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

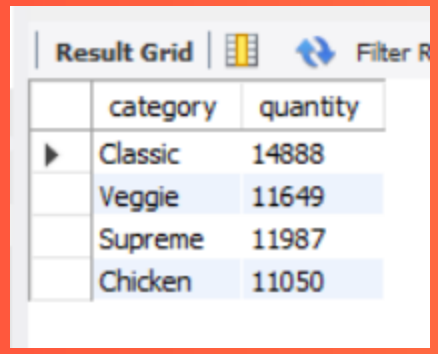
Problem 06 : Join the necessary tables to find the total quantity of each pizza category ordered.

❑ Query :

```sql
-- Problem 06 : Join the necessary tables to find the total quantity of each pizza category ordered.

SELECT
    pizza_types.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 pizzas.pizza_id = order_details.pizza_id
GROUP BY pizza_types.category
```

➢ Result :



category	quantity
Classic	14888
Veggie	11649
Supreme	11987
Chicken	11050

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

❑ Query :

```sql
-- Problem 07 : Determine the distribution of orders by hour of the day.

SELECT
    HOUR(order_time) AS hour_of_day,
    COUNT(order_id) AS total_orders
FROM
    orders
GROUP BY hour_of_day
ORDER BY hour_of_day ASC
```

➢ Result :

Result Grid | Filter Rows:

hour_of_day	total_orders
9	1
10	8
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

Problem 08 : Join relevant tables to find the category-wise distribution of pizzas.

❑ Query :

```sql
-- Problem 08 : Join relevant tables to find the category-wise distribution of pizzas.

SELECT
    category, COUNT(category) AS total_pizza
FROM
    pizza_types
GROUP BY category
ORDER BY total_pizza DESC
```

➢ Result :

Result Grid | Filter Ro

category	total_pizza
Supreme	9
Veggie	9
Classic	8
Chicken	6

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

❑ Query :

```sql
-- Problem 09 : Group the orders by date and calculate the average number of pizzas ordered per day.

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

➢ Result :

Result Grid | Filter Rows:

Avg_Pizza_Order_per_day
138

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

- **Query :**

```sql
-- Problem 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
    order_details
        JOIN
    pizzas ON order_details.pizza_id = pizzas.pizza_id
        JOIN
    pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3
```

- **Result :**

Result Grid | Filter Rows:

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

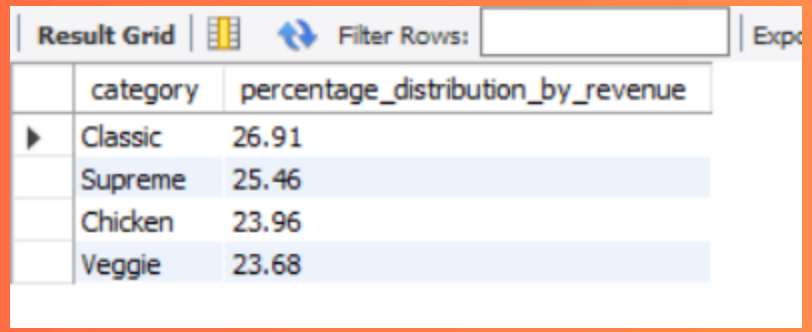
Problem 11 : Calculate the percentage contribution of each pizza type to total revenue.

❑ Query :

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

SELECT
    pizza_types.category,
    ROUND(((SUM(order_details.quantity * pizzas.price)) / (SELECT
                    SUM((order_details.quantity * pizzas.price)) AS total_bill
                FROM
                    order_details
                        JOIN
                    pizzas ON order_details.pizza_id = pizzas.pizza_id) * 100),
            2) AS percentage_distribution_by_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 percentage_distribution_by_revenue DESC
```

➢ Result :


Result Grid | Filter Rows: | Expo

category	percentage_distribution_by_revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

Problem 12 : Analyze the cumulative revenue generated over time.

❑ Query :

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

select
order_date,
round(sum(revenue) over(order by order_date),2) as cumulative_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 :

order_date	cumulative_revenue
2015-01-01	2713.85
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	23990.35
2015-01-11	25862.65
2015-01-12	27781.7
2015-01-13	29831.3
2015-01-14	32358.7
2015-01-15	34343.5
2015-01-16	36937.65
2015-01-17	39001.75
2015-01-18	40978.6

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

❑ Query :

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

select category, name, round(revenue,2) as Total_revenue_by_pizza,rn
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 pizzas.pizza_id = order_details.pizza_id
group by pizza_types.category,pizza_types.name) as a) as b
where rn <= 3
```

➢ Result :

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

category	name	Total_revenue_by_pizza	rn
Chicken	The Thai Chicken Pizza	43434.25	1
Chicken	The Barbecue Chicken Pizza	42768	2
Chicken	The California Chicken Pizza	41409.5	3
Classic	The Classic Deluxe Pizza	38180.5	1
Classic	The Hawaiian Pizza	32273.25	2
Classic	The Pepperoni Pizza	30161.75	3
Supreme	The Spicy Italian Pizza	34831.25	1
Supreme	The Italian Supreme Pizza	33476.75	2
Supreme	The Sicilian Pizza	30940.5	3
Veggie	The Four Cheese Pizza	32265.7	1
Veggie	The Mexicana Pizza	26780.75	2
Veggie	The Five Cheese Pizza	26066.5	3

❖ Conclusion

- ✓ The analysis of pizza sales data using SQL has provided us with a comprehensive understanding of the business's performance and customer preferences. Through basic, intermediate, and advanced queries, we were able to uncover significant insights that can guide strategic decisions:

Total Orders and Revenue: We quantified the total number of orders placed and the total revenue generated, establishing a baseline for sales performance.

Product Insights: By identifying the highest-priced pizza and the most common pizza size ordered, we gained insights into pricing strategies and customer preferences.

Top Performing Pizzas: Listing the top 5 most ordered pizza types along with their quantities highlighted the best-sellers and potential focus areas for promotions.

❖ Conclusion

Category and Temporal Analysis: Joining tables to find the total quantity of each pizza category ordered and the distribution of orders by hour of the day provided a deeper understanding of sales patterns.

Revenue Insights: Grouping orders by date and determining the top 3 most ordered pizza types based on revenue helped identify peak sales periods and high-performing products.

Advanced Metrics: Calculating the percentage contribution of each pizza type to total revenue and analyzing cumulative revenue over time offered a detailed view of financial performance.

Category-wise Performance: Identifying the top 3 most ordered pizza types based on revenue for each category provided actionable insights for targeted marketing and inventory management.

THANK YOU !