ANNSHUL SAJWAN

Email-Annshulsajwan28@gmail.com

Linkdin-www.linkedin.com/in/annshulsajwan

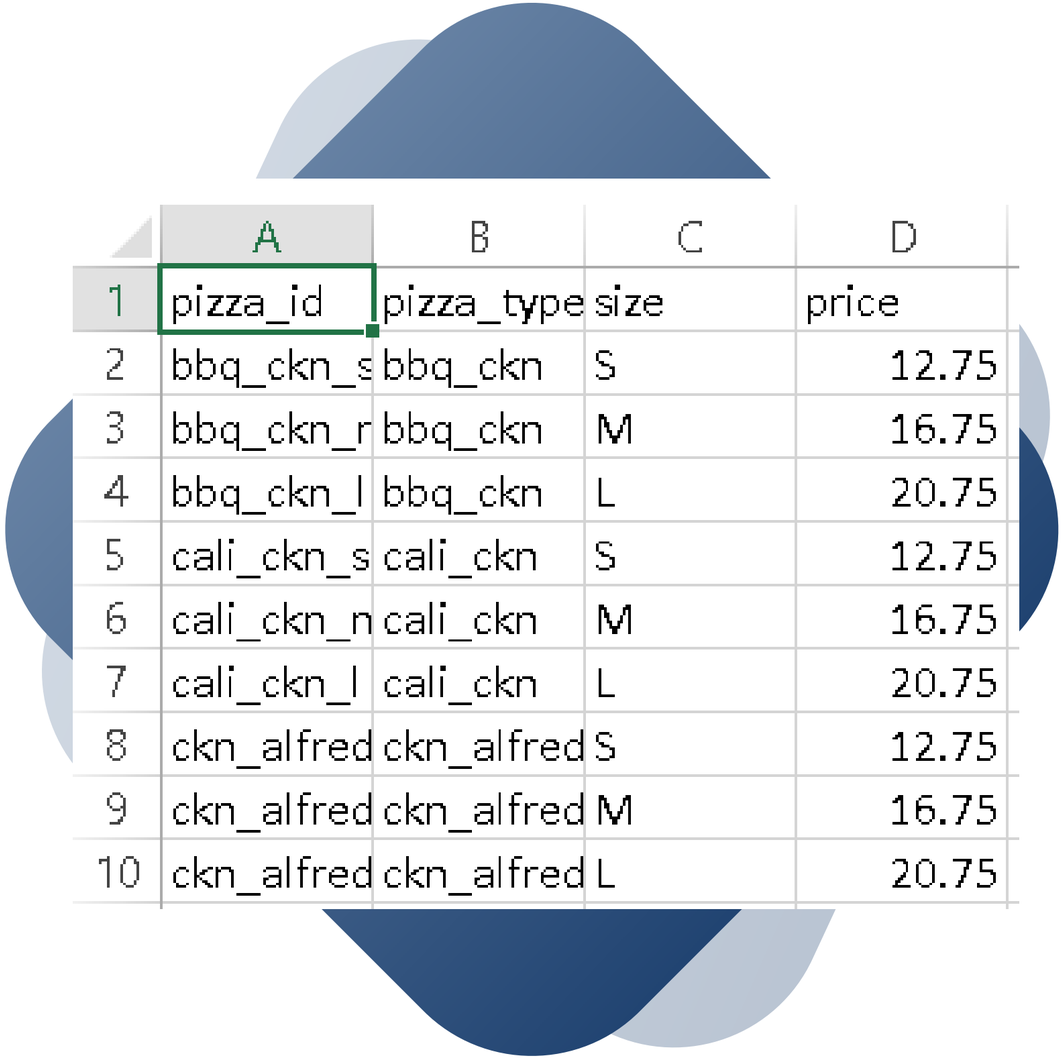
SQL Sales Analysis And Optimization

Introduction

In this project, we analyze pizza orders using SQL queries to gain insights into customer preferences, sales performance, and operational trends. By examining a dataset, we apply various SQL techniques—from basic aggregations to advanced window functions—to address real-world business challenges. The goal is to offer actionable insights for improving menu options, pricing strategies, and inventory management.

Datasets Used

- Orders (orders.csv)
- Order Details (order_details.csv)
- Pizza Types (pizza_types.csv)
- Pizzas (pizzas.csv)



	A	B	C	D
1	pizza_id	pizza_type	size	price
2	bbq_ckn_s	bbq_ckn	S	12.75
3	bbq_ckn_r	bbq_ckn	M	16.75
4	bbq_ckn_l	bbq_ckn	L	20.75
5	cali_ckn_s	cali_ckn	S	12.75
6	cali_ckn_n	cali_ckn	M	16.75
7	cali_ckn_l	cali_ckn	L	20.75
8	ckn_alfred	ckn_alfred	S	12.75
9	ckn_alfred	ckn_alfred	M	16.75
10	ckn_alfred	ckn_alfred	L	20.75

All files will be accessible in my GitHub repository.

Approach

- I created a new database named dominos to organize and store the data.
- Within the dominos database, I established key tables
- I imported data from four CSV files into these tables to build a comprehensive dataset for analysis.

Database creation

```
create database dominos;
use dominos;
```

Table Creation

```
create table orders (
order_id int not null ,
 order_date date not null ,
 order_time time not null ,
 primary key(order_id) );

 create table order_details (
order_details_id int not null ,
order_id int not null,
pizza_id text not null ,
 quantity int not null ,
 primary key(order_details_id) );
```

How many total orders were placed?

Explanation: The query counts the total number of orders placed in the orders table using the COUNT() function.

Insight: The total number of orders placed are 21,350.

Query:

```
2 •    SELECT
3          COUNT(order_id) AS total_orders
4      FROM
5          orders;
6
```

Output:

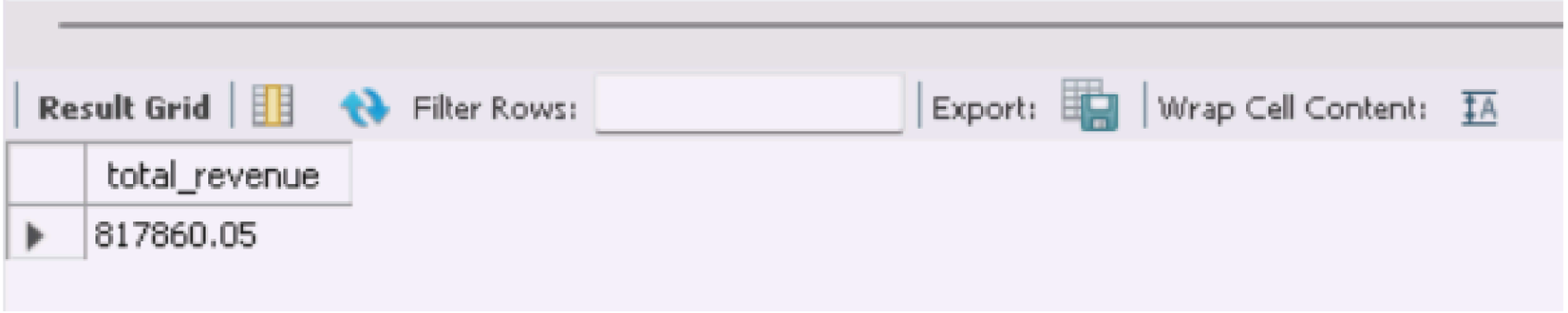
What is the total revenue generated from pizza sales?

Explanation: This query calculates the total revenue generated by multiplying the quantity of pizzas sold by their price.
Insight: The total revenue generated from pizza sales is $817,860 .

Query:

```
2 •   SELECT
3         ROUND(SUM(order_details.quantity * pizzas.price),
4                 2) AS total_revenue
5     FROM
6         order_details
7             JOIN
8         pizzas ON order_details.pizza_id = pizzas.pizza_id;
```

Output:



total_revenue
817860.05

Which pizza has the highest price?

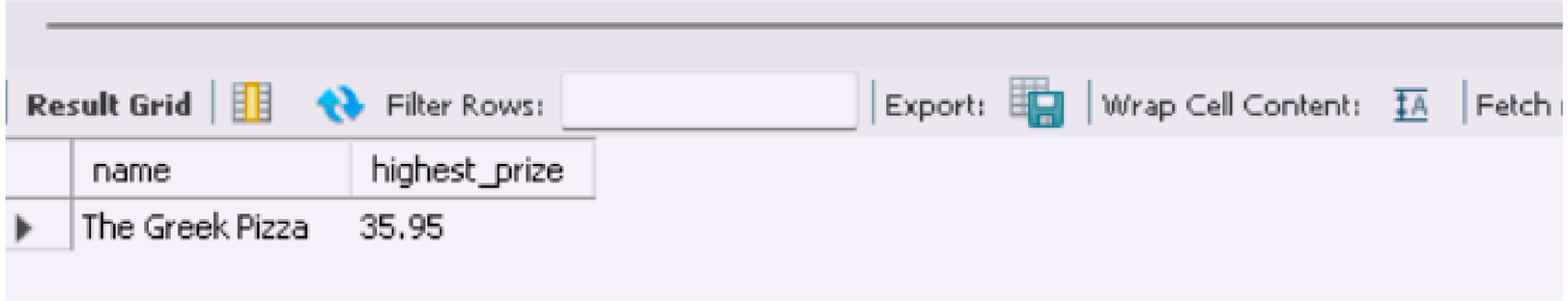
Explanation: I identified the highest-priced pizza by joining the pizza_types table with the pizzas table and ordering the results in descending order of price. The LIMIT 1 clause was used to return only the top entry, which is the highest-priced pizza.

Insight: The highest-priced pizza is The Greek Pizza, priced at $35.95

Query:

```
2 •   SELECT
3         pizza_types.name, (pizzas.price) AS highest_prize
4     FROM
5         pizza_types
6             JOIN
7         pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
8     ORDER BY pizzas.price DESC
9     LIMIT 1;
```

Output:



What is the most common pizza size ordered?

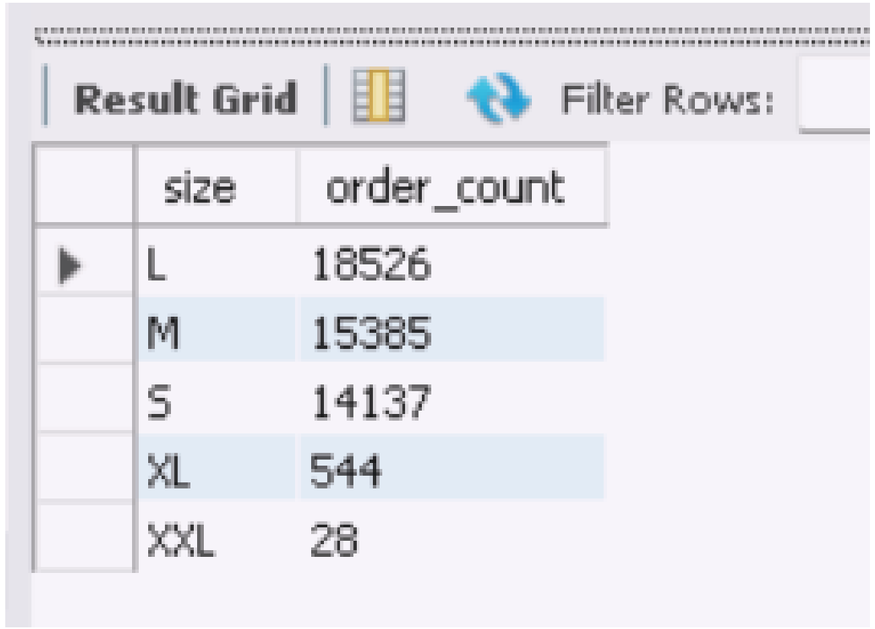
Explanation: To find the most common pizza size, I joined the pizzas and order_details tables. I then grouped the data by pizza size and used the COUNT() function to determine the size that appears most frequently in the orders.

Insight: The most common pizza size ordered is Large followed by Meduim and then Small.

Query:

```
2 •   SELECT
3         pizzas.size,
4         COUNT(order_details.order_details_id) AS order_count
5     FROM
6         pizzas
7             JOIN
8         order_details ON pizzas.pizza_id = order_details.pizza_id
9     GROUP BY pizzas.size
10    ORDER BY order_count DESC ;
```

Output:



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

What are the top 5 most ordered pizza types, and what are their quantities?

Explanation: I determined the top 5 most ordered pizza types by joining the pizza_types, pizzas, and order_details tables. I summed up the quantities for each pizza type and ordered the results in descending order, using the LIMIT 5 clause to get the top 5 entries.

Insight: The top most ordered pizza type is The Classic Delux Pizza with over 2.4k orders.

Query:

```
4  •  SELECT
5         pizza_types.name,
6         SUM(order_details.quantity) AS total_quantity
7     FROM
8         pizza_types
9             JOIN
10        pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
11            JOIN
12        order_details ON order_details.pizza_id = pizzas.pizza_id
13    GROUP BY pizza_types.name
14    ORDER BY total_quantity DESC
15    LIMIT 5;
```

Output:

Result Grid | Filter Rows:

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

What is the total quantity of each pizza category ordered?

Explanation: By joining the pizza_types, pizzas, and order_details tables, I was able to group the results by pizza category and sum the total quantities for each category. This provided a clear picture of how many pizzas were ordered from each category.

Insight: The total quantity ordered for each pizza category is Classic with 14.8k orders, Supreme with 11.9k orders, and so forth.

Query:

```
3 •   SELECT DISTINCT
4         (pizza_types.category),
5         SUM(order_details.quantity) AS total_quantity
6     FROM
7         pizza_types
8             JOIN
9         pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
10            JOIN
11        order_details ON order_details.pizza_id = pizzas.pizza_id
12    GROUP BY pizza_types.category
13    ORDER BY total_quantity DESC;
```

Output:

Result Grid | Filter Rows:

	category	total_quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

What is the distribution of pizzas by category ?

Explanation: I used a GROUP BY clause on the category field in the pizza_types table to count how many pizzas belong to each category. This query helped in understanding the popularity of different pizza categories.

Insight: The distribution of pizzas by category shows Chicken with 6 pizzas, Classic with pizzas, and so on.

Query:

```
3
4 •    select category,count(name) as name from pizza_types
5      group by category ;
```

Output:

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

category	name
Chicken	6
Classic	8
Supreme	9
Veggie	9

What is the average number of pizzas ordered per day when grouped by date?

Explanation: To find the average number of pizzas ordered per day, I first grouped the orders by date and summed the quantities. Then, I used the AVG() function to calculate the average number of pizzas ordered daily.

Insight: The average number of pizzas ordered per day is 138.

Query:

```
3  •  SELECT
4         ROUND(AVG(quantity), 0) AS avg_pizza_perday
5     FROM
6         (SELECT
7             orders.order_date, SUM(order_details.quantity) AS quantity
8         FROM
9             orders
10        JOIN order_details ON orders.order_id = order_details.order_id
11        GROUP BY orders.order_date) AS order_quantity;
```

Output:

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

avg_pizza_perday
138

Which are the top 3 most ordered pizza types based on revenue?

Explanation: I calculated the total revenue generated by each pizza type. I then ordered the results by revenue in descending order and used the LIMIT 3 clause to identify the top 3 revenue-generating pizza types.

Insight: The top 3 most ordered pizza types by revenue are The Thai Chicken Pizza with $43434 revenue followed by Barbecue and California chicken pizza.

Query:

```
7  •  select pizza_types.name , ROUND(SUM(order_details.quantity * pizzas.price),2)
8     AS total_revenue from pizza_types join pizzas on
9     pizzas.pizza_type_id = pizza_types.pizza_type_id join order_details
10    on order_details.pizza_id = pizzas.pizza_id
11    group by pizza_types.name order by total_revenue desc limit 3;
```

Output:

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

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

What is the percentage contribution of each pizza type to the total revenue?

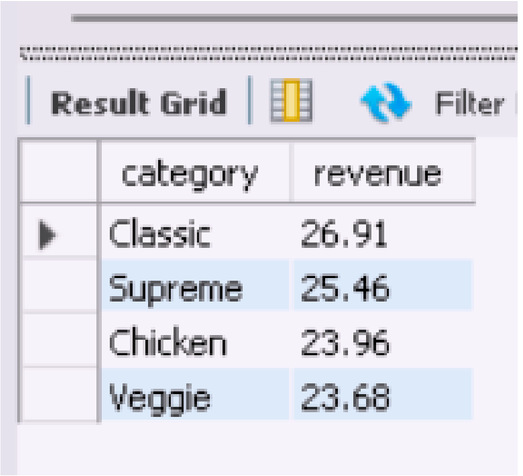
Explanation: To determine the percentage contribution of each pizza type, I calculated the total revenue for each type and then divided it by the overall revenue.

Insight: The percentage contribution to total revenue is 26.91% for Classic, 25.46% for Supreme, and so on.

Query:

```
3   SELECT
4       pizza_types.category,
5       ROUND(SUM(pizzas.price * order_details.quantity) / (SELECT
6                       ROUND(SUM(order_details.quantity * pizzas.price),
7                                   2) AS total_revenue
8                   FROM
9                       order_details
10                          JOIN
11                      pizzas ON order_details.pizza_id = pizzas.pizza_id) * 100,
12              2) AS revenue
13  FROM
14      pizza_types
15          JOIN
16      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
17          JOIN
18      order_details ON order_details.pizza_id = pizzas.pizza_id
19  GROUP BY pizza_types.category
20  ORDER BY revenue DESC;
```

Output:



	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

How does the cumulative revenue generated evolve over time?

Explanation: The query calculates cumulative revenue by summing daily revenues from pizza sales. A window function is used to accumulate these sums over time, showing consistent revenue growth throughout the period.

Insight: The cumulative revenue generated over time shows a steady increase, starting from $2,713.85 on 2015-01-01 and reaching $32,358.70 by 2015-01-14 and so on.

Query:

```
4  •  select order_date , sum(revenue) over(order by order_date) as cumulative_revenue from
5     (select orders.order_date , sum(order_details.quantity*pizzas.price)
6     as revenue from order_details join
7     pizzas on pizzas.pizza_id = order_details.pizza_id join orders on
8     orders.order_id = order_details.order_id
9     group by orders.order_date ) as sales;
10
```

Output:

Result Grid | Filter Rows:

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

Summary

- The analysis provides a detailed understanding of customer ordering patterns, pizza pricing, and revenue generation.
- We identified the most popular pizzas, their sizes, and categories, and how these factors contribute to the overall sales and revenue.
- This data-driven approach reveals opportunities to optimize menu offerings, focus on high-revenue items, and better understand customer preferences.

Key Recommendations

Pumping up those sales and cash flow!

- Focus marketing efforts on Large and Medium pizzas, offering deals or bundles to drive sales of these popular sizes.
- Feature The Thai Chicken Pizza, Barbecue, and California Chicken Pizza prominently in promotions to capitalize on their strong revenue performance.
- Enhance menu offerings and promotions around the Classic and Supreme pizza categories, which have the highest order volumes and revenue contributions.

Conclusion

- The project successfully analyzed key metrics such as total orders, revenue, and product popularity, offering actionable insights.
- The findings can be leveraged to improve business strategies, such as marketing high-revenue pizzas and optimizing inventory based on demand.
- Future analyses could include customer demographics and seasonal trends to further refine marketing and operational decisions.