DOMINO'S SHOP ANALYSIS WITH SQL

1

PROJECT INTRODUCTION

2

PROBLEM ANALYSIS

3

PLANNING

4

SOLUTION

5

REPORT

Introduction to Domino's Pizza SQL Project

- Welcome to my SQL project, where I will be analyzing and exploring the world of Domino's Pizza through data analysis. As part of this project, I will be following a structured roadmap that includes problem analysis, planning, solutions, and finally, a comprehensive report.

Background

- Domino's Pizza is a multinational pizza restaurant chain that operates in over 85 countries worldwide. With a vast customer base and a wide range of menu items, Domino's generates a significant amount of data that can be leveraged to gain insights into customer behavior, sales trends, and operational efficiency.

Objective

- The primary objective of this project is to analyze the Domino's Pizza database using SQL queries to extract meaningful insights and trends. Through this analysis, I aim to identify areas of improvement, opportunities for growth, and optimize business processes to increase customer satisfaction and revenue

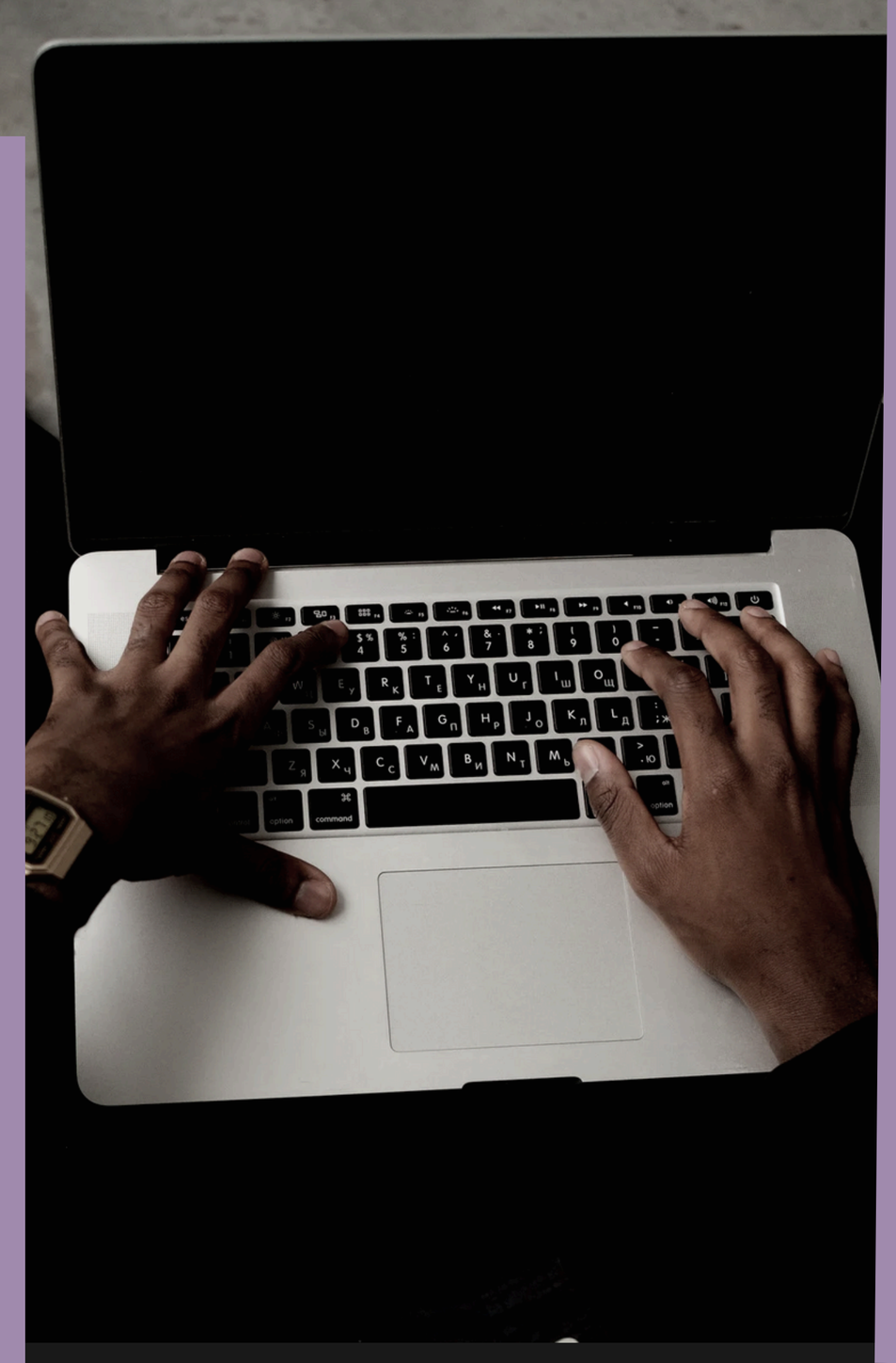
Methodology

to achieve the objectives of this project, I will be following a structured approach that includes:

1. Problem Analysis: Identifying key business problems and opportunities for improvement
2. Planning: Developing a comprehensive plan for data analysis, including data extraction, transformation, and loading
3. Solutions: Designing and executing SQL queries to extract insights and trends from the Domino's Pizza database
4. Report: Compiling and presenting findings, recommendations, and insights in a clear and concise manner

Expected Outcomes

- Data-driven insights into customer behavior and preferences
- Identification of areas for operational improvement and cost reduction
- Development of targeted marketing strategies and promotions
- Recommendations for menu optimization and pricing strategies



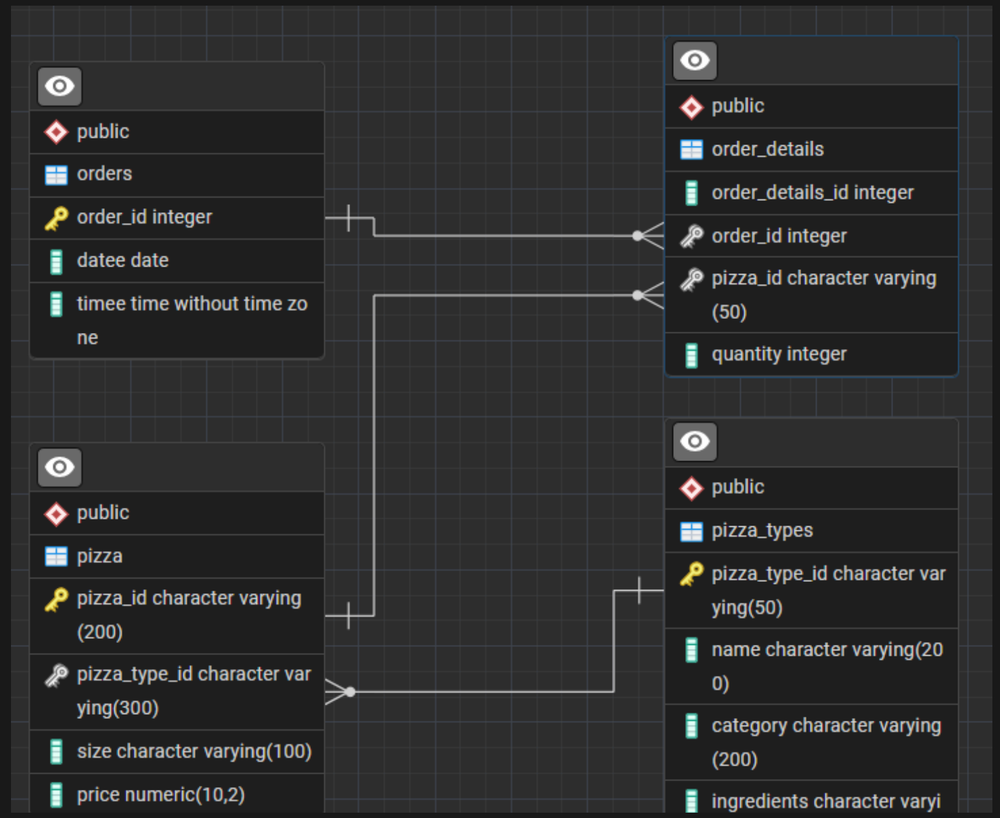
PROBLEMS

1) Retrieve the total number of orders placed.
2). Calculate the total revenue generated from pizza sales.
3). Identify the highest-priced pizza.
4). Identify the most common pizza size ordered.
5). List the top 5 most ordered pizza types along with their quantities.
6). Join the necessary tables to find the total quantity of each pizza category ordered.
7). Determine the distribution of orders by hour of the day.
8). Join relevant tables to find the category-wise distribution of pizzas.
9). Group the orders by date and calculate the average number of pizzas ordered per day and total reveneu per month with cumulative sum
10). Determine the top 3 most ordered pizza types based on revenue.
11). Calculate the percentage contribution of each pizza type to total revenue.
12). Analyze the cumulative revenue generated over time.
13). Determine the top 3 most ordered pizza types based on revenue for each pizza category.

ERD DIAGRAM

{Orders table}
order_id
date
time

{Order_details table}
order_details_id
order_id
pizza_id
quantity



{Pizza table}
order_details
pizza_id
pizza_type_id
price

{Pizza_types table}
pizza_type_id
name
category
ingridient

Retrieving the Total Number of Orders Placed

- The count(*) function returns the count of all rows in the table, including rows with null values.
- The * is a wildcard character that represents all columns.
- The FROM clause specifies the table(s) we want to retrieve data from, which in this case is the order_details table.
- The order_details table contains information about each order, such as the order ID, customer ID, order date, and other relevant details.

Query | Query History

```sql
select * from order_details;
select * from orders;
select * from pizza;
select * from pizza_types;


--Retrieve the total number of orders placed.

select count(*)
from order_details;
```

Data Output | Messages | Notifications

	count bigint
1	48620

Retrieving the Most Expensive Pizza

- SELECT pizza_id, price: This selects the pizza_id and price columns from the pizza table.
- FROM pizza: This specifies the table to retrieve data from, which is the pizza table.
- ORDER BY price DESC: This sorts the results in descending order based on the price column, so the most expensive pizza appears first.
- LIMIT 1: This limits the output to only the top 1 result, which is the most expensive pizza.

Query Query History

```
21
22
23  --Identify the highest-priced pizza.
24
25  select pizza_id, price
26  from pizza
27  order by price desc
28  limit 1;
29
30
31
32
33
34
35
```

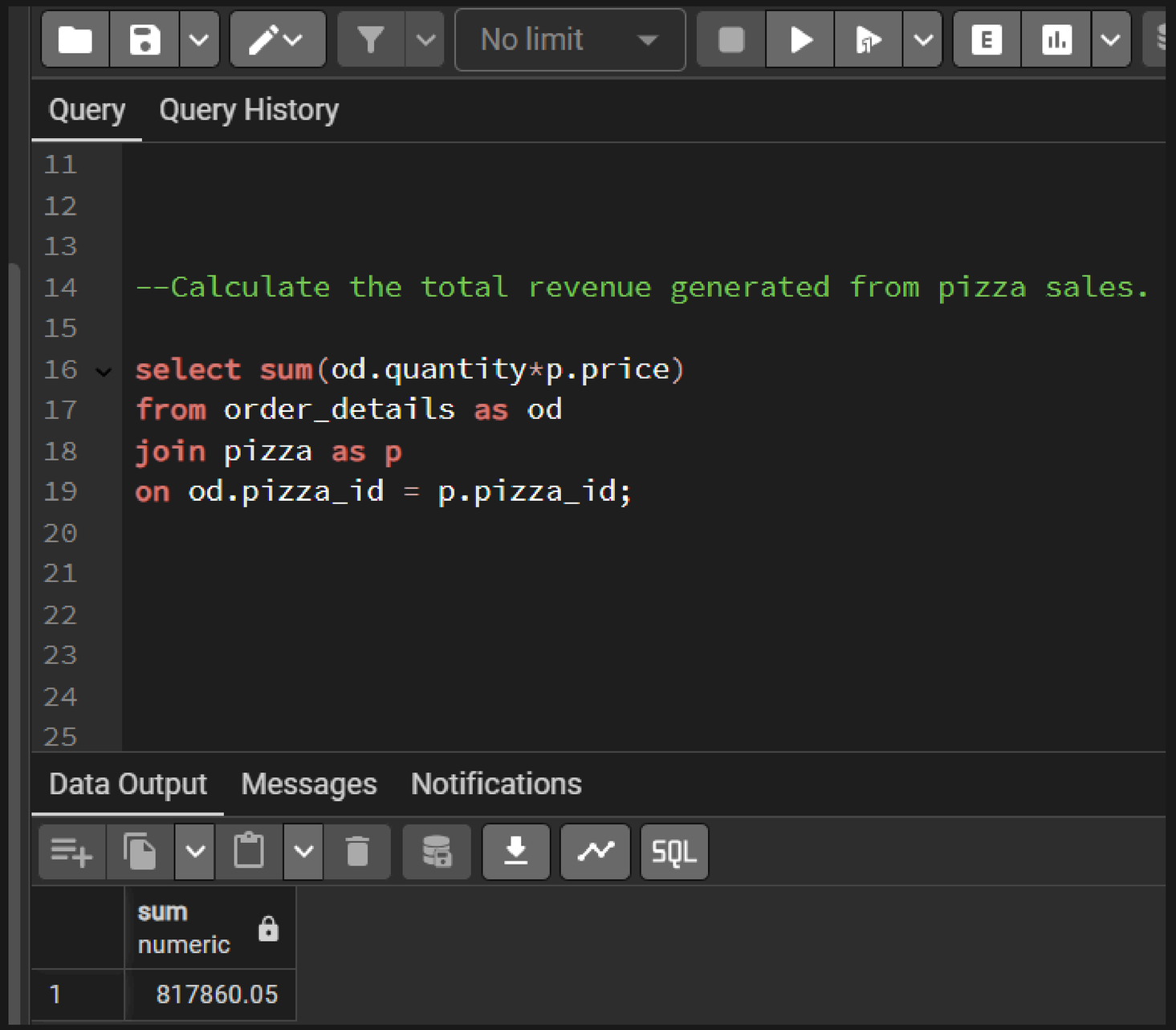
Data Output Messages Notifications

SQL

	pizza_id character varying (200)	price numeric (10,2)
1	the_greek_xxl	35.95

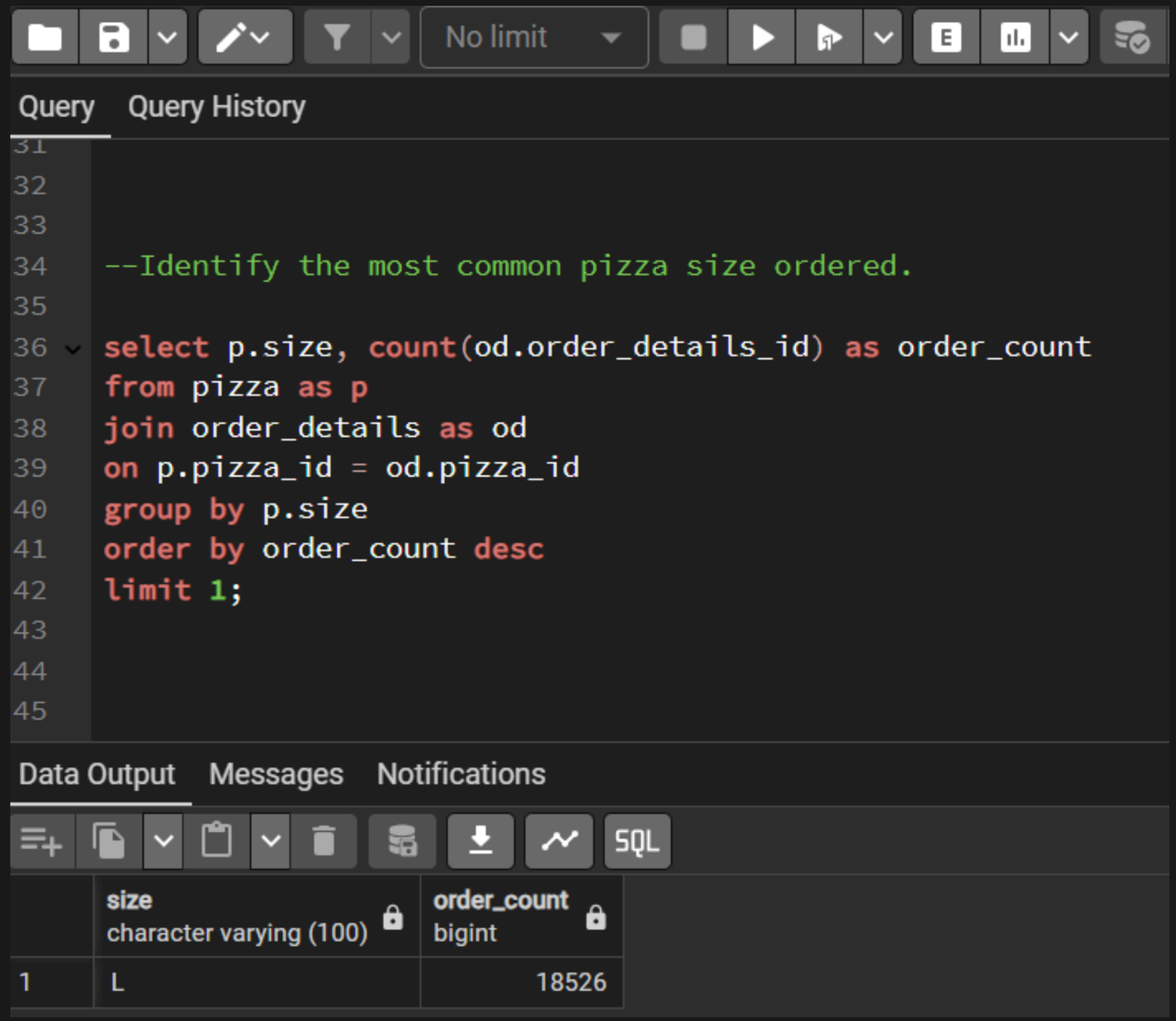
Calculating Total Revenue of Pizza Sales

- SELECT SUM(od.quantity*p.price): This calculates the total revenue by multiplying the quantity of each pizza sold (from the order_details table) with its corresponding price (from the pizza table). The SUM function then adds up these values to give the total revenue.
- FROM order_details as od: This specifies the order_details table as the primary table for the query, and assigns it the alias od.
- JOIN pizza as p: This joins the pizza table to the order_details table, and assigns it the alias p.
- ON od.pizza_id = p.pizza_id: This specifies the join condition, which is that the pizza_id in the order_details table matches the pizza_id in the pizza table.



Identifying the Most Common Pizza Size Ordered

- SELECT p.size, count(od.order_details_id) as order_count: This selects the size column from the pizza table and counts the number of orders for each size using the count function. The AS keyword assigns the alias order_count to the count column.
- FROM pizza as p: This specifies the pizza table as the primary table for the query, and assigns it the alias p.
- JOIN order_details as od: This joins the order_details table to the pizza table, and assigns it the alias od.
- ON p.pizza_id = od.pizza_id: This specifies the join condition, which is that the pizza_id in the pizza table matches the pizza_id in the order_details table.
- GROUP BY p.size: This groups the results by the size column, so that the count of orders is calculated for each unique pizza size.
- ORDER BY order_count DESC: This sorts the results in descending order based on the order_count column, so that the most common pizza size appears first.
- LIMIT 1: This limits the output to only the top 1 result, which is the most common pizza size.



Listing the Top 5 Most Ordered Pizza Types along with their Quantities

- SELECT pt.name, count(od.order_details_id), sum(od.quantity): This selects the name column from the pizza_types table, counts the number of orders for each pizza type, and calculates the total quantity of each pizza type.
- FROM pizza_types as pt: This specifies the pizza_types table as the primary table for the query.
- JOIN pizza as p: This joins the pizza table to the pizza_types table.
- ON pt.pizza_type_id = p.pizza_type_id: This specifies the join condition, which is that the pizza_type_id in the pizza_types table matches the pizza_type_id in the pizza table.
- JOIN order_details as od: This joins the order_details table to the pizza table.
- ON p.pizza_id = od.pizza_id: This specifies the join condition, which is that the pizza_id in the pizza table matches the pizza_id in the order_details table.
- GROUP BY pt.name: This groups the results by the name column, so that the count and sum are calculated for each unique pizza type.
-

```
6
7   --List the top 5 most ordered pizza types along with their quantities.
8
9   select pt.name,
10  count(od.order_details_id),
11  sum(od.quantity)
12  from pizza_types as pt
13  join pizza as p
14  on pt.pizza_type_id = p.pizza_type_id
15  join order_details as od
16  on p.pizza_id = od.pizza_id
17  group by pt.name
```

Data Output | Messages | Notifications

SQL

	name character varying (200)	count bigint	sum bigint
1	The Italian Vegetables Pizza	975	981
2	The Napolitana Pizza	1451	1464
3	The Thai Chicken Pizza	2315	2371
4	The Four Cheese Pizza	1850	1902
5	The Hawaiian Pizza	2370	2422
6	The Green Garden Pizza	987	997
7	The Barbecue Chicken Pizza	2372	2432
8	The Southwest Chicken Pizza	1885	1917
9	The Pepperoni, Mushroom, and Peppers Pizza	1342	1359

Finding the Total Quantity of Each Pizza Category Ordered

- SELECT pt.category, count(od.order_details_id) total_orders: This selects the category column from the pizza_types table and counts the number of orders for each category.
- FROM pizza_types as pt: This specifies the pizza_types table as the primary table for the query.
- JOIN pizza as p: This joins the pizza table to the pizza_types table.
- ON pt.pizza_type_id = p.pizza_type_id: This specifies the join condition, which is that the pizza_type_id in the pizza_types table matches the pizza_type_id in the pizza table.
- JOIN order_details as od: This joins the order_details table to the pizza table.
- ON p.pizza_id = od.pizza_id: This specifies the join condition, which is that the pizza_id in the pizza table matches the pizza_id in the order_details table.
- GROUP BY pt.category: This groups the results by the category column, so that the count is calculated for each unique pizza category.

Query History

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

select pt.category, count(od.order_details_id) total_orders
from pizza_types as pt
join pizza as p
on pt.pizza_type_id = p.pizza_type_id
join order_details as od
on p.pizza_id = od.pizza_id
group by pt.category;
```

SQL

	category character varying (200)	total_orders bigint
1	Veggie	11449
2	Chicken	10815
3	Supreme	11777
4	Classic	14579

This query analyzes the orders by hour of the day.

- SELECT extract(hour from o.timee) as hours, count(od.order_details_id) as total_orders:
- extract(hour from o.timee) as hours: This extracts the hour of the day from the timee column in the orders table and assigns it to the alias hours.
- count(od.order_details_id) as total_orders: This counts the number of orders for each hour and assigns it to the alias total_orders.
- FROM orders as o: This specifies the orders table as the primary table for the query, assigning it the alias o.
- JOIN order_details as od: This joins the order_details table to the orders table, assigning it the alias od.
- ON od.order_id = o.order_id: This specifies the join condition, which is that the order_id in the order_details table matches the order_id in the orders table.
- GROUP BY extract(hour from o.timee): This groups the results by the hour of the day.
- ORDER BY total_orders DESC: This sorts the results in descending order by the total number of orders for each hour.

```
21
22  --Determine the distribution of orders by hour of the day.
23
24  select extract(hour from o.timee)
25  as hours, count(od.order_details_id) as total_orders
26  from orders as o
27  join order_details as od
28  on od.order_id = o.order_id
29  group by  extract(hour from o.timee)
30  order by total_orders desc;
31
32
```

Data Output | Messages | Notifications

	hours double precision	total_orders bigint
1	12	6543
2	13	6203
3	18	5359
4	17	5143
5	19	4350
6	16	4185
7	14	3521
8	20	3487
9	15	3170
10	11	2672

This query calculates the revenue by month and cumulative total revenue.

- Extracts the year and month from the datee column
- Counts the number of orders for each month
- Calculates the revenue for each month
- Joins the orders, order_details, and pizza tables
- Groups the results by year and month
- Sorts the results by revenue in ascending order
- Calculates the cumulative total revenue

y Query History

```sql
--Group the orders by date and calculate the average number of pizzas ordered per day
--total reveneu per month and the cumulative sum

with x as (select to_char( o.datee, 'yyyy-mm') as year_month ,
count(od.order_details_id) as order_count,
sum(od.quantity*p.price) as rev_by_month
from orders o
join order_details as od
on o.order_id = od.order_id
join pizza as p
on od.pizza_id = p.pizza_id
group by year_month
order by rev_by_month asc)

select x.*,
sum(x.rev_by_month) over (order by year_month range between unbounded preceding and current row)
as cumulative_rev
from x;
```

	year_month text	order_count bigint	rev_by_month numeric	cumulative_rev numeric
1	2015-01	4156	69793.30	69793.30
2	2015-02	3892	65159.60	134952.90
3	2015-03	4186	70397.10	205350.00
4	2015-04	4067	68736.80	274086.80
5	2015-05	4239	71402.75	345489.55
6	2015-06	4025	68230.20	413719.75
7	2015-07	4301	72557.90	486277.65
8	2015-08	4094	68278.25	554555.90

This query provide the avgrage number of order per day

- Extracts the year and month from the datee column
- Counts the number of orders for each month
- Calculates the revenue for each month
- Joins the orders, order_details, and pizza tables
- Groups the results by year and month
- Sorts the results by revenue in ascending order
- Calculates the cumulative total revenue

y Query History

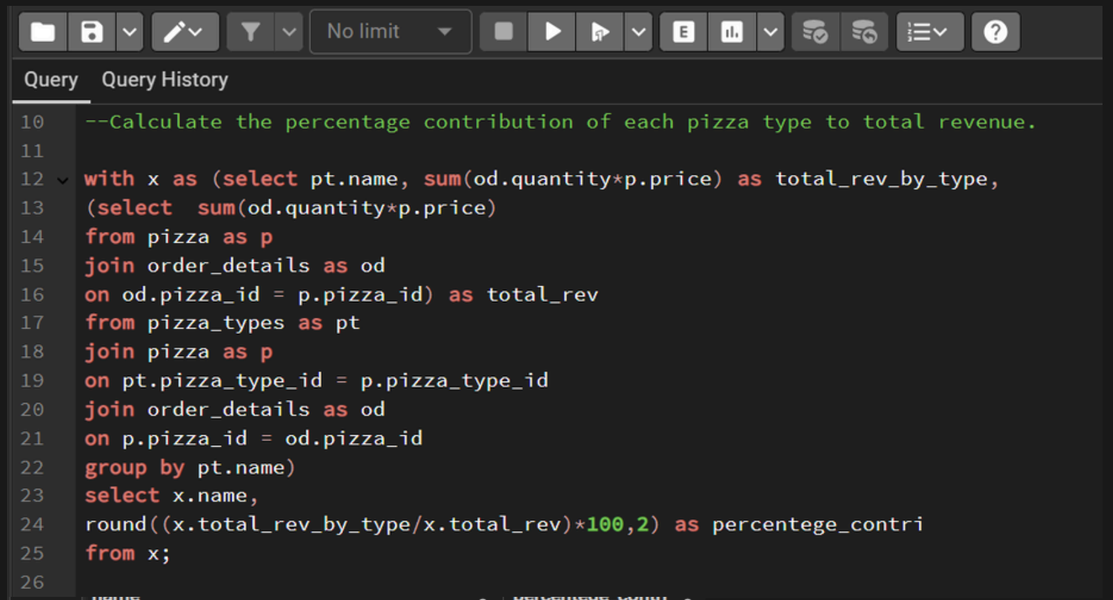
```sql
--Group the orders by date and calculate the average number of pizzas ordered per day.

select o.datee,
round(avg(od.order_details_id),2) as avg_order_dayy
from orders as o
join order_details as od
on o.order_id = od.order_id
group by o.datee
order by o.datee asc;
```

	datee date	avg_order_dayy numeric
1	2015-01-01	81.00
2	2015-01-02	241.50
3	2015-01-03	398.50
4	2015-01-04	528.50
5	2015-01-05	642.00
6	2015-01-06	774.50
7	2015-01-07	913.00
8	2015-01-08	1065.00
9	2015-01-09	1212.00
10	2015-01-10	1346.00

This query calculates the percentage contribution of each pizza type to the total revenue.

- Calculates the total revenue for each pizza type
- Calculates the total revenue across all pizza types
- Joins the pizza_types, pizza, and order_details tables
- Groups the results by pizza type name
- Calculates the percentage contribution of each pizza type to the total revenue



	character varying (200)	numeric
1	The Italian Vegetables Pizza	1.96
2	The Napolitana Pizza	2.95
3	The Thai Chicken Pizza	5.31
4	The Four Cheese Pizza	3.95
5	The Hawaiian Pizza	3.95
6	The Green Garden Pizza	1.71
7	The Barbecue Chicken Pizza	5.23
8	The Southwest Chicken Pizza	4.24
9	The Pepperoni, Mushroom, and Peppers Pizza	2.30
10	The Vegetables + Vegetables Pizza	2.98
11	The Spinach Pesto Pizza	1.91
12	The Italian Supreme Pizza	4.09
13	The Mediterranean Pizza	1.88
14	The Mexicana Pizza	3.27
15	The Spinach and Feta Pizza	2.85
16	The Brie Carre Pizza	1.42
17	The Pepperoni Pizza	3.69
18	The Chicken Pesto Pizza	2.04
19	The Chicken Alfredo Pizza	2.07

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

with q as (with x as (select pt.category, pt.name,
cast(sum(od.quantity*p.price)as integer) as total_rev
from pizza_types as pt
join pizza as p
on pt.pizza_type_id = p.pizza_type_id
join order_details as od
on p.pizza_id = od.pizza_id
group by pt.name,  pt.category
order by   pt.category)

select x.*,
row_number() over (partition by x.category order by total_rev desc) as rn
from x)
select q.category, q.name, q.total_rev
from q
where rn <= 4;
```

	category character varying (200)	name character varying (200)	total_rev integer
1	Chicken	The Thai Chicken Pizza	43434
2	Chicken	The Barbecue Chicken Pizza	42768
3	Chicken	The California Chicken Pizza	41410
4	Chicken	The Southwest Chicken Pizza	34706
5	Classic	The Classic Deluxe Pizza	38181
6	Classic	The Hawaiian Pizza	32273
7	Classic	The Pepperoni Pizza	30162
8	Classic	The Greek Pizza	28454
9	Supreme	The Spicy Italian Pizza	34831
10	Supreme	The Italian Supreme Pizza	33477
11	Supreme	The Sicilian Pizza	30941
12	Supreme	The Pepper Salami Pizza	25529
13	Veggie	The Four Cheese Pizza	32266
14	Veggie	The Mexicana Pizza	26781
15	Veggie	The Five Cheese Pizza	26067
16	Veggie	The Vegetables + Vegetables Pizza	24375

This query calculates the revenue by month and cumulative total revenue.

- Extracts the year and month from the datee column
- Counts the number of orders for each month
- Calculates the revenue for each month
- Joins the orders, order_details, and pizza tables
- Groups the results by year and month
- Sorts the results by revenue in ascending order
- Calculates the cumulative total revenue

REPORT

Total Orders and Revenue: The shop received 48,620 orders, generating a total revenue of ₹817,860.
Most Popular Pizza: The "Thai Chicken Pizza" was the most popular in terms of revenue contribution.
Category-wise Orders: Classic pizzas were the most ordered, followed by Supreme and Veggie.
Order Size Analysis: The most ordered pizza size was Large (L).

Data Analysis

Order Volume and Revenue
Total Orders: 48,620
Total Revenue: ₹817,860

Category-wise Orders

Category	Total Orders
Veggie	11,449
Chicken	10,815
Supreme	11,777
Classic	14,579

Export to Sheets

Top 5 Pizzas by Revenue Contribution

Pizza Name	Revenue Contribution (%)
The Thai Chicken Pizza	5.31
The Barbecue Chicken Pizza	5.23
The Italian Supreme Pizza	4.09
The Four Cheese Pizza	3.95
The Hawaiian Pizza	3.95

Export to Sheets

Total Orders per Month and Revenue with Cumulative Sum

year_month	order_count	rev_by_month	cumulative_rev
2015-01	4156	69793.30	69793.30
2015-01	4301	72557.90	486277.65
2015-02	3892	65159.60	134952.90
2015-03	4186	70397.10	205350.00
2015-04	4067	68736.80	274086.80
2015-05	4239	71402.75	345489.55
2015-06	4025	68230.20	413719.75
2015-08	4094	68278.25	554555.90

Export to Sheets

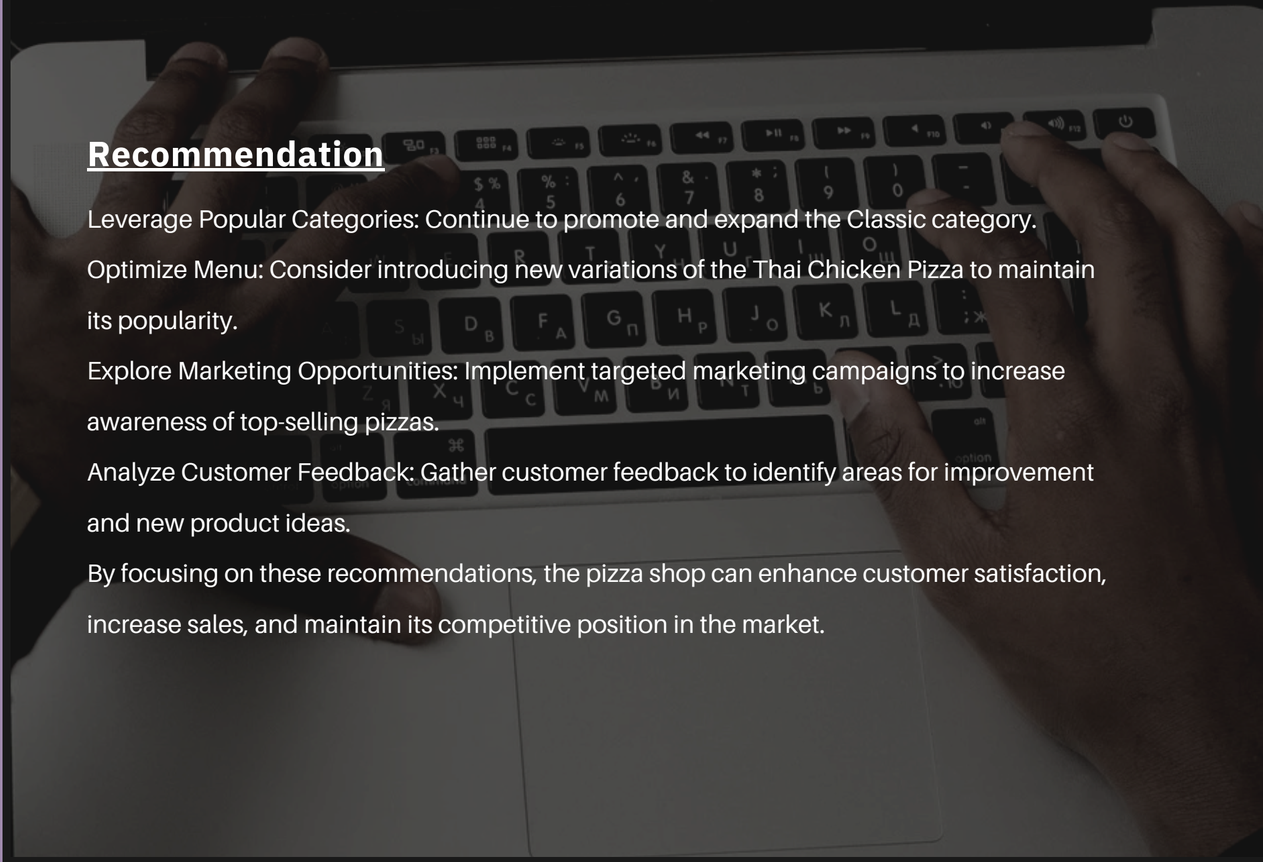
Key insight

Classic Pizzas: The Classic category consistently outperformed others, indicating a strong customer preference.

Thai Chicken Pizza: This pizza's popularity suggests a well-crafted flavor profile that resonates with customers.

Revenue Contribution: The top 5 pizzas account for a significant portion of the total revenue, highlighting the importance of these products.

Order Size: The Large size's popularity suggests a preference for larger portions or sharing.



Recommendation

Leverage Popular Categories: Continue to promote and expand the Classic category.

Optimize Menu: Consider introducing new variations of the Thai Chicken Pizza to maintain its popularity.

Explore Marketing Opportunities: Implement targeted marketing campaigns to increase awareness of top-selling pizzas.

Analyze Customer Feedback: Gather customer feedback to identify areas for improvement and new product ideas.

By focusing on these recommendations, the pizza shop can enhance customer satisfaction, increase sales, and maintain its competitive position in the market.