MySQL Portfolio Project

Introduction

In this project, I have worked on a pizza store database to answer 12 SQL questions ranging from basic to advanced. These queries demonstrate my understanding of SQL concepts like aggregation, joins, subqueries, window functions, and more.

Questions:

1. Basic: Retrieve the Total Number of Orders Placed

Question: Retrieve the total number of orders placed.

SQL Query:

```sql
SELECT
    COUNT(order_id)
FROM
    orders;
```

Explanation: This query counts the total number of order_id entries in the orders table, giving us the total number of orders placed.

Result: (Include a sample result showing the count)

Result Grid | Filter Rows:

COUNT(order_id)
2241

2. Basic: Calculate the Total Revenue Generated from Pizza Sales

Question: Calculate the total revenue generated from pizza sales.

SQL Query:

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

Explanation: This query multiplies the quantity of each pizza by its price, sums up the values, and rounds the total to two decimal places. This gives us the total revenue from pizza sales.

Result: (Include a sample result showing the total revenue)

Result Grid

	total_sales
▶	35880.45

3. Basic: Identify the Highest-Priced Pizza

Question: Identify the highest-priced pizza.

SQL Query:

```sql
SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
WHERE
    price = (SELECT
            MAX(price)
        FROM
            pizzas);
```

Explanation: The query retrieves the name and price of the highest-priced pizza by finding the maximum price in the pizzas table and joining it with the pizza_types table for the name.

Result: (Include a sample result showing the highest-priced pizza)

Result Grid | Filter Row

	name	price
▶	The Greek Pizza	35.95

4. Basic: Identify the Most Common Pizza Size Ordered

Question: Identify the most common pizza size ordered.

SQL Query:

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

Explanation: This query counts the number of orders for each pizza size, then sorts the results in descending order to find the most frequently ordered size.

Result: (Include a sample result showing the most common pizza size)

Result Grid | Filter Rows:

	size	order_count
▶	L	830

5. Basic: List the Top 5 Most Ordered Pizza Types Along with Their Quantities

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

SQL Query:

```sql
SELECT
    pizza_types.name, COUNT(order_details.quantity) AS quantity
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 quantity DESC
LIMIT 5;
```

Explanation: This query counts the quantity of each pizza type ordered, groups them by type, and then orders the results to show the top 5 most ordered types.

Result: (Include a sample result showing the top 5 pizza types)

Result Grid Filter Rows:

name	quantity
The Pepperoni Pizza	124
The Classic Deluxe Pizza	100
The Barbecue Chicken Pizza	100
The Thai Chicken Pizza	99
The Sicilian Pizza	97

6. Intermediate: Find the Total Quantity of Each Pizza Category Ordered

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

SQL Query:

```sql
SELECT
    category, COUNT(quantity)
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 category;
```

Explanation: This query counts the total quantity of pizzas ordered in each category by joining the order_details, pizzas, and pizza_types tables and grouping by the category.

Result: (Include a sample result showing the quantities for each category)

Result Grid | Filter Rows:

category	COUNT(quantity)
Classic	624
Veggie	511
Supreme	540
Chicken	453

7. Intermediate: Determine the Distribution of Orders by Hour of the Day

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

SQL Query:

```sql
SELECT
    HOUR(order_time) AS hours, COUNT(order_id) AS counts
FROM
    orders
GROUP BY hours;
```

Explanation: This query groups orders by the hour of the day they were placed, allowing us to see the distribution of order times.

Result: (Include a sample result showing the distribution by hour)

Result Grid

hours	counts
11	129
12	251
13	245
14	196
15	154
16	199
17	244
18	249
19	208
20	179
21	121
22	65
23	1

8. Intermediate: Find the Category-Wise Distribution of Pizzas

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

SQL Query:

```sql
SELECT
    category, COUNT(name)
FROM
    pizza_types
GROUP BY category;
```

Explanation: This query counts the number of pizzas in each category, grouped by the category column in the pizza_types table.

Result: (Include a sample result showing the category-wise distribution)

Result Grid Filter Rows:

category	COUNT(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

9. Intermediate: Calculate the Average Number of Pizzas Ordered Per Day

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

SQL Query:

```sql
SELECT
    ROUND(AVG(quantity), 2)
FROM
    (SELECT
        order_date, SUM(order_details.quantity) AS quantity
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY order_date) AS order_quantity;
```

Explanation: This query first calculates the total number of pizzas ordered per day and then averages these daily totals to find the average number of pizzas ordered per day.

Result: (Include a sample result showing the average number of pizzas per day)

Result Grid Filter Rows:

ROUND(AVG(quantity), 2)
135.56

10. Advanced: Calculate the Percentage Contribution of Each Pizza Type to Total Revenue

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

SQL Query:

```sql
SELECT
    category,
    ROUND(SUM(order_details.quantity * pizzas.price) / (SELECT
                    ROUND(SUM(order_details.quantity * pizzas.price),
                                2) AS total_sales
                FROM
                    order_details
                        JOIN
                    pizzas ON order_details.pizza_id = pizzas.pizza_id) * 100,
            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 category
ORDER BY revenue DESC
;
```

Explanation: This query calculates the revenue generated by each pizza type, divides it by the total revenue, and then multiplies by 100 to get the percentage contribution of each type.

Result: (Include a sample result showing the percentage contributions)

Result Grid | Filter R

category	revenue
Supreme	26.21
Classic	26.18
Veggie	24.39
Chicken	23.22

11. Advanced: Analyze the Cumulative Revenue Generated Over Time

Question: Analyze the cumulative revenue generated over time.

SQL Query:

```sql
SELECT
    order_date, round(sum(revenue) over(order by order_date),2) as cumulative_revenue
FROM
(SELECT
    orders.order_date, SUM(quantity * price) AS revenue
FROM
    pizzas
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
        JOIN
    orders ON order_details.order_id = orders.order_id
GROUP BY orders.order_date )as sales
ORDER BY cumulative_revenue desc;
```

Explanation: This query calculates the revenue generated on each day and then uses a window function to calculate the cumulative revenue over time.

Result: (Include a sample result showing the cumulative revenue)

Result Grid | Filter Rows:

order_date	cumulative_revenue
2015-01-16	35880.45
2015-01-15	34343.5
2015-01-14	32358.7
2015-01-13	29831.3
2015-01-12	27781.7
2015-01-11	25862.65
2015-01-10	23990.35
2015-01-09	21526.4
2015-01-08	19399.05
2015-01-07	16560.7
2015-01-06	14358.5
2015-01-05	11929.55
2015-01-04	9863.6
2015-01-03	8108.15
2015-01-02	5445.75
2015-01-01	2713.85

12. Advanced: Determine the Top 3 Most Ordered Pizza Types Based on Revenue

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

SQL Query:

```sql
SELECT
    category, round(SUM(quantity * 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 category
ORDER BY revenue DESC
LIMIT 3;
```

Explanation: This query calculates the revenue generated by each pizza type and orders them by revenue, then limits the result to the top 3.

Result: (Include a sample result showing the top 3 pizza types based on revenue)

Result Grid | Filter Row

category	revenue
Supreme	9404.2
Classic	9395.25
Veggie	8750.75

Conclusion

This project provided a deep dive into SQL querying techniques by analyzing data from a pizza store database. The queries demonstrated fundamental SQL operations, as well as more advanced techniques like window functions and subqueries. This exercise has strengthened my understanding of relational databases and SQL.