Pizza Sales Analysis Using MySQL

Author Information

Name: Lavish Kumar
Email: lavishkumar1825@gmail.com
Phone:+91-90012-98501

Project Overview

This project focuses on analyzing **pizza sales data** using MySQL to derive meaningful insights. The dataset consists of **four tables**:

1. **pizzas** – Contains details about different pizzas, including their price and size.
2. **pizza_types** – Provides information about the categories and names of pizzas.
3. **orders** – Stores order-related details such as order ID and timestamp.
4. **order_details** – Contains order-specific information like quantity and pizza type.

By performing **SQL queries**, we answer various business-related questions at three levels—**Basic, Intermediate, and Advanced**—to help understand sales trends, revenue generation, and customer preferences.

SQL Queries and Insights

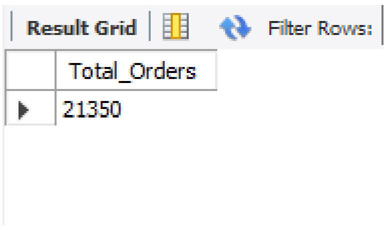
Basic Level

1. **Retrieve the total number of orders placed.**
   - This query calculates the total count of unique order IDs to determine how many orders were placed in the dataset.

   Query:

```
SELECT
    COUNT(order_id) AS Total_Orders
FROM
    orders;
```

Solution:



Result Grid | Filter Rows:

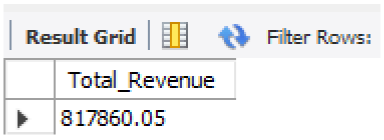
```
  Total_Orders
▶ 21350
```

2. **Calculate the total revenue generated from pizza sales.**
    - By summing up the product of **price and quantity** from order details and pizzas, we compute the total revenue earned.

Query:

```
SELECT
    CAST(SUM(od.quantity * p.price) AS DECIMAL (10 , 2 )) AS Total_Revenue
FROM
    order_details od
        JOIN
    pizzas p ON od.pizza_id = p.pizza_id;
```

Solution:



Result Grid | Filter Rows:

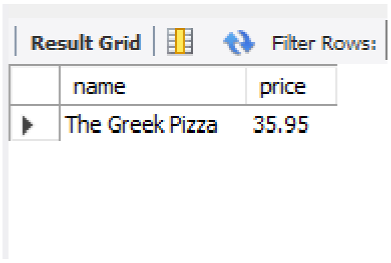
```
  Total_Revenue
▶ 817860.05
```

3. **Identify the highest-priced pizza.**
    - We fetch the pizza with the maximum price from the **pizzas** table and the name of that pizza from pizza_types table by joining them on pizza_type_id.

Query:

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

Solution:



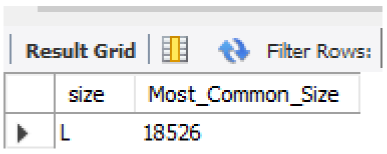
```
name             price
The Greek Pizza  35.95
```

4. **Identify the most common pizza size ordered.**
    - This involves grouping pizzas by size and counting occurrences from order_details to determine which size is the most popular among customers.

Query:

```
SELECT
    p.size, COUNT(od.order_details_id) as Most_Common_Size
FROM
    pizzas p
        JOIN
    order_details od ON p.pizza_id = od.pizza_id
GROUP BY p.size
ORDER BY Most_common_Size DESC
LIMIT 1;
```

Solution:



Result Grid | Filter Rows:

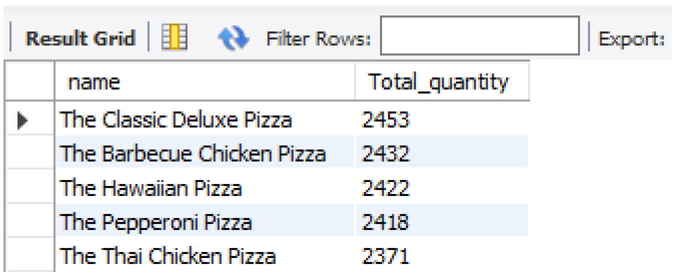
	size	Most_Common_Size
▶	L	18526

5. **List the top 5 most ordered pizza types along with their quantities.**
    - By aggregating the total quantity for each pizza type and sorting them in descending order, we find the top five most popular pizzas.

Query:

```
SELECT
    pt.name, SUM(od.quantity) AS Total_quantity
FROM
    pizzas p
        JOIN
    order_details od ON p.pizza_id = od.pizza_id
        JOIN
    pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.name
ORDER BY Total_quantity DESC
LIMIT 5;
```

Solution:



Result Grid | Filter Rows: | Export:

	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

Intermediate Level

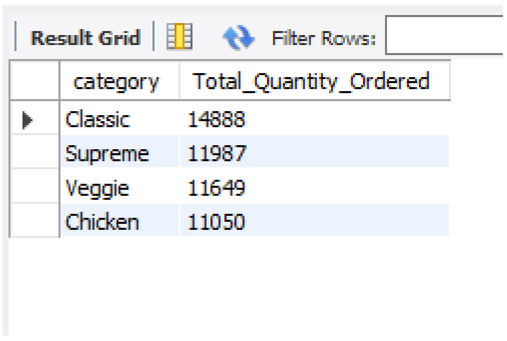
6. **Join the necessary tables to find the total quantity of each pizza category ordered.**

- We combine the **pizza_types, pizzas,** and **order_details** tables to compute the total quantity sold for each category.

Query:

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

Solution:



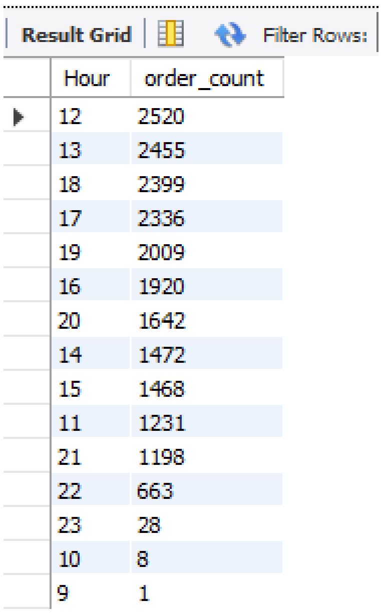
category	Total_Quantity_Ordered
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

7. **Determine the distribution of orders by hour of the day.**
    - Extracting the hour from the order timestamp, we count how many orders were placed in each hour to observe peak order times.

Query:

```
SELECT
    HOUR(order_time) AS Hour, COUNT(order_id) AS order_count
FROM
    orders
GROUP BY hour
ORDER BY order_count DESC;
```

Solution:


Result Grid | Filter Rows:

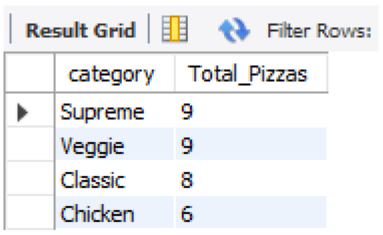
Hour	order_count
12	2520
13	2455
18	2399
17	2336
19	2009
16	1920
20	1642
14	1472
15	1468
11	1231
21	1198
22	663
23	28
10	8
9	1

8. **Join relevant tables to find the category-wise distribution of pizzas.**
    - Using SQL joins, we calculate the total number of pizzas sold per category to analyze category preferences.

Query:

```
SELECT
    category, COUNT(name) AS Total_Pizzas
FROM
    pizza_types
GROUP BY category
ORDER BY Total_Pizzas DESC;
```

Solution:


Result Grid | Filter Rows:

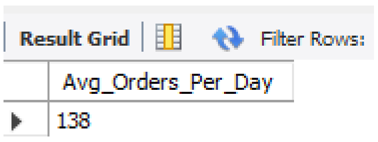
category	Total_Pizzas
Supreme	9
Veggie	9
Classic	8
Chicken	6

9. **Group the orders by date and calculate the average number of pizzas ordered per day.**
   - We group orders by **date** and compute the **average quantity of pizzas sold per day** to analyze daily trends.

   Query:

```
SELECT
    CAST(AVG(Total_Orders) AS DECIMAL (10.2)) AS Avg_Orders_Per_Day
FROM
    (SELECT
        CAST(SUM(od.quantity) AS DECIMAL (10 , 2 )) AS Total_Orders
    FROM
        orders o
    JOIN order_details od ON o.order_id = od.order_id
    GROUP BY o.order_date
    ORDER BY Total_orders DESC) AS order_quantity;
```

   Solution:


Result Grid | Filter Rows:

Avg_Orders_Per_Day
138

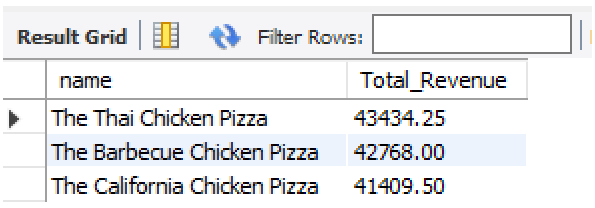
10. **Determine the top 3 most ordered pizza types based on revenue.**

- By calculating revenue per pizza type and sorting in descending order, we identify the top three revenue-generating pizzas.

Query:

```
SELECT
    pt.name,
    CAST(SUM(od.quantity * p.price) AS DECIMAL (10 , 2 )) AS Total_Revenue
FROM
    pizza_types pt
        JOIN
    pizzas p ON pt.pizza_type_id = p.pizza_type_id
        JOIN
    order_details od ON od.pizza_id = p.pizza_id
GROUP BY pt.name
ORDER BY Total_Revenue DESC
LIMIT 3;
```

Solution:


Result Grid | Filter Rows:

name	Total_Revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768.00
The California Chicken Pizza	41409.50

Advanced Level

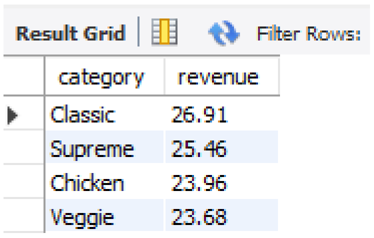
11. **Calculate the percentage contribution of each pizza type to total revenue.**

- We compute the revenue share of each pizza type as a percentage of the total revenue to determine its impact.

Query:

```
SELECT
    pt.category,
    ROUND(CAST(SUM(od.quantity * p.price) AS DECIMAL (10 , 2 )) / (SELECT
                    SUM(od.quantity * p.price)
                FROM
                    pizza_types pt
                        JOIN
                    pizzas p ON pt.pizza_type_id = p.pizza_type_id
                        JOIN
                    order_details od ON p.pizza_id = od.pizza_id) * 100,
            2) AS revenue
FROM
    pizza_types pt
        JOIN
    pizzas p ON pt.pizza_type_id = p.pizza_type_id
        JOIN
    order_details od ON p.pizza_id = od.pizza_id
GROUP BY pt.category
ORDER BY revenue DESC;
```

Solution:


Result Grid | Filter Rows:

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

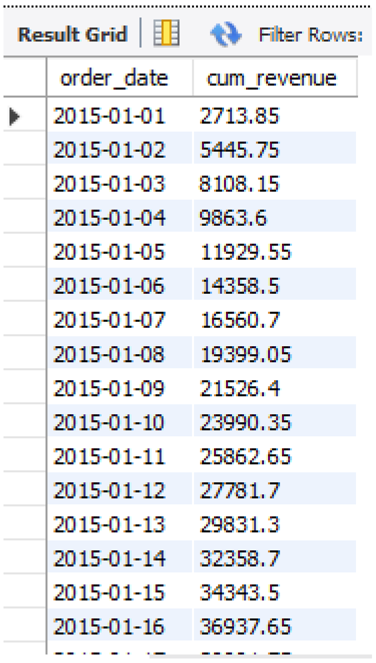
12. **Analyze the cumulative revenue generated over time.**

- Using SQL **window functions**, we calculate the running total of revenue to track sales growth.

Query:

```
select order_date, round(sum(revenue) over (order by order_date),2) as cum_revenue
from
(select o.order_date, sum(od.quantity*p.price) as revenue
from order_details od join pizzas p
on od.pizza_id = p.pizza_id
join orders o
on o.order_id = od.order_id
group by o.order_date) as sales;
```

Solution:


Result Grid | Filter Rows:

order_date	cum_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

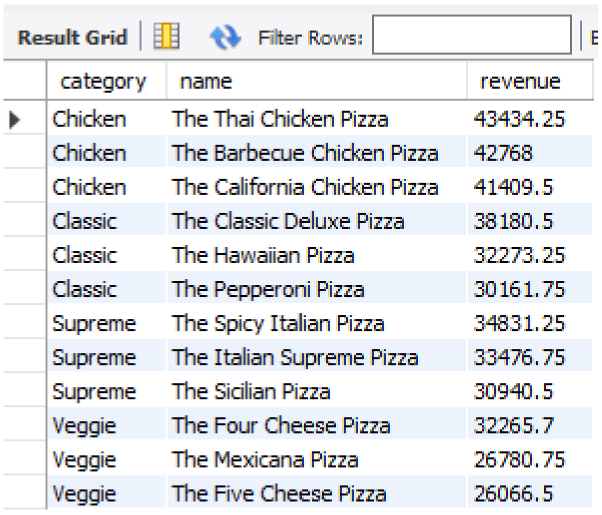
13. **Determine the top 3 most ordered pizza types based on revenue for each pizza category.**

- We rank pizza types within each category based on revenue to see which pizzas perform best in different categories.

Query:

```
select category, name, revenue from
(select category, name, revenue,
rank() over(partition by category order by revenue desc) as top from
(select pt.name, pt.category, round(sum(od.quantity*p.price),2) as revenue
from order_details od join pizzas p
on od.pizza_id = p.pizza_id
join pizza_types pt
on pt.pizza_type_id = p.pizza_type_id
group by pt.category, pt.name) as sales) as rank_find
where top<=3;
```

Solution:



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
Supreme	The Spicy Italian Pizza	34831.25
Supreme	The Italian Supreme Pizza	33476.75
Supreme	The Sicilian Pizza	30940.5
Veggie	The Four Cheese Pizza	32265.7
Veggie	The Mexicana Pizza	26780.75
Veggie	The Five Cheese Pizza	26066.5

Conclusion

This project provides valuable insights into **sales performance, customer preferences, and revenue distribution** for a pizza business. By leveraging SQL queries, we extracted meaningful patterns that can help in decision-making, marketing strategies, and inventory management.
