Uncovering Sales Insights with the Power of SQL!

PIZZA SALES REPORT - SQL PROJECT

By Amber Khare

PROJECT OVERVIEW

This project analyzes pizza sales data using MySQL to uncover key business insights.
By writing sql queries, I explored order trends, revenue patterns, and customer preferences across different pizza types and sizes.

PROBLEMS

Basic:

- Retrieve the total number of orders placed.
- Calculate the total revenue generated from pizza sales.
- Identify the highest-priced pizza.
- Identify the most common pizza size ordered.

Intermediate:

- List the top 5 most ordered pizza types along with their quantities.
- find the total quantity of each pizza ordered.
- Determine the distribution of orders by hour of the day.
- find category wise distribution of pizzas.

Advanced:

- Determine the top 3 most ordered pizza types based on revenue.
- Analyze the cumulative revenue generated over time.
- Determine the top 3 most ordered pizza types based on revenue for each pizza category.
- Group the orders by date and calculate the average number of pizzas ordered per day.

BASIC

1. **Retrieve the total number of orders placed.**

```
-- 1. Total No. Of Orders Placed.
select count(order_id) as Total_Orders
 from orders;
```

Result Grid

	Total_Orders
▶	21350

2. **Calculate the total revenue generated from pizza sales.**

```
-- 2. Total Revenue generated from Pizza Sales.

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

Result Grid

	Total_Revenue
▶	817860.05

BASIC

3.Identify the highest-priced pizza.

```sql
-- 3. Higest Priced Pizza

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

Result Grid | Filter Rows:

name	Higest_Prized_Pizza
The Greek Pizza	35.95

4.Identify the most common pizza size ordered.

```sql
-- 4. Most common Pizza size Ordered
SELECT
    pizzas.size,
    COUNT(order_details.order_details_id) AS Most_Common_size
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY Most_Common_size DESC;
```

Result Grid | Filter Rows

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

INTERMEDIATE

1. **List the top 5 most ordered pizza types along with their quantities.**

```sql
-- 5. Top 5 most ordered pizza types with quantities
SELECT
    pizza_types.name,
    COUNT(order_details.order_details_id) AS Total_Orders,
    SUM(order_details.quantity) AS Quantity
FROM
    order_details
        JOIN
    pizzas ON order_details.pizza_id = pizzas.pizza_id
        JOIN
    pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY pizza_types.name
ORDER BY Total_Orders DESC
LIMIT 5;
```

	name	Total_Orders	Quantity
▶	The Classic Deluxe Pizza	2416	2453
	The Barbecue Chicken Pizza	2372	2432
	The Hawaiian Pizza	2370	2422
	The Pepperoni Pizza	2369	2418
	The Thai Chicken Pizza	2315	2371

INTERMEDIATE

2.**Find the total quantity of each pizza ordered by category.**

```sql
-- 6. Join necessary tables to find total quantity of each pizza category
SELECT
    pizza_types.category,
    SUM(order_details.quantity) AS Total_quantity
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 Total_quantity DESC;
```

category	Total_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

INTERMEDIATE

3.**Determine the distribution of orders by hour of the day.**

Hours	Order_count
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
10	8
9	1

```sql
-- 7. Determine distribution of orders by hour of the day

SELECT
    HOUR(order_time) AS Hours, COUNT(order_id) AS Order_count
FROM
    orders
GROUP BY HOUR(orders.order_time) ;
```

INTERMEDIATE

4.**Find category wise distribution of pizzas.**

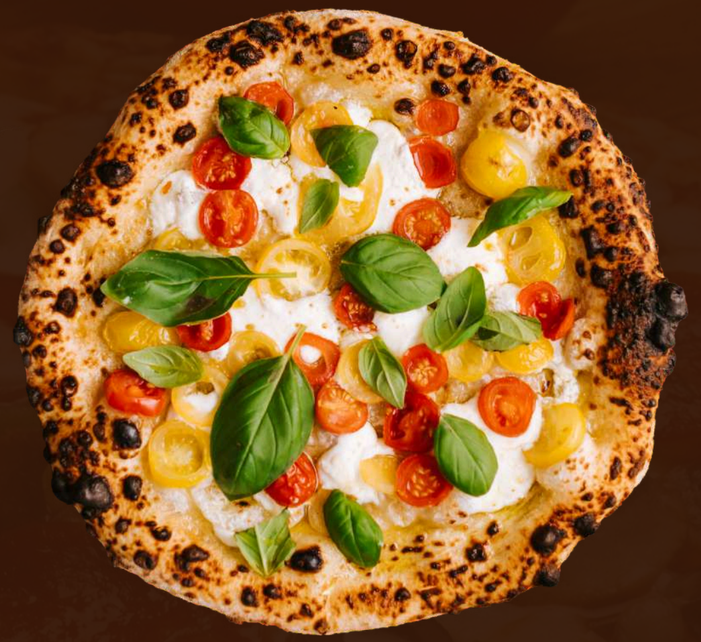
```
-- 8. Category wise distribution of pizzas

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

	category	pizza_types
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

ADVANCED

1.Determine the top 3 most ordered pizza types based on Revenue.

```sql
-- 9. determine Top 3 most ordered pizza types based on revenue of each pizza category
select category , Revenue , name,
rank() over(partition by category order by Revenue desc) as ranking
from
(select pizza_types.category , pizza_types.name,
sum(order_details.quantity * pizzas.price) as Revenue
from order_details
join pizzas
on order_details.pizza_id = pizzas.pizza_id
join pizza_types
on pizza_types.pizza_type_id = pizzas.pizza_type_id
group by  pizza_types.category , pizza_types.name) as sales
limit 3 ;
```

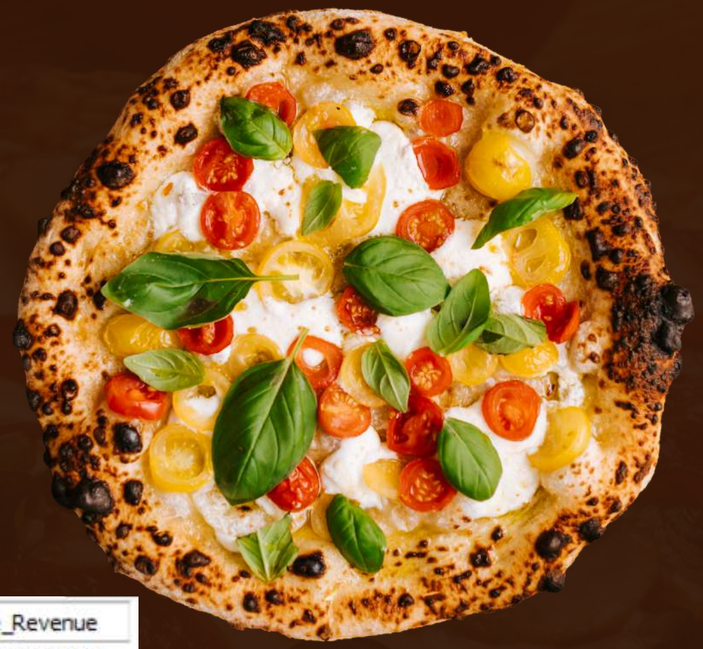


	category	Revenue	name	ranking
▶	Chicken	43434.25	The Thai Chicken Pizza	1
	Chicken	42768	The Barbecue Chicken Pizza	2
	Chicken	41409.5	The California Chicken Pizza	3

ADVANCED

2.Analyze the cumulative revenue generated over time.

```sql
-- 10. Analyze Cumlative revenue generated over time

select order_date , sum(Revenue)
over(order by order_date) 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;
```

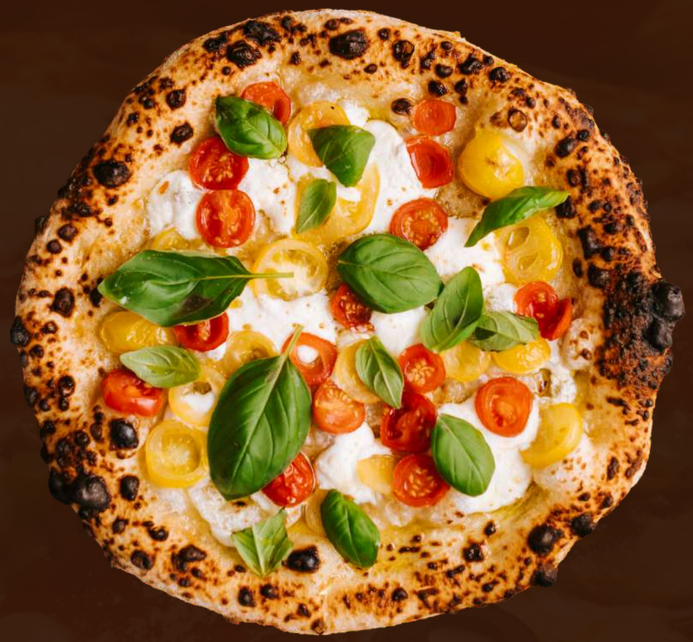


order_date	cumulative_Revenue
2015-01-01	2713.8500000000004
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55
2015-01-06	14358.5
2015-01-07	16560.7
2015-01-08	19399.05
2015-01-09	21526.4
2015-01-10	23990.350000000002
2015-01-11	25862.65
2015-01-12	27781.7

ADVANCED

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

```sql
-- 11. Top 3 most ordered pizza types based on revenue

SELECT
    pizza_types.name AS Pizza_name,
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS Revenue
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id
        JOIN
    pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY pizza_types.name
ORDER BY Revenue DESC
LIMIT 3;
```

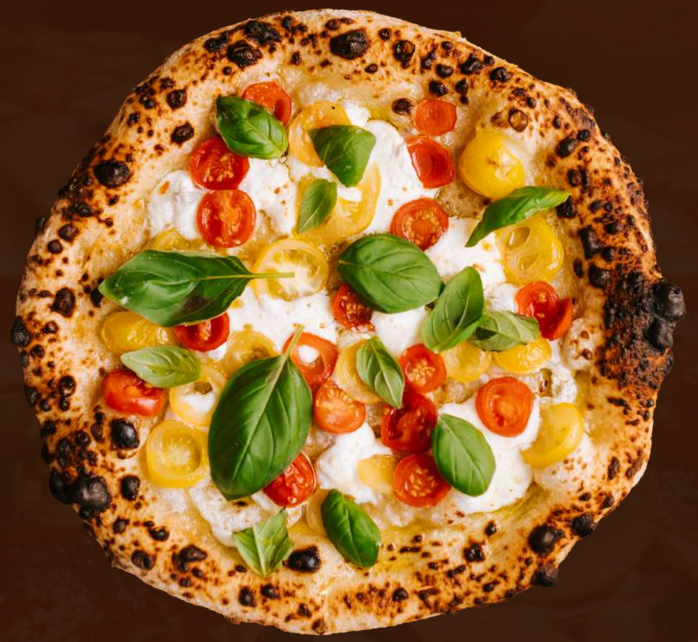


	Pizza_name	Revenue
▶	The Thai Chicken Pizza	43434.25
	The Barbecue Chicken Pizza	42768
	The California Chicken Pizza	41409.5

ADVANCED

4.**Group the orders by date and calculate the average number of pizzas ordered per day.**

```sql
-- 12. group the orders by date and calculate average number of pizzas ordered per day
SELECT
    ROUND(AVG(Total_Quantity), 0) AS Average_pizzas_ordered_perday
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 Order_Quantity;
```



	Average_pizzas_ordered_perday
▶	138

CONCLUSION & KEY TAKEWAYS

- SQL is a powerful tool for deriving business insights from raw data
- Sales trends, revenue analysis, and customer preferences were uncovered
- Used MySQL to handle real-world data with JOINs, GROUP BY, and aggregations
- This project improved my skills in data analysis and query optimization
- The insights can help businesses make data-driven decisions

https://www.linkedin.com/in/amber-khare2002

THANK YOU

Amberkhare961@gmail.com

https://github.com/Amber8680