PIZZA SALES ANALYSIS USING SQL

BY : RAHUL

OBJECTIVE:

The goal of this project is to analyze pizza sales data to better understand customer preferences, how revenue is generated, and ordering habits. By looking at various metrics, I aim to identify trends and opportunities to improve the pizza sales process. This analysis will help me enhance my understanding of customer behavior, streamline operations, and ultimately increase both profitability and customer satisfaction.

DATABASE SCHEMA

pizzas

- pizza_id
- pizza_type_id
- price
- size

orders

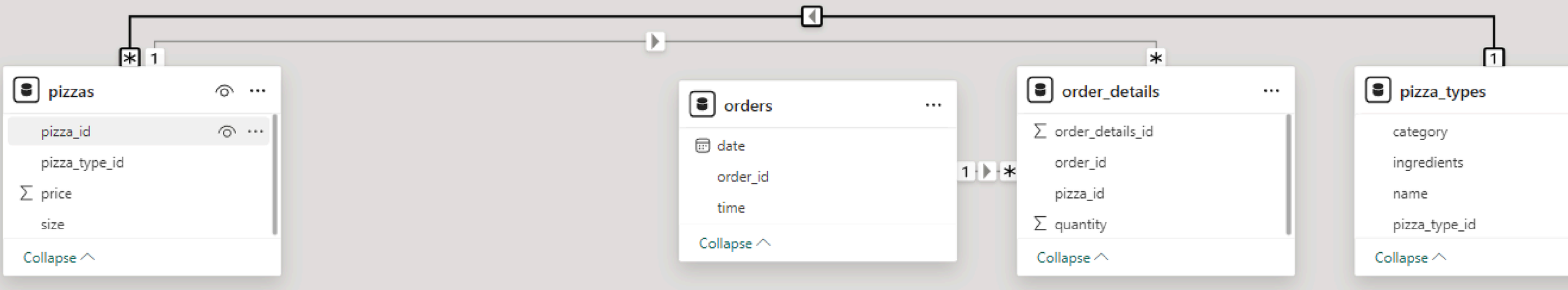
- date
- order_id
- time

order_details

- order_details_id
- order_id
- pizza_id
- quantity

pizza_types

- category
- ingredients
- name
- pizza_type_id



RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

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

Result Grid

total_orders
21350

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

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

Result Grid

total_revenue
817860.05

Identify the highest-priced pizza.

```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(pizzas.price)
        FROM
            pizzas);

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

Result Grid | Filter

name	price
The Greek Pizza	35.95

Identify the most common pizza size ordered.

```sql
SELECT
    pizzas.size, count(order_details.quantity) AS total_orders
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY total_orders DESC
Limit 1;
```

Result Grid

size	total_orders
L	18526

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

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

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

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

```sql
SELECT
    pizza_types.category,
    SUM(order_details.quantity) AS categ_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.category
ORDER BY categ_quantity DESC;
```

Result Grid | Filter Rov

category	categ_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

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

Result Grid

hour	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

JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

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

Result Grid | Filter Ro

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

GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

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

Result Grid

avg_orders_per_day
138

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

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

Result Grid | Filter Rows:

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

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

```sql
SELECT
    pizza_types.category,
    ROUND(SUM(order_details.quantity * pizzas.price) / (SELECT
                    ROUND(SUM((pizzas.price * order_details.quantity)),
                                2) AS total_revenue
                FROM
                    pizzas
                        JOIN
                    order_details ON pizzas.pizza_id = order_details.pizza_id) * 100,
            2) AS reveneue_contribution
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizza_types.category;
```

Result Grid | Filter Rows:

category	reveneue_contribution
Classic	26.91
Veggie	23.68
Supreme	25.46
Chicken	23.96

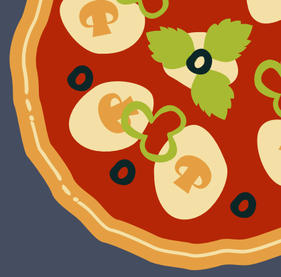
ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```sql
SELECT
    order_date,
    sum(revenue) OVER(ORDER BY order_date) AS cumulative_revenue
FROM
    (SELECT orders.order_date,
    sum(pizzas.price*order_details.quantity) AS revenue
    FROM
        pizzas
            JOIN
        order_details
        ON pizzas.pizza_id=order_details.pizza_id
            JOIN
        orders
        ON order_details.order_id=orders.order_id
        GROUP BY orders.order_date) AS sales;
```

Result Grid | Filter Rows:

order_date	cumulative_revenue
2015-01-01	2713.850000000004
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
2015-01-15	34343.50000000001
2015-01-16	36937.65000000001
2015-01-17	39001.75000000001
2015-01-18	40978.600000000006
2015-01-19	43365.75000000001
2015-01-20	45763.65000000001
2015-01-21	47804.20000000001

DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.



```sql
select category, name, revenue from
(select category, name, revenue, rank() over(partition by category order by revenue desc) as rn
from
(SELECT
    pizza_types.category, pizza_types.name,
    ROUND(SUM(pizzas.price * order_details.quantity),
            2) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizza_types.category, pizza_types.name) as pizza_categ_revenue) as pizza_rank
where rn<=3;
```

Result Grid | Filter Rows:

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

Understanding customer prefrences and behaviour is essential for any business seeking to thrive in a competitive markrt.By analyzing pizza sales data, we can tailor our offerings to meet customer demands more effectively, optimizing pricing strategies, and improve operational effeciency.