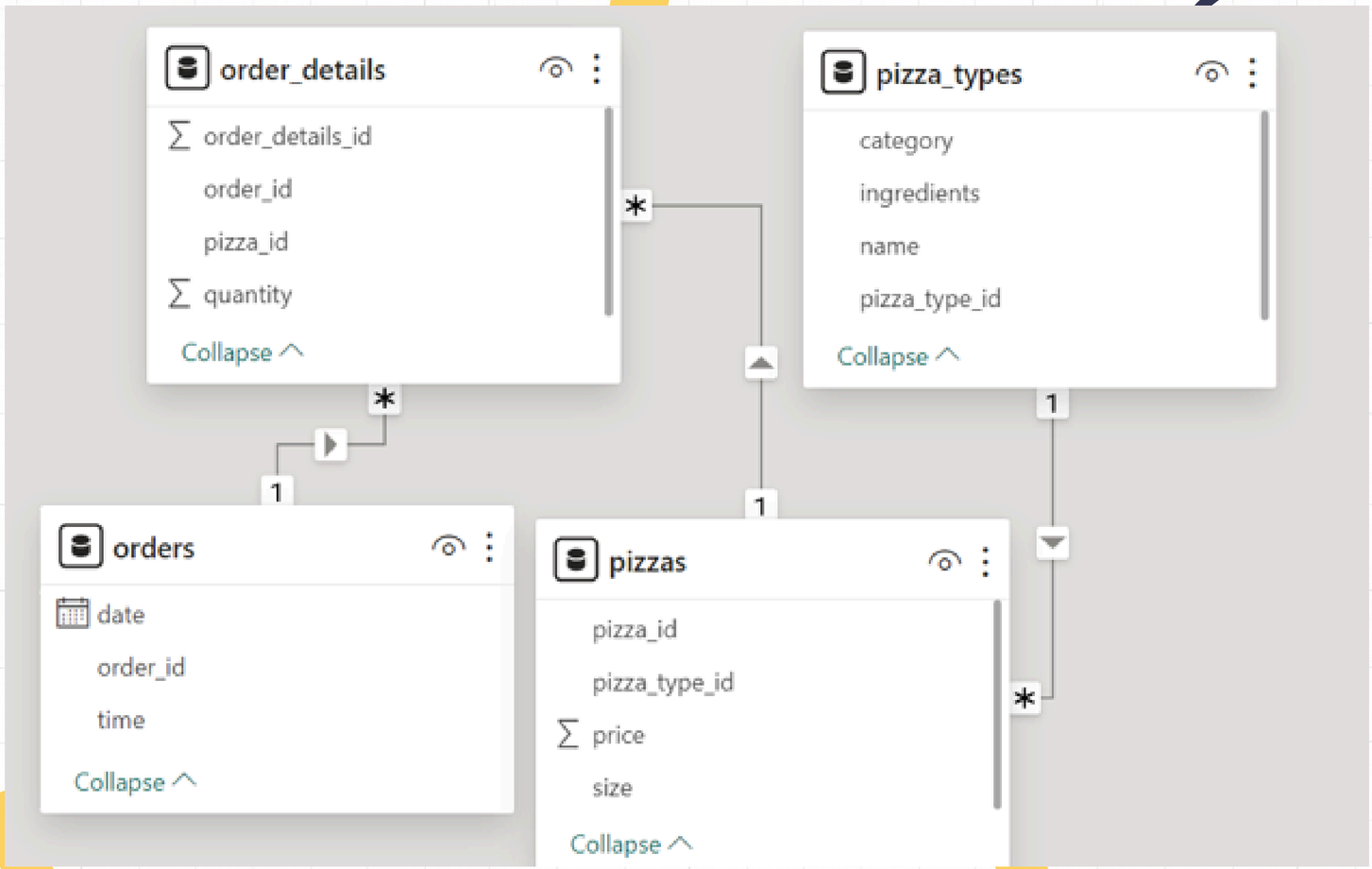
PIZZA SALES

CREATIVE PROJECT

BY - HARSHIT CHAUBEY

HELLO !



I am Harshit Chaubey.
In this project i have utilised MySQL queries to solve questions that are related to pizza sales.

THE PROJECT

Introduction: Our project focuses on using MySQL to analyze pizza sales data across four key tables: pizza, pizza_types, orders, and order_details.

Objective: We aim to extract insights on customer preferences and sales trends to inform strategic decision-making in the culinary industry.

Conclusion: Through MySQL querying, our project seeks to provide actionable insights for optimizing business strategies and driving growth in the pizza market.

QUERIES

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

INTERMEDIATE

- Join the necessary tables to find the total quantity of each pizza category ordered.
- Determine the distribution of orders by hour of the day.
- Join relevant tables to find the category-wise distribution of pizzas.
- Group the orders by date and calculate the average number of pizzas ordered per day.
- Determine the top 3 most ordered pizza types based on revenue.

ADVANCED

- Calculate the percentage contribution of each pizza type to total revenue.
- Analyze the cumulative revenue generated over time.
- Determine the top 3 most ordered pizza types based on revenue for each pizza category.

1. RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

```
SELECT
    COUNT(order_id) AS total_orders_placed
FROM
    pizzahut.orders;
```

Result Grid | Filt

	total_orders_placed
▶	21350

2. CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

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

Result Grid

	total_sales
▶	817860.05

3. IDENTIFY THE HIGHEST-PRICED PIZZA.

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

Result Grid | Filter Row

	name	price
▶	The Greek Pizza	35.95

4. IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

```
SELECT
    pizzas.size, COUNT(order_details.order_details_id)
FROM
    order_details
        JOIN
    pizzas ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizzas.size
ORDER BY size;
```

Result Grid | Filter Rows:

size	count(order_details.order_details_id)
L	18526
M	15385
S	14137
XL	544
XXL	28

5. LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.

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

Result Grid | Filter Rows:

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

6. JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

```
SELECT
    pizza_types.category,
    SUM(order_details.quantity) AS 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 quantity DESC;
```

Result Grid | Filter

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

7. DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

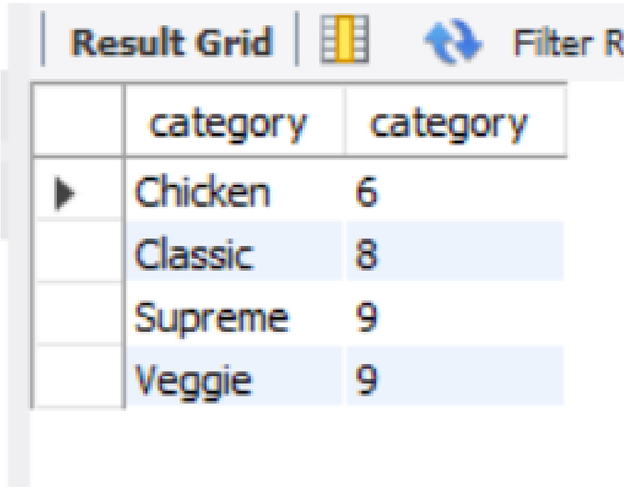
```
SELECT
    HOUR((orders.order_time)), COUNT(orders.order_id)
FROM
    orders
GROUP BY HOUR(orders.order_time);
```

Result Grid | Filter Rows: | Expor

HOUR((orders.order_time))	COUNT(orders.order_id)
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

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

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



	category	category
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

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

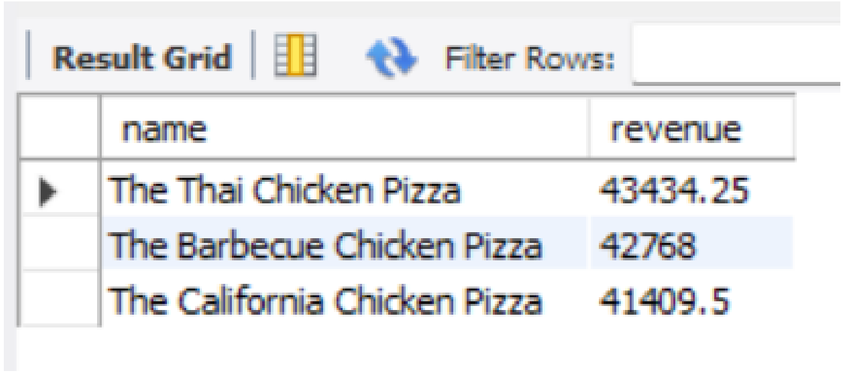
```
SELECT
    AVG(quantity)
FROM
    (SELECT
        orders.order_date, SUM(order_details.quantity) AS quantity
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY orders.order_date) AS order_quantity;
```

Result Grid

	avg(quantity)
▶	138.4749

10. DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```
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 order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```



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

11. CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

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

Result Grid | Filter

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

12. ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```
SELECT order_date,
SUM(revenue) OVER (ORDER BY order_date) AS cum_revenue
FROM
(SELECT orders.order_date,ROUND(SUM(order_details.quantity * pizzas.price),2) AS revenue
FROM order_details JOIN orders
ON order_details.order_id = orders.order_id
JOIN pizzas
ON order_details.pizza_id = pizzas.pizza_id
GROUP BY orders.order_date
ORDER BY orders.order_date) AS sales;
```

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	21526.399999999998
2015-01-10	23990.35
2015-01-11	25862.649999999998
2015-01-12	27781.699999999997
2015-01-13	29831.299999999996
2015-01-14	32358.699999999997

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

```
SELECT name,revenue
FROM
(SELECT category,name,revenue,
rank() OVER (PARTITION BY category ORDER BY revenue DESC) AS rk
FROM
(SELECT pizza_types.category,pizza_types.name, SUM(order_details.quantity * pizzas.price) as revenue
FROM pizzas JOIN order_details
ON pizzas.pizza_id = order_details.pizza_id
JOIN pizza_types
ON pizzas.pizza_type_id = pizza_types.pizza_type_id
GROUP BY pizza_types.category,pizza_types.name) AS a) AS b
WHERE rk <= 3;
```

Result Grid | Filter Rows:

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
The Classic Deluxe Pizza	38180.5
The Hawaiian Pizza	32273.25
The Pepperoni Pizza	30161.75
The Spicy Italian Pizza	34831.25
The Italian Supreme Pizza	33476.75
The Sicilian Pizza	30940.5
The Four Cheese Pizza	32265.70000000065
The Mexicana Pizza	26780.75
The Five Cheese Pizza	26066.5

CONCLUSION

In conclusion, our project has demonstrated the power of MySQL in extracting actionable insights from pizza sales data. From uncovering customer preferences to identifying sales trends, we've gained valuable knowledge that can inform strategic decisions in the culinary industry

THANKYOU !