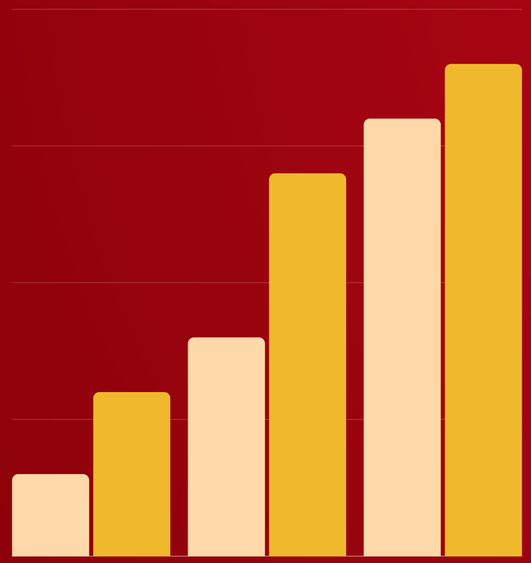
PIZZA SALES ANALYTICS USING SQL

LEVERAGING SQL TO DERIVE KEY INSIGHTS FROM SALES DATA

IMAAN ZAHRA



MySQL®

ABOUT ME

I am a Python and SQL developer with a passion for data science and data analysis. With a solid foundation in SQL, I have worked on projects like pizza sales analytics to extract actionable business insights.

KEY SKILLS

- SQL (MySQL/SQLite)
- Data Analysis
- Python
- Power BI

PROJECT OVERVIEW

QUESTIONS COVERED FROM BASIC TO ADVANCE LEVEL

BASIC

1. Retrieve the total number of orders placed.
2. Calculate the total revenue generated from pizza sales.
3. Identify the highest-priced pizza.
4. Identify the most common pizza size ordered.
5. List the top 5 most ordered pizza types along with their quantities.

INTERMEDIATE

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

ADVANCE

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

DATA OVERVIEW

TABLES USED

- Orders
- Pizza Types
- Order Details
- Pizzas

KEY COLUMNS

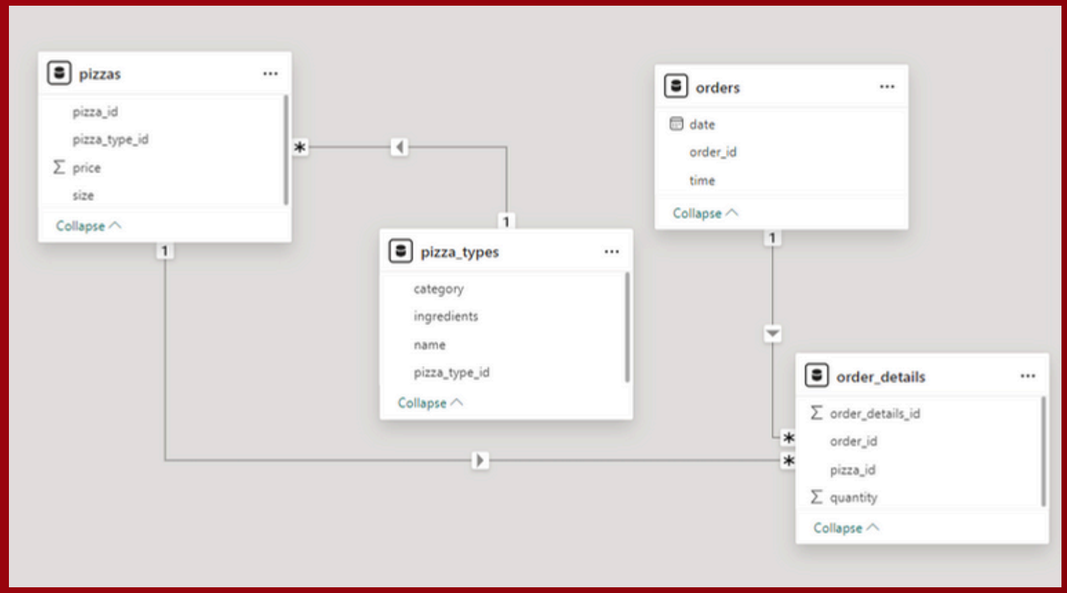
order_id, pizza_type_id, pizza_id, category, name, price, quantity

DATASET

This dataset provided all the necessary information to analyze pizza sales and extract insights using SQL queries.

LINK TO DATASET: https://shorturl.at/emPpM

DATA MODELING USING POWER BI



SQL QUERIES - BASIC

Retrieve the total number of orders placed.

```
1      -- total number of orders placed
2 •    USE pizzastore;
3 •    SELECT count(order_id) FROM orders;
```

OUTPUT

	count(order_id)
▶	21350

SQL QUERIES - BASIC

Calculate the total revenue generated from pizza sales.

```sql
-- total revenue generated from sales
SELECT
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS total_sales
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id;
```

OUTPUT

	total_sales
▶	817860.05

SQL QUERIES - BASIC

Identify the highest-priced pizza.

```sql
-- highest price pizza
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;
```

OUTPUT

	name	price
▶	The Greek Pizza	35.95

SQL QUERIES - BASIC

Identify the most common pizza size ordered.

```sql
-- most common pizza ordered

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

OUTPUT

	size	order_count
▶	L	18526

SQL QUERIES - BASIC

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

```sql
-- top 5 most ordered pizzas with their quantities
SELECT
    pizza_types.name,
    SUM(order_details.quantity) AS quantity_ordered
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_ordered DESC
LIMIT 5;
```

OUTPUT

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

SQL QUERIES - INTERMEDIATE

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

```sql
-- total quantity of each 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;
```

OUTPUT

	category	quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

SQL QUERIES - INTERMEDIATE

Determine the distribution of orders by hour of the day.

```sql
-- Distribution of orders by hour

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

OUTPUT

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

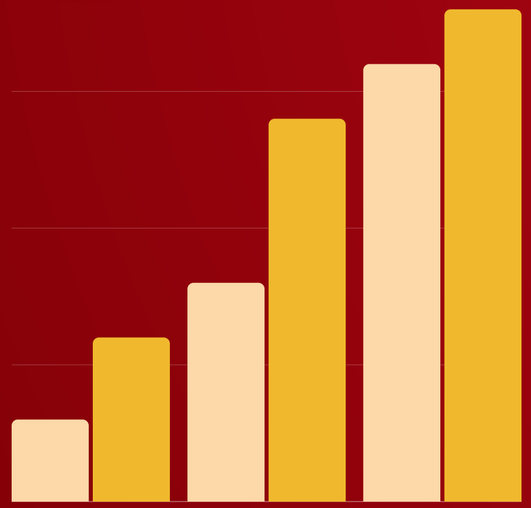
SQL QUERIES - INTERMEDIATE

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

```sql
-- Find the category-wise distribution of pizzas.
SELECT
    category, COUNT(name)
FROM
    pizza_types
GROUP BY category;
```

OUTPUT

	category	count(name)
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9



SQL QUERIES - INTERMEDIATE

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

```sql
-- orders by date and average, no of pizzas ordered per day

SELECT
    ROUND(AVG(quantity), 0) as avg_pizza_ordered_per_day
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;

```

OUTPUT

	avg_pizza_ordered_per_day
▶	138

SQL QUERIES - INTERMEDIATE

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

```sql
-- top 3 ordered pizza 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;
```

OUTPUT

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

SQL QUERIES - ADVANCE

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

```sql
-- pizza type contribution in revenue on category by percentage

SELECT
    pizza_types.category,
    ROUND((SUM(order_details.quantity * pizzas.price) / (SELECT
                    ROUND(SUM(order_details.quantity * pizzas.price),
                                2)
                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 pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY pizza_types.category
ORDER BY revenue DESC;
```

OUTPUT

	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

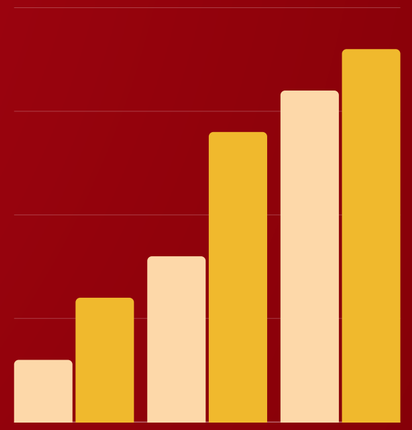
SQL QUERIES - ADVANCE

Analyze the cumulative revenue generated over time.

```sql
-- commultative revenue generated over time
SELECT order_date,
sum(revenue) over(order by order_date) as cum_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;
```

OUTPUT

order_date	cum_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
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



SQL QUERIES - ADVANCE

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

```sql
-- Top 3 of each category based on revenue

SELECT 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,
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.category, pizza_types.name) AS a) AS b
WHERE rn<=3;
```

OUTPUT

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

- This SQL analysis of pizza sales provided actionable insights for optimizing sales and product placement. The business could focus on promoting high-revenue pizza categories, adjust staffing during peak hours, and capitalize on high-demand days like weekends.
- Through this project, I enhanced my SQL skills, particularly in query optimization, data aggregation, and deriving business insights.

MySQL®

LET'S CONNECT

Name: Imaan Zahra
Email: imaanzahra529@gmail.com
LinkedIn: www.linkedin.com/in/imaan-zahra-/
GitHub: https://github.com/ImaanZahra/