PIZZA SALES ANALYSIS

SQL

ABOUT ME

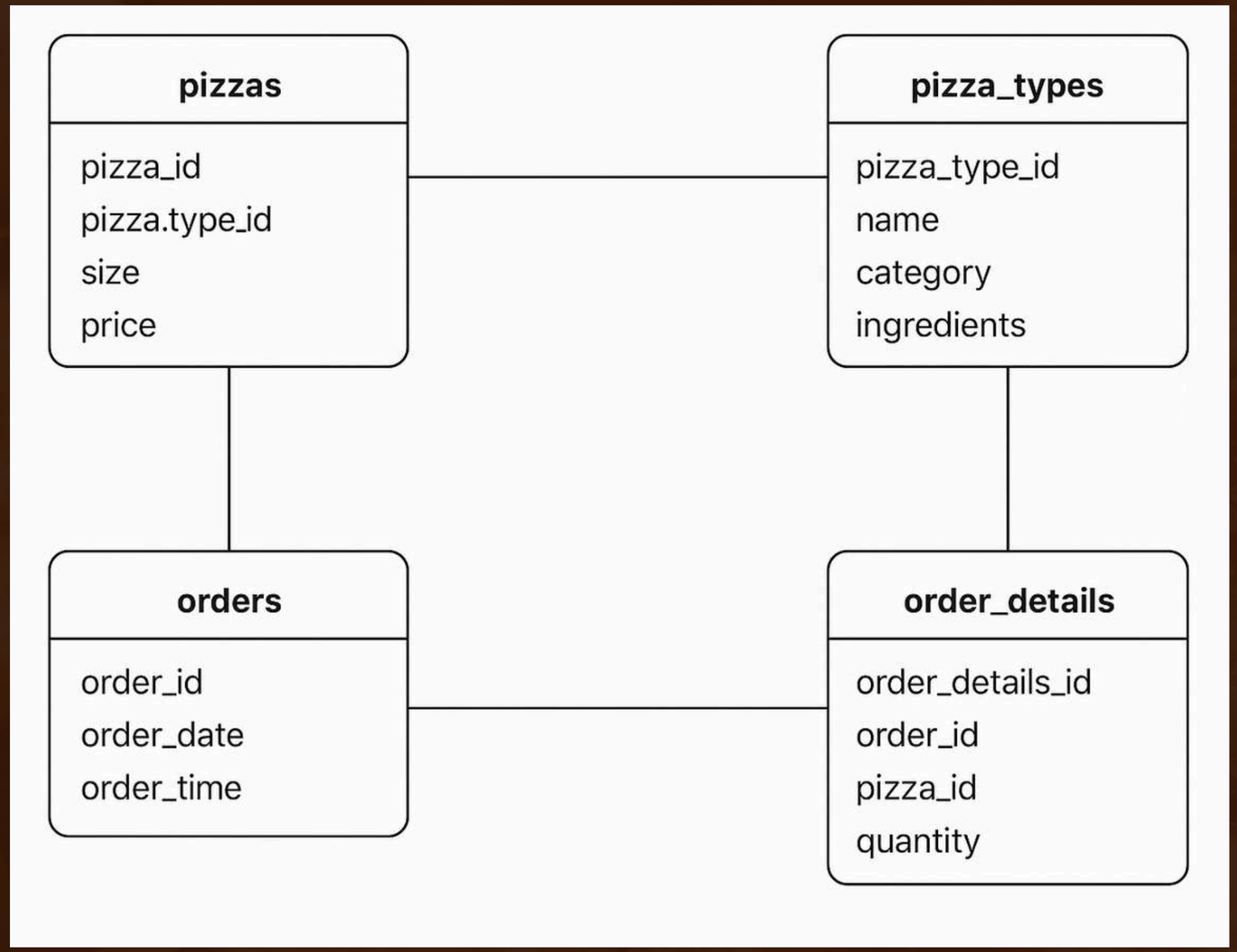
I am an aspiring data analyst with a strong interest in SQL, data storytelling, and problem-solving. This is my first project in SQL, created to build a solid foundation in writing queries, understanding database structure, and generating insights from structured data. I followed a guided tutorial to learn practical skills and understand how business-related questions can be answered through data.

ABOUT PROJECT

This project is a sales analysis based on a pizza store dataset. Using SQL, I explored the dataset to answer real-world business questions such as identifying top-selling items, analyzing order trends, and evaluating revenue patterns.

The aim was to practice SQL concepts, work with relational data, and extract meaningful insights that could support business decisions.

ABOUT DATABASE SCHEMA



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

Result Grid

total_revenue
817860.05

IDENTIFY THE HIGHEST-PRICED 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;
```

Result Grid | Filter Rows:

name	price
The Greek Pizza	35.95

IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

```sql
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;
```

Result Grid | Filter

size	order_count
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_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 total_quantity DESC LIMIT 5;
```

Result Grid | Filter Rows:

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

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

```sql
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 pizzas.pizza_id = order_details.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity DESC;
```

category	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 time, COUNT(order_id) AS order_count
FROM
    orders
GROUP BY HOUR(order_time)
ORDER BY HOUR(order_time);
```

Result Grid

time	order_count
9	1
10	8
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336

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 Rows:

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), 0) AS average_pizza_perday
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 | Filte

	average_pizza_perday
▶	138

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 pizzas.pizza_type_id = pizza_types.pizza_type_i
        JOIN
    order_details ON order_details.pizza_id = pizzas.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(order_details.quantity * pizzas.price),2) AS total_revenue
FROM order_details JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id) * 100,2)
AS revenue
FROM pizza_types JOIN pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
JOIN order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category ORDER BY revenue DESC;
```

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

Result Grid | Filter Rows

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

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,
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_details_id
group by orders.order_date) as sales;
```

ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

Result Grid | Filter Rows:

order_date	cum_revenue
2015-01-01	1171.45
2015-01-02	2316.1000000000004
2015-01-03	3433.8
2015-01-04	4341.8
2015-01-05	5247.25
2015-01-06	6299.9
2015-01-07	7284.7
2015-01-08	8542.35
2015-01-09	9570.800000000001
2015-01-10	10748.900000000001
2015-01-11	11616.050000000001
2015-01-12	12484.400000000001
2015-01-13	13331.050000000001

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 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;
```

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

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

SUMMARY

This project, titled Pizza Sales Analysis, is a structured SQL-based analytical project built using a sample pizza sales dataset. It includes four interrelated tables—pizzas, pizza_types, orders, and order_details—and focuses on extracting meaningful business insights through SQL queries. The primary objective was to understand customer preferences, identify best-selling pizzas, evaluate revenue trends, and analyze order patterns based on date, time, and size. Built as a first project by following a tutorial, it helped lay the foundation for working with relational databases, understanding how to join and manipulate multiple tables, and applying real-world SQL logic to solve analytical problems.

THANK YOU FOR ATTENTION

RIYA RAWAT